mi computer

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

TOMO 3

Editorial Delta, S.A.


mi COMPUTER


mi COMPUTER

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones Francisco Martín
Asesor técnico:	Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

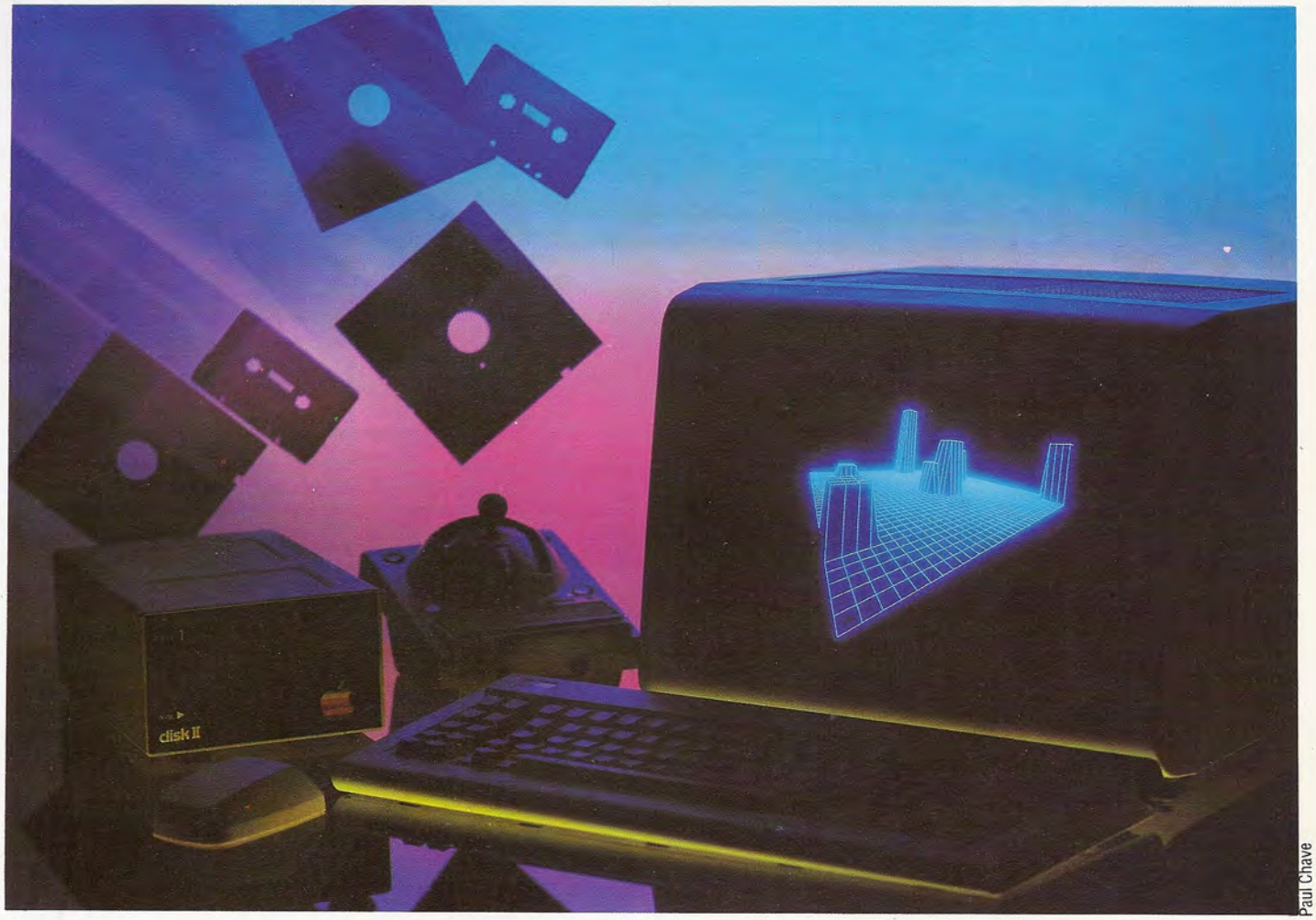
Realización gráfica: Luis F. Balaguer

ISBN: 84-85822-94-3 (tomo 3) 84-85822-82-X (obra completa)
Depósito legal: B. 52/1984
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona)
Impresso en España - Printed in Spain - 1984

mi COMPUTER

VOLUMEN 3

Editorial Delta, S.A.



Paul Chave

Para seguir avanzando

Finalizada la etapa introductoria, iniciamos ahora un auténtico curso avanzado, donde se estudiarán exhaustivamente temas ya tratados y se abordarán otros nuevos

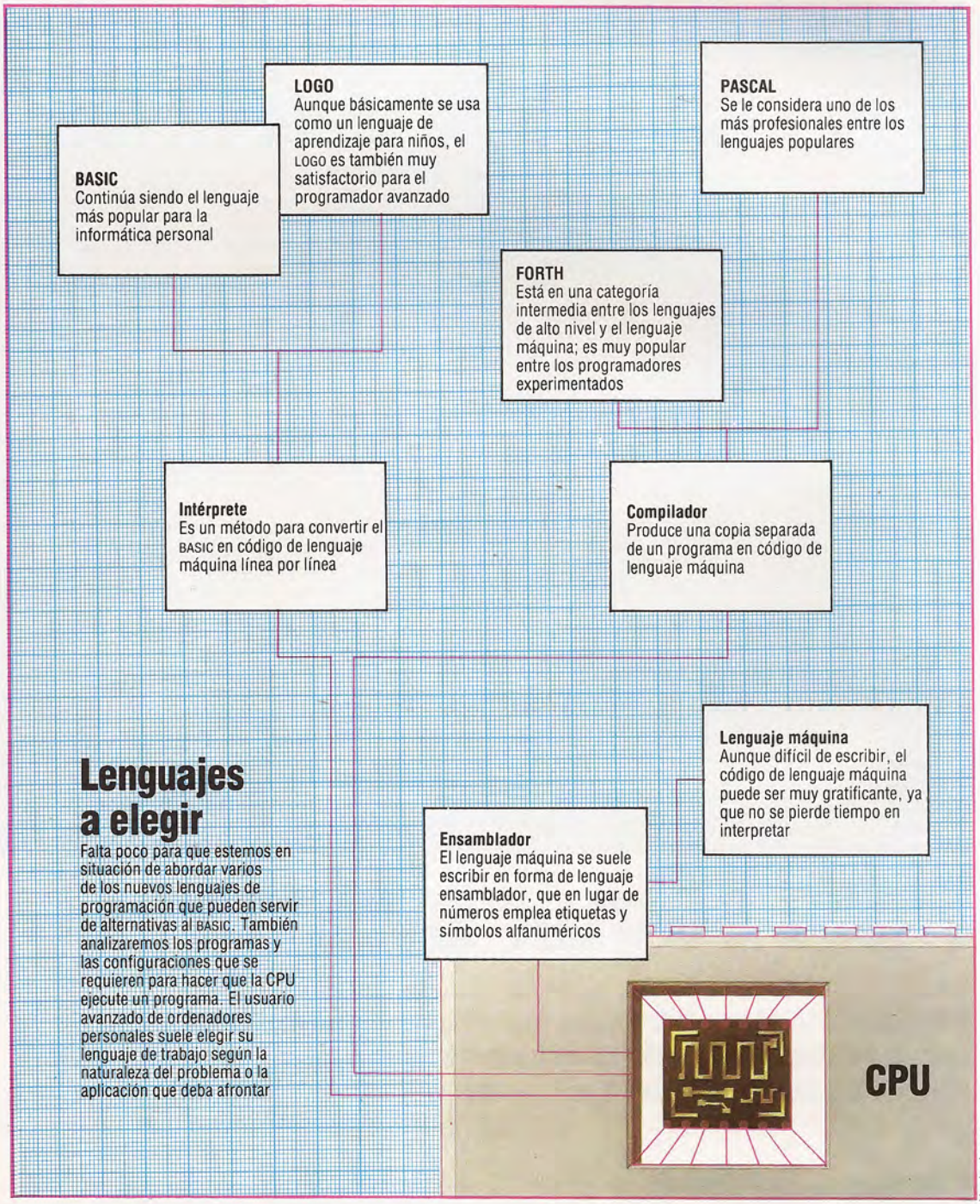
A los profesionales de la informática (operadores, programadores y analistas de sistemas) les resulta bastante difícil mantenerse al día en los últimos adelantos en esta área, pero estas personas al menos pueden esperar el apoyo de sus directores y de los proveedores de los equipos con los cuales trabajan. En cambio, ¿adónde pueden acudir, en busca de una orientación imparcial, los entusiastas de la informática, que aprenden por sí mismos y por sus medios, dedicando a ello su tiempo libre?

¿Qué hacer primero? ¿Ir de tiendas y comprarse un ordenador económico? Hay tantos que resulta difícil decidirse por un modelo determinado sin contar con un consejo de confianza. Y, habiéndose decidido por alguno, ¿qué hacer después? Quizá la máquina elegida ofrezca más de un lenguaje de programación. ¿Cuál de ellos es el más indicado para las necesidades del usuario? ¿Qué paquetes de software ofrecen la mejor relación calidad-precio? ¿Cartuchos o programas en cinta? ¿Debe el usuario adquirir una unidad de disco? ¿O bastará con una unidad de cintas de cassette? ¿Necesita realmente una impresora ahora, o podría esperar hasta que el precio de las impresoras más sofisticadas baje un poco? Si tiene hijos, ¿cómo hacer frente a los constantes deseos de éstos por jugar a un juego detrás de otro, si la motivación principal de la inversión fue favorecer su "alfabetización informática" y ayudarlos en las tareas escolares?

El revolucionario avance de la tecnología del ordenador ha sido tan rápido y se ha introducido hasta tal punto en nuestro entorno cotidiano (el hogar, la oficina, la fábrica y el coche) que muchos de nosotros la hemos rodeado de una cierta aureola mágica.

Es un hecho que cada vez se están entrenando más y más personas para operar ordenadores y terminales de ordenador. Pero existe una importante

El desafío del futuro
En menos de una década, el ordenador se ha convertido en una parte esencial del tejido de nuestra sociedad, y cada año los cambios y los adelantos tecnológicos son más y más amplios y profundos. Mantenerse al corriente de la situación exige un esfuerzo considerable, pero para el recién iniciado en la materia constituye un auténtico desafío. Un curso de estudios a domicilio como el que le viene ofreciendo *Mi Computer*, bien planificado, puede ser una gran ayuda



Kevin Jones

diferencia entre entrenamiento y aprendizaje. Entrenar significa ir conociendo un trabajo de forma mecánica. Aprender supone prepararse para dar un salto más allá de los límites del mero trabajo manual, hacia una amplia comprensión de cómo trabajan los sistemas, de sus potencialidades y de sus limitaciones.

Para muchas personas que trabajan en la industria electrónica o en escuelas y colegios, la respuesta parece ser un curso programado de informática, expuesto de tal modo que resulte comprensible para todos desde el principio. Los manuales de instrucciones individuales para máquinas específicas no pueden proporcionar un panorama equilibrado que permita relacionar un tipo de ordenador con otro. Tampoco reseñarán los escollos ocultos en la multiplicidad de máquinas existentes, ni le aconsejarán cabalmente sobre cómo sacar el máximo provecho de su adquisición. Después de todo, ¿qué clase de fabricante sería aquel que anunciara gratuitamente los productos de sus competidores?

Seguir un curso a domicilio estructurado de manera adecuada, apoyado posiblemente con una clase semanal en una escuela especializada en cursos de informática general y programación de ordenadores, podría ser una forma conveniente y económica de obtener una sólida educación en esta nueva ciencia.

A través de un curso de este tipo, usted no sólo aprendería a programar y a manejar su ordenador personal, sino que adquiriría una visión más amplia de cómo se utilizan los ordenadores en la vida cotidiana. Además de proporcionar una formación en programación y en análisis básico de sistemas, le

ofrecerá un panorama de todos los ordenadores que están en uso en el momento en vez de concentrarse en la máquina que emplea. Debería incluir el estudio de los periféricos y los accesorios que existen para todas ellas, con una explicación de sus principios operativos. Para situar al ordenador en su contexto, se deben examinar en profundidad los campos a los cuales se aplica actualmente y el software que posibilita esas aplicaciones. Por último, el curso debería incluir elementos de lógica formal, de sistemas numéricos y un poco de historia de la informática y de los ordenadores. En resumen, un curso de estudio a domicilio tendrá que cubrir todos los temas que se abarcan en un curso convencional de informática.

A partir de este número nos hemos propuesto estructurar el material para que le resulte un curso de estas características. Basándonos en los conocimientos de BASIC que ha adquirido usted hasta ahora, complementados con lo que ya se expuso sobre gráficos y síntesis de sonido por ordenador, nos hemos propuesto enriquecer sus experiencias en este lenguaje y presentarle los otros lenguajes de alto nivel con que cuentan los microordenadores (PASCAL, FORTH, LOGO y C, p. ej.) al tiempo que le proporcionaremos una buena base sobre programación en código de lenguaje máquina, la llave maestra que abre toda la potencia de un microprocesador.

El conocimiento del código de lenguaje máquina nos permitirá examinar las formas en que se definen los lenguajes de más alto nivel. Entonces, cuando hayamos estudiado cómo trabajan los compiladores y los intérpretes, podremos amalgamar estas dos ramas del conocimiento para empezar a definir nuestro propio lenguaje y escribir un compilador para él.

Esto no quiere decir que abandonemos el BASIC. Hemos de analizar los refinamientos de este lenguaje que nos permitan elaborar después proyectos generadores de software para aplicaciones útiles y de juegos de aventuras y juegos que se basen en pantalla.

Además de las funciones internas del ordenador, exploraremos los métodos para manipulación de archivos, tanto en cinta como en disco flexible, valiéndonos de la experiencia adquirida en cuanto a definir estructuras de datos y jerarquías dentro de la memoria interna del ordenador. De este modo podremos ampliar la capacidad incluso del más pequeño de los ordenadores personales, convirtiéndolo en un serio sistema de procesamiento de la información.

Teniendo siempre presente que no basta con estudiar un tema aislado, consideraremos en profundidad la amplia gama de paquetes de software que existe en la actualidad (hojas electrónicas, procesadores de textos, administradores de bases de datos y similares) con la idea tanto de comprender su funcionamiento y sus métodos como de aprender más acerca de las técnicas de programación profesionales, con el fin de incluir éstas en nuestra propia programación.

Se dedicará alguna atención a la electrónica básica, examinando la función y el diseño de los componentes individuales y de las formas en que éstos se combinan para construir ordenadores y sus periféricos. También volveremos a ocuparnos, ahora ya más detenidamente, de las máquinas en sí mismas: de los microordenadores populares, personales y de gestión, y de sus periféricos, examinando sus especificaciones y evaluando su impacto sobre la informática en general. No olvidaremos tampoco el aspecto humano de la industria del ordenador. Las empresas y las personas que diseñan el software y las que construyen las máquinas, e incluso los usuarios de ordenadores que hayan hecho alguna especial contribución en este campo, tendrán en el curso un lugar dedicado a ellos.

Si le interesa consolidar sus conocimientos acerca de los ordenadores con el fin de ampliar sus perspectivas en el campo laboral, entonces un curso de estudios a domicilio como el que le estamos ofreciendo puede ser el sustituto eficaz del primero-segundo nivel de estudios regulares de informática. Puesto que permite que el estudiante avance a su propio paso, es de igual valor tanto para el alumno rápido como para aquellos que quizá necesiten un poco más de tiempo para captar lo que, al fin y al cabo, es una disciplina compleja.

Por último, si lo que desea es simplemente estar mejor informado acerca de una tecnología que está llamada a cambiar la sociedad en el transcurso de su vida, entonces esta obra le ofrecerá desde ahora una guía exhaustiva. Además de los fundamentos del estudio de la informática, analizaremos el impacto que la nueva tecnología ejercerá sobre toda la sociedad. ¿De qué manera el advenimiento de los ordenadores, al irrumpir en nuestra vida cotidiana, modificará las relaciones entre las personas? ¿Qué cambios políticos tendrán lugar como consecuencia de una "explosión de información" que ha hecho posible el microprocesador de bajo costo? Es difícil dar respuestas razonables a estas preguntas. Los artículos de la prensa y los programas de televisión tienden a trivializarlas; muchas publicaciones especializadas en informática parecen otorgarles mayor complejidad que la que en realidad entrañan. Nosotros seguiremos el camino del medio: le suministraremos los datos necesarios para que elabore sus propias respuestas a estos interrogantes.

Un salto hacia adelante
Su aparición se anunció a la prensa internacional a comienzos de 1984 pero su salida al mercado se planificó para la primavera. El Quantum Leap de Sinclair supone el fin de la larga vinculación de la empresa con el microprocesador Z80. Equipado, en cambio, con una versión del 68000 de Motorola de 32 bits, posee 128 Kbytes de RAM (con más de 512 Kbytes disponibles) y dos microdrives QL incorporados. Sinclair abandona además su característico BASIC de entrada de tecla única

Ian McKinnell

La mejor oferta

Hasta hace poco tiempo, las unidades de disco flexible y las cintas flexibles no estaban al alcance de la mayoría de los usuarios de ordenadores personales, pero hoy la situación ha cambiado

Contacto peligroso

Recuerde no colocar los discos flexibles cerca de objetos imantados. Incluso algo aparentemente tan inocuo como el teléfono contiene electroimanes (se utilizan para hacer sonar el timbre) y hasta el altavoz de un equipo de alta fidelidad doméstico posee unos muy potentes

Los microordenadores son herramientas sumamente versátiles para manipular datos. Sin embargo, la manipulación de información es de poco valor si no se dispone de un medio para almacenarla cuando los datos no se necesitan de momento o cuando se apaga el ordenador. Lo que se puede conseguir de diversas maneras. Quien esté enterado de lo que la informática personal puede hacer se habrá percatado de las limitaciones del cartucho de ROM y de la cinta normal de cassette como métodos de almacenamiento permanente, y deseará conocer esos otros servicios más refinados que sólo los discos magnéticos ofrecen.

Pero antes de analizar los méritos de los discos vamos a considerar los otros sistemas.

El cartucho

Este método de almacenamiento es muy poco útil para el programador. La mayoría de los cartuchos contienen un tipo de PROM (*Programable Read Only Memory*: memoria programable de lectura solamente) que sólo proporciona un medio de dar entrada a los datos en el ordenador, por lo general juegos escritos en un lenguaje máquina extenso y complicado, o facilidades extras como aplicaciones al BASIC. No obstante, los cartuchos pueden contener unas EEPROM (*Electrically Erasable PROM*: PROM que puede borrarse eléctricamente) que se pueden leer y en las que se puede escribir de modo similar a la RAM interna, pero que no son "volátiles", en el sentido de que, cuando se quitan del ordenador o cuando éste se apaga, retienen la información. Igualmente, existen cartuchos para algunos ordenadores que contienen chips de RAM CMOS (*Complementary Metal Oxide Semiconductor*: semiconductor de óxido metálico complementario) de bajo poder, que retienen la información merced a una pila que contienen.

El inconveniente del almacenamiento de EEPROM y RAM CMOS es que son caros.

Ian McKinnell

La unidad de disco del BBC
Para que se puedan utilizar unidades de disco de esta clase con el BBC Modelo B, primero se debe instalar la ROM DOS (*Disk Operating System*: sistema operativo de disco) en la propia máquina. Por el contrario, las unidades del tipo "inteligentes" ya vienen equipadas con un chip DOS

La cinta de cassette

Proporcionadas originalmente porque las unidades de disco eran muy caras, las cintas de cassette siguen siendo, con mucho, el medio de almacenamiento más popular. Por lo general, un reproductor de calidad media será suficiente, aunque en el caso de algunos fabricantes (en especial Commodore y Atari) sólo es posible usar las unidades que ellos han diseñado especialmente.

Los programas y los datos se almacenan en forma binaria como archivos secuenciales a través de la facilidad de grabación normal de la unidad de cassette, utilizando tonos diferentes para representar los 0 y los 1. Por lo general, una información identificada, como puede ser el nombre del archivo (y hasta la dirección de la memoria interna de donde se copia el archivo), se graba primero, seguida del archivo propiamente dicho, de a un bit por vez en bloques de un byte a los que posteriormente se les da formato en segmentos de 256 bytes. Muchos ordenadores incorporan una facilidad para verificación de errores en cada segmento, denominada *suma de control*.

Las órdenes típicas son SAVE, para grabar los archivos, y LOAD, para reproducirlos y recuperarlos. Algunos sistemas proporcionan órdenes adicionales para cassettes para diversas funciones especiales, incluyendo una facilidad para leer una cinta y preparar un catálogo de los nombres de archivos almacenados, y formatos de órdenes para almacenar y recuperar distintos tipos de datos.

El bajo costo y el formato, fácilmente comprensible, de las órdenes para el almacenamiento en cinta de cassette se ven neutralizados por una cantidad de inconvenientes serios:

1. En la mayoría de los casos el usuario opera los mandos de la unidad de cassette de forma manual para almacenar y recuperar, y esto exige una cuidadosa sincronización de la pulsación de los botones y una determinación precisa del volumen.
2. Dado que la información se almacena secuencialmente, la recuperación de un archivo específico (excepto en el caso de la grabadora de cassette Hobbit, controlada por software, y del microcassette incorporado del Epson HX-20) implica o bien la cuidadosa supervisión de un contador de cinta de precisión (¡si es que se dispone de alguno!) para permitir el avance rápido-rebobinado hasta un punto justo antes del archivo deseado, o bien una búsqueda a cargo del ordenador del nombre del ar-

chivo empezando por el comienzo de la cinta. El almacenamiento secuencial también significa que es imposible almacenar eficazmente datos que requieran leerse por secciones pequeñas desde cualquier punto de un archivo sin procesar todo el archivo completo. El tipo de almacenamiento que puede conseguir esto se conoce como *de acceso directo* y es necesario para todo sistema eficaz de archivo de base de datos, como listados de direcciones o entradas de control de existencias.

3. Todo lo anterior, junto con la pequeña cantidad de bits que se almacenan-recuperan por segundo utilizando el almacenamiento en cassette (normalmente, entre 300 y 1 200 bits), supone que un sistema de cinta de cassette sea muy lento.

4. Aun cuando los datos se hayan grabado por primera vez en la forma correcta, éstos se pueden borrar tras haberlos reproducido una cantidad no determinable de veces, debido al desgaste que supone la cabeza de cinta.

5. Las características de las grabadoras varían de un fabricante a otro, y cabe que los datos grabados en un modelo no se puedan reproducir en otro.

El disco flexible

En comparación con los sistemas de almacenamiento en cassette y en cartucho, el almacenamiento en disco ofrece pocos inconvenientes importantes. Las unidades de disco flexible son de construcción compleja y delicada, y bastante caras. Los mismos discos flexibles son, asimismo, costosos. Pero a cambio el usuario consigue un método fiable, cómodo y rápido para almacenar grandes cantidades de datos, operando a una velocidad entre 50 y 200 veces mayor que la del almacenamiento y recuperación en cinta.

Todas las unidades de disco poseen un tipo de sistema operativo de disco (DOS: *Disk Operating System*), con una rutina que formatea la distribución de la información de un disco en pistas. Conviene resumir aquí cuanto fue expuesto en otro lugar de la obra (véanse pp. 324 y 325). Por lo general un disco dispone de 35 a 80 pistas por cada cara y cada pista está dividida en un número variable de arcos llamados sectores. En las pistas más cortas, próximas al centro del disco, hay menos sectores que en las exteriores. Cada sector consta de un bloque de datos, por lo general de 256 bytes.

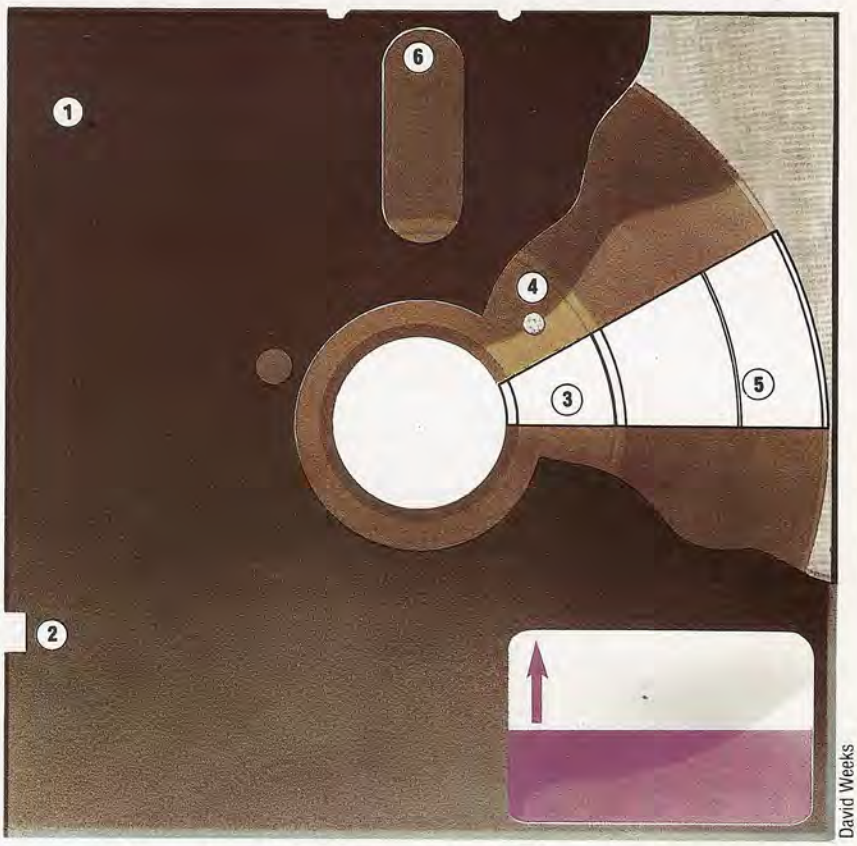
El DOS "recuerda" dónde está almacenada la información que contiene el disco. Esto normalmente se consigue mediante la creación de un mapa de disponibilidad de bloques (BAM: *Block Availability Map*), ya sea almacenado en el disco o retenido en la memoria, y un catálogo o directorio. El BAM contiene un registro de los bloques que ya están en uso y de los que están libres para un nuevo almacenamiento. El catálogo es una lista de los nombres de los archivos, tipos de archivo y las posiciones de pista y sector en que se encuentran. Por lo general se coloca en la pista central y se puede cargar en la memoria del ordenador para su consulta. El DOS posiciona la cabeza de lectura-escritura después de consultar el BAM, y cataloga y organiza el almacenamiento y la recuperación de los datos.

La disposición de la información en pistas y sectores y el posicionamiento exacto de la cabeza de lectura-escritura permite que el DOS ofrezca un tratamiento de acceso directo. Los datos se pueden grabar y extraer en fragmentos tan pequeños como de un byte por vez, si así se requiriera. Grosso modo, las diferencias entre las unidades de disco se reducen a la cantidad de datos que se pueden almacenar (que suelen oscilar entre los 100 y los 400 Kbytes), la velocidad a la cual se pueden transferir los datos, y los medios a través de los cuales el usuario puede controlar el almacenamiento y la recuperación utilizando el DOS.

Existen tres métodos principales para implementar un DOS. El más eficaz consiste en incluirlo en forma de ROM dentro de la unidad de disco, bajo el control del propio microprocesador de la unidad con la RAM asociada. Se dice que esta unidad de disco es una unidad "inteligente": al recibir una instrucción proveniente del procesador central, puede procesar independientemente complejas rutinas para manipulación de disco, permitiendo que el procesador continúe con la ejecución de un programa. Todas las actuales unidades de disco Commodore son inteligentes en este sentido y al operar no utilizan nada de la memoria interna del ordenador.

Un sistema más popular es el tipo que carga el DOS desde el disco a la RAM del ordenador mediante una orden o de forma automática, cuando se enciende el ordenador. El tercer método incluye una forma de DOS en el propio sistema operativo del ordenador. Los Spectrum poseen esta facilidad y la Acorn Computers suministra un DOS para el BBC Micro denominado *Disk Filling System* (sistema de llenado de discos) que proporciona un control limitado sobre el disco. Las rutinas para manipulación de discos incluyen órdenes SAVE y LOAD, una orden CAT (o directorio), una orden para formatear el disco (o cartucho de cinta) y varias órdenes de acceso directo y creación, manipulación y borrado de archivos secuenciales.

El diskette
Los diskettes se componen de Mylar, o una lámina plástica similar, elástica y resistente al desgaste, revestida de un óxido metálico capaz de retener una carga magnética. Encerrado dentro de un contenedor protector cuadrado y de plástico, el disco gira sobre el eje. La cabeza de lectura-escritura accede a la superficie de grabación a través de la ranura que se ve en la parte inferior de la ilustración



David Weeks

1 CONTENEDOR PROTECTOR
2 RANURA DE PROTECCIÓN
3 SECTOR
4 AGUJERO DE ALINEACIÓN
5 PISTA
6 RANURA DE ACCESO

Atacado por hormigas

La importancia de "Ant attack" no reside sólo en la notable calidad de sus gráficos, sino en la sutil aplicación del algoritmo que genera el escenario de la acción, similar a un laberinto

Los escritores y los editores de software jamás han estado de acuerdo con la protección que les brindan las leyes de propiedad intelectual y esta insatisfacción se ha manifestado en los diversos intentos que han realizado con el fin de impedir la copia de sus programas. El autor de este juego, Sandy White, ha intentado, mediante la utilización de otro recurso, evitar que se plagie su obra: ha solicitado patente de privilegio de la técnica de software que produce los gráficos en pantalla. Dado que la Ley de Patentes británica de 1977 niega específicamente la protección de esta clase de programas para ordenador (aduciendo que no se pueden considerar como inventos), se concluye que la patente en cuestión cubre una fórmula matemática o un algoritmo.

Este hecho es interesante, porque no suelen necesitarse algoritmos complejos en un juego de este tipo. ¿Y qué tiene *Ant attack* (Ataque de las hormigas) que exija la creación de nuevas normas encaminadas a la protección de la propiedad intelectual del software?

Ant attack también es inusual en el sentido de que no desciende directamente de ningún juego recreativo. La mayoría de los juegos populares para ordenadores personales tienen sus raíces en las concepciones de Atari, Taito y los otros fabricantes de

1 ¡Mi héroe!
En la primera pasada por el juego, la "víctima" está convenientemente situada junto a las puertas de acceso a la ciudad. Un rápido salto por encima de la muralla protectora y el protagonista (hombre o mujer) es recibido con el grito de "¡Mi héroe, llévame lejos de todo esto!"

2 Situación comprometida
En algunas ocasiones, el hecho de que las hormigas no puedan trepar escaleras resulta de gran utilidad (aunque la razón por la cual nuestro héroe haya subido tan alto aún está por descubrirse). Trepar por obstáculos como éste le permite al protagonista lanzar granadas contra las hormigas guerreras sin temor de que las devuelvan; pero recuerde que está jugando contra reloj

máquinas exclusivas para juegos. A *Ant attack* lo concibió un graduado del Edinburgh College of Art, que ha manifestado su desconocimiento de la tradición de juegos recreativos. Sandy White nunca antes había escrito software para juegos y todos sus esfuerzos por documentarse en este campo se limitaron a preguntarles a sus amigos qué era lo que les gustaba de esa clase de juegos.

Sorprendentemente, su paquete, notablemente vanguardista, fue rechazado por Sinclair Research, quienes no pudieron evaluar la videocinta de *Ant attack* que White les enviara porque, según le dijeron, ¡no tenían ningún aparato de videocassette!

La primera característica novedosa de este juego de laberinto con que se encontrará el usuario es que permite que el jugador escoja el sexo del protagonista principal. Y pisándole los talones viene el primer descuido. Sin tener en consideración el sexo elegido por el usuario, el fotograma que abre el juego, que lo sitúa en el escenario en unas 30 palabras o así, le explica cómo ha llegado a sus oídos una llamada de socorro "irresistible para un héroe como usted".

El protagonista, acosado por monstruosas hormigas, se puede defender solo (o, por supuesto, sola) arrojando granadas. Lamentablemente, no hay coherencia en cuanto al efecto que producen estas granadas en las hormigas. Si bien esto podría ser resultado de un factor de azar deliberado, es

Cortesía de Soft

Ian McKinnell

más probable que sea consecuencia de una programación indiscriminada. Desplazar al protagonista 90 grados en sentido contrario a las agujas del reloj se consigue pulsando la tecla M del Spectrum y la tecla contigua, SYMBOL SHIFT, hace girar la figura en sentido contrario. Las teclas tipo membrana de plástico moldeado del Spectrum no proporcionan control adecuado sobre esta transformación, que invariablemente produce frustración en el jugador.

Al parecer, *Ant attack* se desarrolló antes de que se lanzara al mercado la interface 2 de Sinclair, que acepta dos palancas de mando estándar de Atari. El juego mejoraría muchísimo si se lo actualizara para utilizar estos periféricos, si bien necesitaría dos palancas de mando para manipular la estructura de órdenes.

Además de hacer girar el distintivo, desplazarlo hacia adelante, hacerlo saltar o arrojar granadas (se puede escoger entre cuatro distancias de lanzamiento), el jugador puede elegir uno de cuatro puntos de vista, cada uno de ellos centrado en el distintivo.

Es esta sección de la generación de gráficos del programa lo que lo diferencia de la mayoría de los otros juegos que ocupan menos de 48 Kbytes. La transformación es virtualmente instantánea, eclipsando por completo la ejecución normal de los generadores de gráficos tridimensionales que existen para el Spectrum. La capacidad de cambiar los puntos de vista es esencial para el juego. Sin ella, una considerable porción del campo de juego quedaría a menudo oculta a la vista.

Comprensiblemente, el autor se muestra reacio a revelar demasiadas cosas acerca de los métodos de trabajo que han adoptado él y su colaboradora Angela Sutherland. No obstante, ha dado a entender que el campo de juego no está retenido, como uno esperaría, como una matriz de 128 × 128 × 6. La prueba de ello es evidente si, en vez de entrar en la ciudad, se hace que el distintivo del jugador gire en redondo y se encamine hacia el desierto. Después de una breve caminata, él o ella llegan a otra ciudad, y luego a otra, y así sucesivamente.

Y así llegamos al objetivo del juego en sí mismo. Éste está ambientado en la ciudad de Antescher (así llamada como homenaje de los autores del juego al artista y diseñador holandés M. C. Escher, quien dibujó ingeniosas estructuras ilusorias que en realidad eran imposibles de construir). Parado frente a sus puertas, el jugador escucha los gritos de una persona en peligro, salta por encima de la muralla baja y cae dentro de la ciudad, donde emprende la búsqueda de la víctima, saltando obstáculos o girando para evitarlos y seguir adelante. La ciudad aparece en proyección isométrica y no se realiza ningún intento por ser fiel a la perspectiva.

Sólo una pequeña porción de la ciudad está a la vista en cada momento y los fotogramas se mueven a lo ancho a medida que la figura se desplaza hacia la izquierda, la derecha, arriba o abajo. El *scrolling* es excelente, al igual que la animación de las figuras. Es también notable el sentido del humor de que se hace gala en la animación.

Enseguida se hace evidente que la ciudad está habitada por inmensas hormigas cuya mordedura, aunque no inmediatamente fatal, provoca la muerte si se recibe una cantidad suficiente. Si una hormiga se percata de la presencia del jugador se dispone a seguirlo. Con cierta dosis de habilidad se la podrá quitar de encima, de lo contrario tendrá que recurrir a la nada fiable granada. Y, cuidado, no vaya a lanzarla contra la pared que tiene inmediatamente delante, porque podría volar usted por los aires.

En la primera pasada a través del juego, la figura a rescatar está a la vista, al otro lado de la entrada. En las sucesivas pasadas se va haciendo más difícil de hallar y más difícil de alcanzar. Invariablemente está situada sobre el nivel del suelo. El rescatador puede saltar sólo un nivel cada vez, de modo que si la víctima no está directamente accesible desde el suelo (mediante una escalera, p. ej.), el rescatador tiene un auténtico problema. La única forma es esperar a que las hormigas ataquen en un punto apropiado, paralizar una y saltar sobre el insecto, utilizando su cuerpo como primer peldaño hacia arriba.

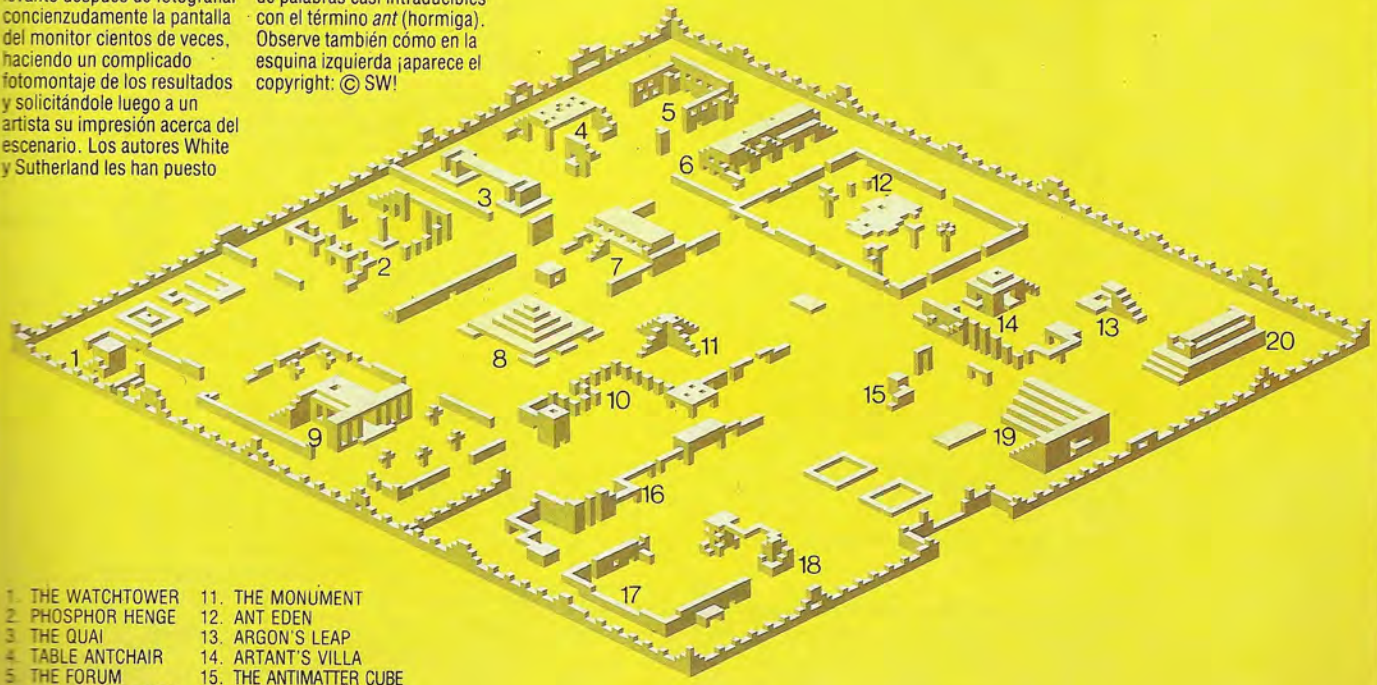
El rescatador también puede, de esta forma, "poner una pierna" sobre la víctima, en caso de que fuera necesario (las hormigas nunca atacarán a la víctima). La pasada termina cuando tanto el rescatador como la víctima están fuera de la ciudad.

Ian McKinnell

Los creadores del juego
Ant attack fue el primer intento de su autor, Sandy White, por escribir software comercial. Sandy, que sólo tenía 23 años cuando el paquete salió al mercado por primera vez, a fines de 1983, acababa de graduarse en el Edinburgh College of Art con una titulación en escultura cuando concibió la idea de crear un programa de juegos para microordenadores personales. Una amiga suya, Angela Sutherland, colaboró en el diseño de las estructuras que componen la ciudad de Antescher

El enigma de los arenales

Este plano general de la ciudad de Antescher se levantó después de fotografiar concienzudamente la pantalla del monitor cientos de veces, haciendo un complicado fotomontaje de los resultados y solicitándole luego a un artista su impresión acerca del escenario. Los autores White y Sutherland les han puesto nombre a las principales estructuras, realizando juegos de palabras casi intraducibles con el término *ant* (hormiga). Observe también cómo en la esquina izquierda ¡aparece el copyright: © SW!



1. THE WATCHTOWER
2. PHOSPHOR HENGE
3. THE QUAI
4. TABLE ANTCHAIR
5. THE FORUM
6. THE ANTICHAMBER
7. SKAZ YANDOR
8. THE PYRAMID
9. THE ANCIENT
10. OXYMINE
11. THE MONUMENT
12. ANT EDEN
13. ARGON'S LEAP
14. ARTANT'S VILLA
15. THE ANTIMATTER CUBE
16. DROXTRAP
17. ADRIANT'S WALL
18. BONZAI WALK
19. THE SQUARENA
20. THE CRYPT

Cortesía de Your Spectrum

Kevin Jones

Álgebra para la toma de decisiones

Los ordenadores efectúan sus funciones haciendo pasar una serie de voltajes altos o bajos por los circuitos electrónicos. Estos voltajes pueden representarse por los bits 1 y 0

El álgebra de Boole, rama de las matemáticas que aplica la lógica verdadero-falso, es la base teórica a partir de la cual se realiza físicamente la arquitectura del ordenador. Los conceptos y las leyes del álgebra booleana son pocos y fáciles de comprender.

En estas páginas estudiaremos con detalle los aspectos teóricos y prácticos del diseño de circuitos lógicos, con ejemplos de circuitos básicos que son los que trabajan dentro de un ordenador personal. Las leyes del álgebra booleana se basan en tres operaciones lógicas simples: AND, OR y NOT (véanse pp. 68 y 69). Estas tres operaciones lógicas siguen muy de cerca la forma en que se emplean dichas tres partículas en el lenguaje cotidiano. Observemos esta oración:

> Si hace buen tiempo AND(y) es sábado, David saldrá a dar un paseo.

Que David salga o no a dar un paseo depende de dos cosas: que haga buen tiempo y que sea sábado. Para tomar una decisión respecto a si salir o no de paseo, David considerará si los enunciados "hace buen tiempo" y "es sábado" son verdaderos o falsos. Cuatro son las posibles combinaciones y sólo una de ellas hará que David salga a dar un paseo. Una tabla que muestre todas las combinaciones posibles de una serie de enunciados es una *tabla de verdad*. He aquí la tabla de verdad para nuestra AND lógica:

Hace buen tiempo	Es sábado	David saldrá a dar un paseo
FALSO	FALSO	FALSO
FALSO	VERDADERO	FALSO
VERDADERO	FALSO	FALSO
VERDADERO	VERDADERO	VERDADERO

Un proceso similar ilustrará la función de la operación lógica OR. Consideremos esta oración:

> Si José OR(o) Ana van al partido, Juan irá también.

Nuevamente hay dos condiciones que determinarán que Juan vaya o no al partido: que José pueda ir, *o* que Ana pueda ir. Al igual que con AND, podemos construir una tabla de verdad para el nexo OR. Dado que hay dos condiciones, cada una de las cuales puede ser verdadera o falsa, otra vez son cuatro las combinaciones posibles. La tabla de verdad para la enunciación será la siguiente:

José irá	Ana irá	Juan irá al partido
FALSO	FALSO	FALSO
FALSO	VERDADERO	VERDADERO
VERDADERO	FALSO	VERDADERO
VERDADERO	VERDADERO	VERDADERO

La tercera operación lógica (NOT) realiza una función muy sencilla. Consideremos esta proposición:

> Si NOT(no) está oscuro, saldré.

Esta vez la única condición a considerar es si está oscuro. Esto puede ser verdadero o falso y, por tanto, para nuestra tabla de verdad sólo hay dos condiciones posibles:

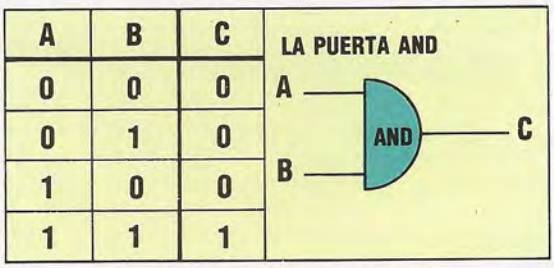
Está oscuro	Saldré
FALSO	VERDADERO
VERDADERO	FALSO

Puertas lógicas

Los dispositivos electrónicos que componen los circuitos lógicos de un ordenador se denominan *puertas lógicas*. Las tres puertas lógicas más simples imitan la función de las operaciones lógicas AND, OR y NOT. Estas puertas funcionan representando una condición VERDADERO mediante el dígito binario 1, y la condición FALSO mediante el dígito binario 0. Así, para cada puerta lógica podemos construir una tabla de verdad con todas las combinaciones de entradas junto con la salida resultante.

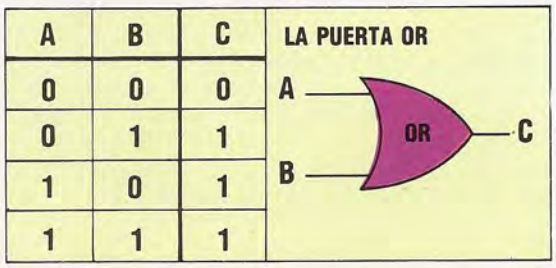
La tabla de verdad y el diagrama para la puerta ADN con entradas A, B y salida C son:

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1



La función de la puerta AND se puede describir en palabras como: "la salida será 1 si ambas entradas son 1; si no, será 0". La notación booleana para la salida de una puerta AND es A.B. La tabla de verdad y el diagrama para la puerta OR son:

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

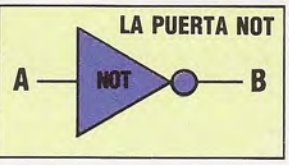


Liz Dixon

La puerta OR se puede resumir con la siguiente frase: "la salida será 1 si alguna de las entradas, o ambas, es 1". La expresión booleana para la salida de una puerta OR es A+B.

A diferencia de AND y OR, la puerta NOT sólo posee una entrada y una salida. Su tabla de verdad es la más simple de las tres:

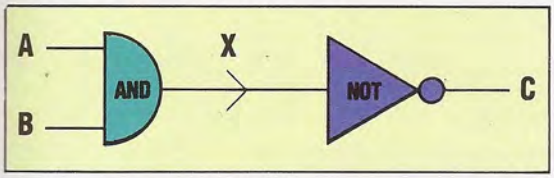
A	B
0	1
1	0



La puerta NOT se resume así: "la salida será lo contrario de la entrada". La expresión booleana para la salida de una puerta NOT es $\bar{A}$.

Combinando puertas lógicas

Podemos unir entre sí varias puertas lógicas para obtener circuitos lógicos secuenciales y combinados. Éstos, a su vez, se combinan para producir la arquitectura del ordenador. Todo circuito lógico se puede representar mediante una tabla de verdad que describa qué salida se puede esperar para cualquier posible combinación de entradas. Observemos este circuito lógico sencillo:

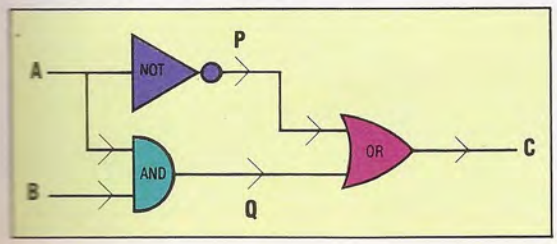


En este circuito hay dos entradas, A y B, y una salida, C. Para ayudar a construir la tabla de verdad para el circuito, hemos llamado X a la salida de la primera puerta. Como hay *dos* entradas para el circuito, ello significa que hay cuatro posibles combinaciones de entradas (o sea, 2^2).

A	B	X	C
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

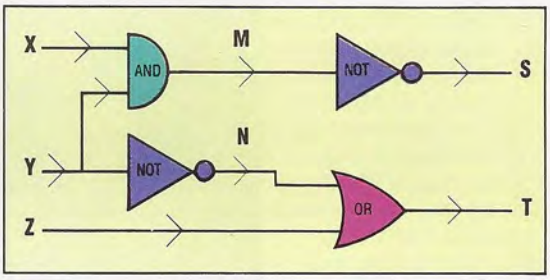
La salida de la puerta AND, X, se hace pasar por la puerta NOT para producir la salida final, C.

He aquí un circuito más complicado y su tabla de verdad. Como sólo hay dos entradas, las combinaciones de entradas posibles siguen siendo cuatro (2^2). La segunda mitad de esta tabla de verdad (las columnas P, Q y C) es un acomodo de parte de una tabla de verdad para puerta OR.



A	B	P	Q	C
0	0	1	0	1
0	1	1	0	1
1	0	0	0	0
1	1	0	1	1

La utilización de tablas de verdad no se limita a circuitos de dos entradas y una salida, sino que se pueden ampliar para cualquier circuito. Veamos a continuación un ejemplo de un circuito de tres entradas y dos salidas.



Como en este circuito hay *tres* entradas, debemos considerar ocho posibles combinaciones (o sea, 2^3):

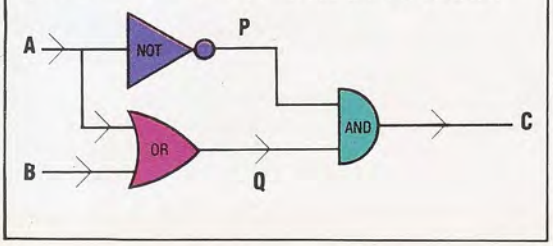
X	Y	Z	M	N	S	T
0	0	0	0	1	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	0	0	0
1	1	1	1	0	0	1

EJERCICIO 1

1) Construir una tabla de verdad para la siguiente situación: "María podría conducir un coche si hubiera aprobado su examen de conducir O (OR) si fuera acompañada por un conductor cualificado".

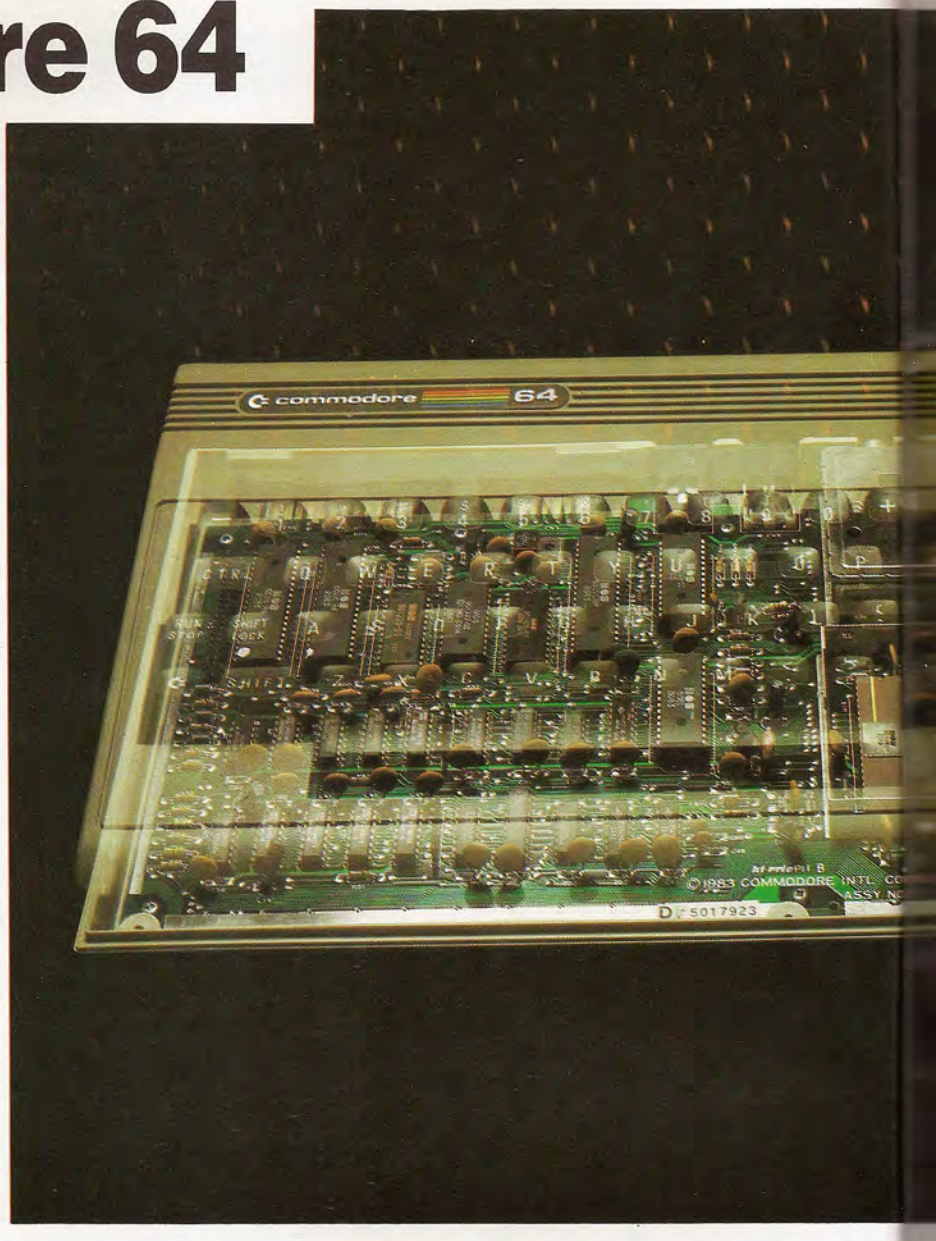
2) Construir una tabla de verdad para esta situación: "Se puede cargar un programa en un ordenador si se dispone de una grabadora de cassette O (OR) de una unidad de disco Y (AND) si el programa NO (NOT) está escrito para ser ejecutado en otra máquina".

3) Construir una tabla de verdad para el circuito:



Commodore 64

Máquina ideal para los amantes de los ordenadores personales, también se puede utilizar para aplicaciones de pequeñas empresas



El parecido físico entre el Commodore 64 y el Vic-20 es engañoso. Aunque entre ambos existe cierta compatibilidad de software, en cuanto al hardware el 64 significa un avance considerable. Comencemos por analizar los 64 Kbytes de RAM, configuración a la que alude el nombre de la máquina. Como oferta de mercado es una ventaja, ya que hasta el advenimiento del microprocesador de 16 bits, ésta era la mayor RAM de que disponían los microordenadores de gestión empresarial. No obstante, equipar un ordenador personal con tal cantidad de memoria entraña ciertas dificultades. Aunque un microprocesador de ocho bits como el 6502, que tanto se utiliza, puede direccionar un total de 64 Kbytes, en éstos se incluyen, además de la RAM, la totalidad de la ROM, y también los chips de entrada-salida con objeto de controlar el teclado, la pantalla y los periféricos.

La solución estaba en el *conmutador de banco*, una técnica con la que las secciones de memoria se encienden y se apagan en el mapa direccionable de memoria a medida que se las necesita. No existe ningún límite teórico en cuanto a la cantidad total de memoria que un ordenador puede incorporar utilizando este método, pero dado que el microprocesador aún puede direccionar sólo 64 Kbytes a la vez, cuanta más memoria haya, más conmutación de bancos se necesitará, como consecuencia de lo cual se reducirá la eficacia.

Caja de sorpresas

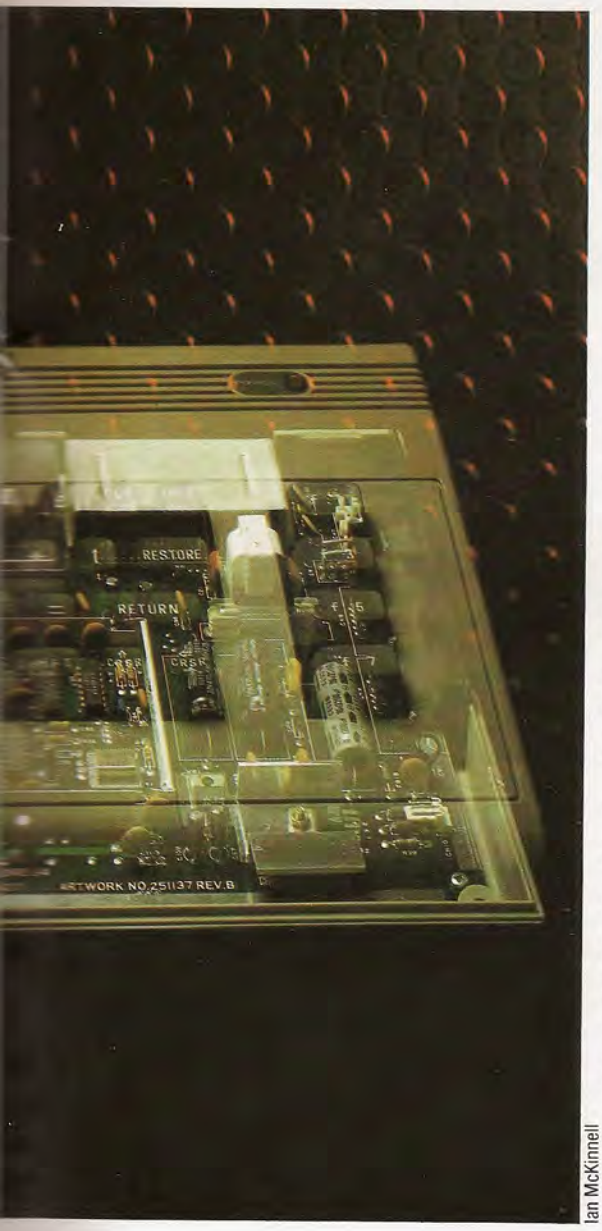
El SX-64 es una versión autónoma y portátil del Commodore 64 y se puede adquirir con diversas configuraciones. La versión más popular incorpora una unidad de disco (el espacio superior se puede utilizar para almacenar diskettes) y un monitor en color de cinco pulgadas. El SX-64 puede ejecutar, sin ninguna modificación, software basado en cartucho o en disco para el Commodore 64 estándar.

En muchos sentidos se trata de uno de los mejores ordenadores "de maleta", término que se acuñó para distinguir a este tipo de ordenadores de las verdaderas máquinas portátiles, como el Epson HX-20 y el Tandy Modelo 100. El teclado incorpora teclas totalmente esculpidas con los letreros de gráficos inscritos en el frente, y se puede separar de la unidad principal. En la parte superior de la carcasa hay una ranura para albergar los cartuchos de ROM; cuando no está ocupada por ningún cartucho, la abertura se cubre con una tapa para evitar el polvo.

La carcasa es a la vez robusta y compacta y se parece al equipo portátil de comprobaciones que utilizan los ingenieros de reparaciones, en especial porque el asa para transportarla sirve al mismo tiempo como soporte. El asa tiene estrías para evitar que resbale sobre el escritorio, aunque el transportarla es algo incómodo. En líneas generales, el diseño físico es el mejor que ha producido la Commodore hasta la fecha, empañado sólo por el pequeño detalle del cable y el enchufe para la red eléctrica que no se pueden guardar en ningún sitio dentro de la carcasa

SX-64, cortesía de Commodore

Ian McKinnell



Ian McKinnell

En el caso concreto del Commodore 64, ello significa que si usted desea ejecutar un programa en BASIC, será necesario que se enciendan las ROM que contienen el intérprete del lenguaje, lo que reduce la cantidad de RAM disponible a 40 Kbytes (parte de los cuales aún se tendrán también que destinar a las variables del sistema y a la RAM de pantalla).

Aunque el conmutador de banco se ha incluido en muy pocos ordenadores personales con alguna modificación, en el Commodore 64 se obtiene mediante la utilización de un microprocesador especial. El 6510 es muy similar al 6502, de reconocida popularidad en el diseño de ordenadores personales. El conjunto de instrucciones es idéntico e incorpora un bus de datos de ocho bits, un bus de direcciones de 16 bits y diversas señales de control. Sin embargo, también ostenta una puerta programable de entrada-salida de ocho bits. Esto significa que en el chip hay ocho patillas adicionales, cada una de las cuales se puede establecer en 1 o 0, o se pueden utilizar para leer los valores colocados en ellas mediante un dispositivo externo. Normalmente este tipo de puertas se implementan mediante un chip especial (denominado PIO, PIA o VIA, según el fabricante), y un ordenador personal corriente incluye varios de éstos para manipular las puertas de los periféricos y el teclado.

A la puerta le corresponden en el mapa las dos posiciones de memoria más bajas ($0000 y $0001). La primera es para leer y escribir los bits individuales, mientras que la segunda indica si cada bit se establece como una entrada o como una salida. Tener esta puerta incorporada en el microprocesador significa que el 6510 sería ideal para incorporarlo a numerosos dispositivos domésticos, desde el lavavajillas hasta los juguetes programables. En el Commodore 64 se utiliza para seleccionar los bancos de memoria. Se podría hacer lo mismo con sentencias POKE en BASIC, pero actuando de este modo existe una clara posibilidad de "reventar" el sistema, lo que obligaría a tener que recomponer a continuación su ordenador. Por consiguiente, la mayoría de las conmutaciones de memoria se realizan en código de lenguaje máquina.

COMMODORE 64

MEDIDAS

404 x 216 x 75 mm

CPU

6510

MEMORIA

64 K de RAM, de los cuales hay 39 K disponibles para programas en BASIC.
20 K de ROM, incluyendo el generador de caracteres

PANTALLA

25 filas por 40 columnas. En baja resolución hay 16 colores disponibles desde teclado para caracteres, borde y fondo. El máximo en alta resolución es 320 x 200 pixels. Se pueden definir y utilizar hasta ocho sprites

INTERFACES

Palancas de mando (2) más lápiz óptico, RS232 (se necesita adaptador), en paralelo de 8 bits, cassette, en serie (para disco e impresora), monitor compuesto, entrada y salida de audio, TV, cartuchos

LENGUAJES DISPONIBLES

BASIC, FORTH, LOGO, lenguaje ensamblador 6502.

TECLADO

Estilo máquina de escribir, con teclas de cursor y cuatro teclas de función programable

DOCUMENTACION

El ordenador viene con un manual de instrucciones adecuado, pero para obtener el máximo provecho de sus configuraciones debe adquirir la *Guía de referencia para el programador*, o alguna de las muchas guías independientes que se han publicado para el Commodore 64

VENTAJAS

Mucha memoria estándar. Gráficos sprite. Sofisticado control de sonido. Teclado de calidad. Buena gama de periféricos. Dispone de más software de gestión que la mayoría de los ordenadores personales

INCONVENIENTES

Requiere la unidad para cassette del fabricante. BASIC pobre en órdenes útiles (a menos que compre un cartucho accesorio). Selección limitada de modalidades para gráficos y de resoluciones. Unidad de disco lenta

Posibilidades de gestión

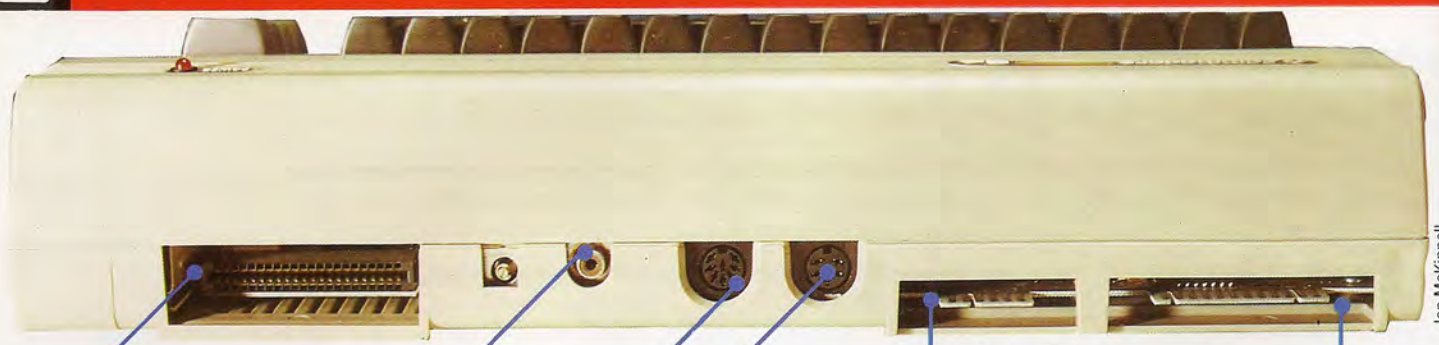
Se puede afirmar que el Commodore 64 ha heredado de sus predecesores, el PET y el Vic-20, una gran proporción de su software de base. El intérprete de BASIC es muy parecido en las tres máquinas y también existen grandes similitudes en cuanto a los sistemas operativos de disco. Dado que el software de gestión desarrollado para la gama PET sólo se podía utilizar con máquinas Commodore, no es nada sorprendente que los productores de software fueran tan rápidos en sacar partido del nuevo mercado potencial que abrió el 64.

Para aplicaciones de gestión empresarial existe una amplia selección de paquetes para tratamiento de textos, varios de los cuales poseen verificadores de ortografía. Dos de los ejemplos más populares son el EasyWrite/EasySpell de Commodore y el VizaWrite/VizaSpell. Otros dos paquetes populares, aunque sin la opción de ortografía, son el Paperclip 64 y el Wordcraft 40. Este último es distinto de la mayoría de los paquetes de tratamiento de textos, por el hecho de que la pantalla visualiza el texto en el formato en el que éste se imprimirá finalmente, mientras que la mayoría de los otros visualizan "controles encajados", es decir, símbolos compuestos por un único carácter para significar un retorno del carro o que el título se ha de centrar en la página.

Existen hojas electrónicas y, entre ellas, un paquete que merece una mención especial: el CalcResult; es más caro que la mayoría de las hojas electrónicas para micros de precio económico, pero es a todo color, incluye una capacidad para visualizar gráficos de las cifras en cualquier columna de la hoja electrónica y trabaja en tres dimensiones. Todo ello quiere decir que se pueden retener simultáneamente en la memoria varias páginas de memoria y que se pueden sumar entre sí cifras de todas estas hojas.

Magpie es también un programa sobresaliente, que entra en la categoría de los generadores de aplicaciones. Una aplicación se define dibujando los trazados para los registros de pantalla y las formas impresas en la pantalla, y especificando luego las relaciones entre los campos en aquellos documentos: IVA = TOTAL* 15 %, por ejemplo

Ian McKinnell



Ian McKinnell

Conexión cartuchos
Si se enchufa aquí un cartucho ROM (de hasta 16 Kbytes), anulará efectivamente cualquier otra memoria que ocupe las mismas posiciones. Si los primeros nueve bytes de la ROM contienen una secuencia específica de valores, entonces el programa "comenzará automáticamente" cuando se encienda. Así funcionan los cartuchos para juegos

Salida de TV
A diferencia del Vic-20, el Commodore 64 contiene un modulador de RF incorporado, de modo que la salida se puede conectar directamente a un televisor

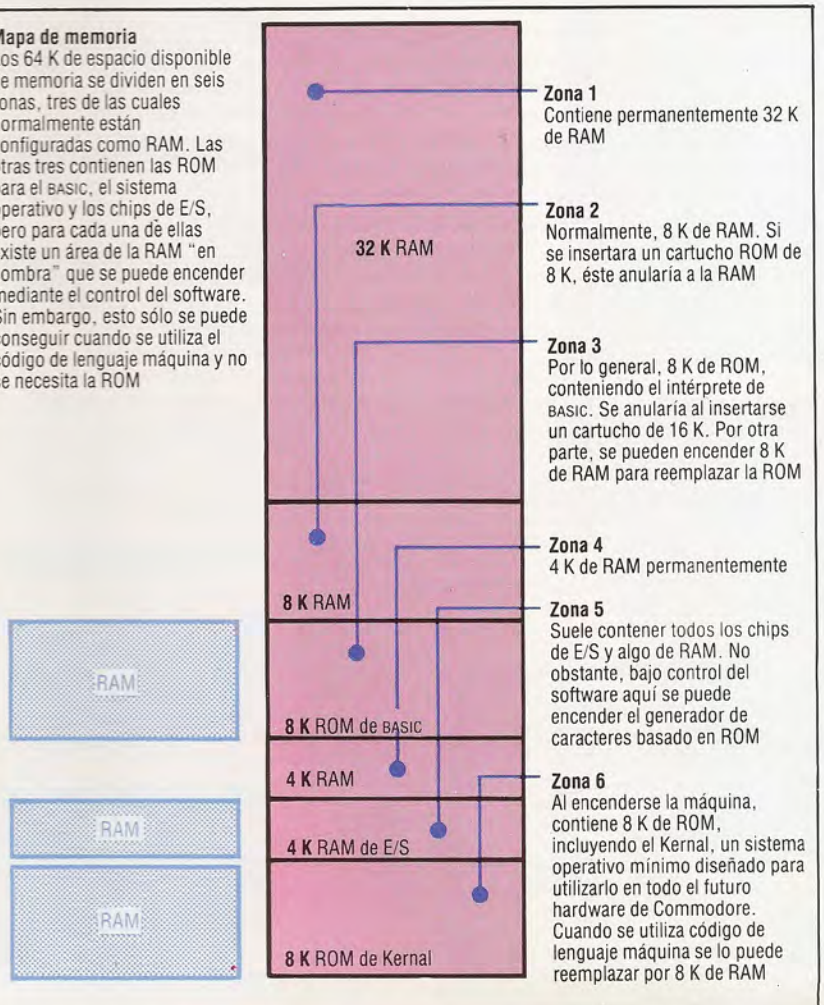
Conexión audio-video
Se proporciona una señal de video compuesta para activar un monitor en color (aunque no un monitor RGB), y hay una salida de audio separada que se puede conectar con un sistema de alta fidelidad. También hay una línea de entrada de audio que le permite al usuario mezclar música grabada con sonidos sintetizados

Bus en serie
Ésta es una interface especial diseñada por la Commodore para activar varios dispositivos (incluyendo sus discos e impresoras) simultáneamente. El protocolo es similar al de la IEEE488 estándar, con la excepción de que sólo hay una línea de datos (en serie) en lugar de ocho líneas en paralelo

Conexión cassette
Todos los ordenadores Commodore requieren la unidad para cassette del fabricante. Cuando se comercializó por primera vez, el sistema Commodore era más rápido y más fiable que las unidades domésticas. Ahora sucede todo lo contrario

Conexión dispositivos
Esta conexión posee dos funciones. En primer lugar, puede recibir una interface en serie RS232 completa, si bien se requiere un accesorio para convertir los voltajes del 64 a los que utilizan la mayoría de los otros dispositivos en serie. Al mismo tiempo, puede funcionar como una puerta en paralelo que se puede utilizar para experimentación

Mapa de memoria
Los 64 K de espacio disponible de memoria se dividen en seis zonas, tres de las cuales normalmente están configuradas como RAM. Las otras tres contienen las ROM para el BASIC, el sistema operativo y los chips de E/S, pero para cada una de ellas existe un área de la RAM "en sombra" que se puede encender mediante el control del software. Sin embargo, esto sólo se puede conseguir cuando se utiliza el código de lenguaje máquina y no se necesita la ROM



Zona 1
Contiene permanentemente 32 K de RAM

Zona 2
Normalmente, 8 K de RAM. Si se insertara un cartucho ROM de 8 K, éste anularía a la RAM

Zona 3
Por lo general, 8 K de ROM, conteniendo el intérprete de BASIC. Se anularía al insertarse un cartucho de 16 K. Por otra parte, se pueden encender 8 K de RAM para reemplazar la ROM

Zona 4
4 K de RAM permanentemente

Zona 5
Suele contener todos los chips de E/S y algo de RAM. No obstante, bajo control del software aquí se puede encender el generador de caracteres basado en ROM

Zona 6
Al encenderse la máquina, contiene 8 K de ROM, incluyendo el Kernal, un sistema operativo mínimo diseñado para utilizarlo en todo el futuro hardware de Commodore. Cuando se utiliza código de lenguaje máquina se lo puede reemplazar por 8 K de RAM

Otros tres chips se encargan del resto de las configuraciones del 64. Hay un CIA *(Complex Interface Adaptor)* 6526, que es una versión más refinada de los PIA y VIA que hemos mencionado anteriormente. Además de las habituales líneas programables de entrada-salida, incluye sincronizadores y registros de desplazamientos para la conversión en paralelo de datos en serie y viceversa. Hay también un reloj de 24 horas con una alarma programable, que el intérprete de BASIC parece no utilizar en absoluto.

La visualización de gráficos y de video la manipula otro chip, el 6566, que es un nuevo desarrollo del *Video Interface Chip* (chip para interface de video), del cual el Commodore Vic-20 ha tomado su nombre. Éste ofrece distintas modalidades para las visualizaciones tanto de texto como de gráficos de alta resolución, y los gráficos sprite se han documentado muy bien. Aunque sólo puede manipular ocho sprites a la vez (frente a los 32 del Memotech MTX512, p. ej.), se pueden imitar bastantes más. Los sprites se definen en la memoria como un bloque de bytes y su posición se indica "colocando" (POKE) la dirección en los registros del chip Vic-II. Resulta relativamente sencillo conmutar el señalador rápida y repetidamente entre diferentes conjuntos de valores para simular de este modo más de ocho unidades.

El chip 6581 es conocido por SID (*Sound Interface Device*: dispositivo para interface de sonido), y contiene funciones mucho más avanzadas que algunos de los primeros sintetizadores de música diseñados especialmente. Además del control total de ADSR sobre la envoltura de volumen de cada sonido, las funciones que desempeña este chip incluyen la filtración, distintas formas de onda y modulación circular (es decir, la modificación de un sonido mediante otro).

Paso a paso

Para hablar con un ordenador hay que tener las ideas muy claras. La técnica de diagramación facilita esta tarea

Usted ya sabe en qué consiste un programa y las múltiples aplicaciones que tiene. Le falta por conocer en profundidad *cómo* se construyen, es decir, cuál es la lógica interna que sostiene a los programas. Una serie de capítulos que comienza con el presente le enseñará a analizar un problema y a visualizar los pasos que usted daría hasta llegar a la solución. Esta visualización mediante símbolos gráficos de uso universal es lo que se denomina *técnica de diagramación (flow-charting)*.

Cualquier problema, desde que se plantea hasta que se resuelve en forma de programa apto para introducirlo en el ordenador, debe pasar por una serie de fases que se podrían sintetizar en tres:

1. Análisis del problema y su resolución. Se entiende por análisis el estudio de los elementos que forman parte del planteamiento, así como cuándo y de qué manera tales elementos intervienen en lo que acaba siendo una solución lógica del problema, bien sea mediante un desarrollo puramente mental o con la utilización de notas y apuntes. La construcción se basa en un conjunto de ideas que, secuencialmente seguidas, llevan a una solución.
2. Diagrama: representación gráfica del análisis.
3. Codificación. Obtiene el programa, es decir, transcribe el análisis en el lenguaje informático usado (BASIC, PASCAL, etc.).

Esta tercera fase depende siempre de las dos previas. Sin duda la primera fase (el análisis) es completamente imprescindible, pues constituye la solución lógica que sólo la mente puede elaborar, mientras que la segunda facilita la programación.

La diagramación podría definirse como el medio que facilita el análisis de las aplicaciones mediante figuras geométricas simbólicas que representan las diferentes fases de la solución de un problema.

Se usan estos diseños gráficos porque resultan más claros y simples a la hora de un seguimiento que si la resolución se describiera a base de referir todos los pasos con un texto convencional.

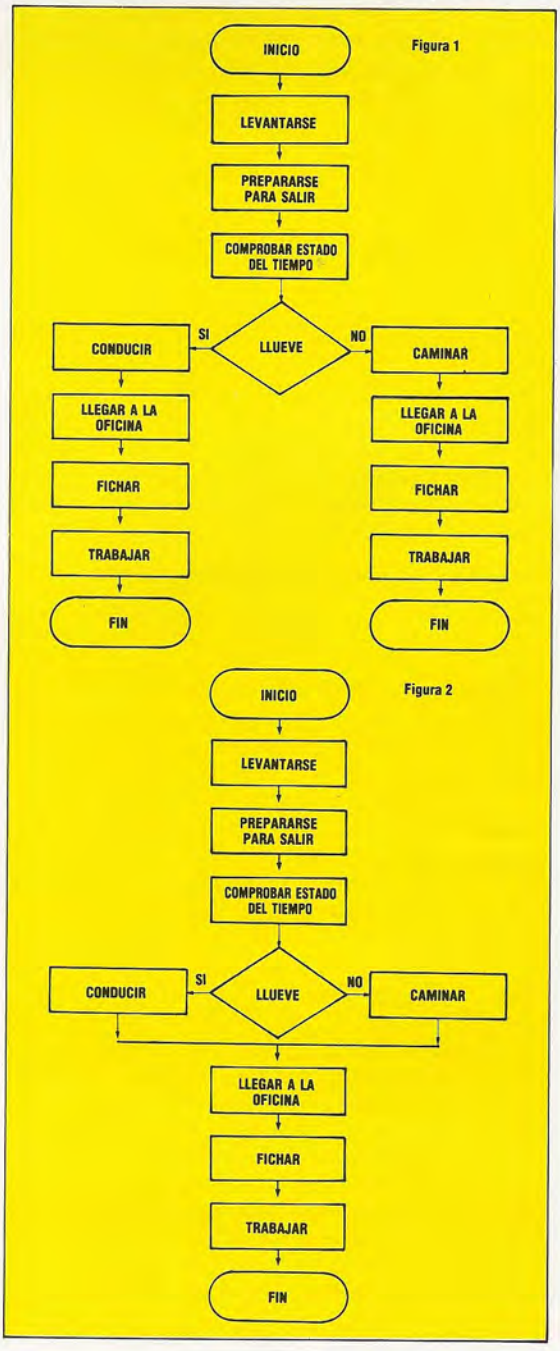
Todo proceso encaminado a resolver un problema se compone de una secuencia determinada de fases elementales, simbolizables mediante diagramas por complejo que sea el problema.

La figura 1 es la representación gráfica de este sencillo ejemplo: "Juan realiza cada mañana los siguientes actos: se levanta, se prepara para salir y comprueba qué tiempo hace. Si llueve, va en coche hasta la oficina, llega, ficha y comienza a trabajar. Mientras que si el tiempo es bueno camina hasta la oficina, donde ficha y se pone a trabajar".

Cada acto o fase se representa por medio de un rectángulo, mientras que la fase decisoria del estado del tiempo se hace con un rombo (también podría ser un hexágono). Todos ellos van unidos por líneas que marcan el orden de la secuencia lógica.

Ahora bien, frente a un mismo problema podemos llegar a diferentes conclusiones a la hora de presentar una solución. Si todas son buenas, ¿cuál escogeremos? Siempre resulta más válida aquella en la que no se vuelven a escribir unas mismas partes del programa, denominadas *rutinas*, comunes a varios caminos fundamentales.

Así, la figura 2 ilustra cómo se puede evitar la repetición del área en que se representan los elementos "llegar a la oficina", "fichar" y "trabajar".



El Basic ZX de Sinclair

El BASIC se ha convertido en el lenguaje oficial de los microordenadores, pero casi todos poseen su propia versión. Una de las más utilizadas es el Basic ZX de Sinclair

Empecemos por los nombres de las variables, que siempre constituyen una fuente de confusión entre los dialectos del BASIC. En el Sinclair, los nombres de las variables alfanuméricas deben tener una sola letra y no hay distinción entre letras en mayúscula y letras en minúscula. Esto significa que las variables a$ y A$ se refieren a la misma posición de memoria. Los nombres de las matrices de cadenas siguen las mismas reglas que las variables simples, y las inicializan, de modo que después de haber DIMensionado la matriz de cadenas H$, todas las ulteriores menciones a H$ en el programa se tomarán como referidas a la matriz H$. Esto se deriva del hecho de que el BASIC de Sinclair considera todas las variables alfanuméricas como variables de tipo matriz, algunas de ellas DIMensionadas formalmente y otras no.

Los nombres de las variables numéricas están menos restringidos que los de las variables alfanuméricas: deben empezar con una letra y deben estar compuestos de letras o dígitos, pero pueden ser de cualquier longitud. Pueden incluir espacios y pueden ser una combinación de letras en mayúscula y en minúscula, pero si bien estos factores son de gran ayuda para el programador, no tienen ninguna significación para la máquina, que hará caso omiso de ellos. Algunos nombres de variables numéricas posibles serían:

```
qwert, ub40, curso de informática
```

y los siguientes son sus equivalentes exactos:

```
QWERT, UB 40, Curso de Informática
```

Los nombres de las matrices numéricas deben ser letras únicas, sin que sea óbice para utilizarlas en simples variables numéricas: la variable matriz v(8) es muy diferente de la variable numérica simple v. Las variables numéricas de letra única que no son matrices, como v, son las únicas que admiten los contadores de los bucles FOR...NEXT, de modo que FOR V = 1 to 9...NEXT V es correcto, pero FOR bucle = 1 TO 9 no lo es.

Las principales diferencias entre el lenguaje Sinclair y otras versiones de BASIC radican en el tratamiento de las cantidades de las cadenas. Vamos a comenzar por la sentencia DIM. En el BASIC de Sinclair, cuando se ejecuta la sentencia DIM a$(12), se reservan 12 bytes de memoria exclusivamente para el empleo de la variable a$, y estos bytes se inicializan con espacios. A cada uno de estos bytes se puede aludir como variables subíndice, o se pueden reclamar los 12 bytes colectivamente como a$. La longitud de esta variable siempre será 12, y las asignaciones a la misma se rellenarán con espacios o se truncarán por la derecha cuanto sea necesario para respetar su longitud. Supongamos que escribimos:

```
DIM a$(12): LET a$ = "123456789"
```

Entonces a$ realmente contendrá estos nueve caracteres "123456789" seguidos de tres espacios, dando un total de 12 caracteres. Si, en cambio, escribimos:

```
DIM a$(12): LET a$ = "ABCDEFGHIJKLMN"
```

Entonces a$ en realidad contendrá sólo los primeros 12 caracteres "ABCDEFGHIJKL": la cantidad de la cadena "ABCDEFGHIJKLMN" se ha truncado por la derecha para que encaje en la longitud DIMensionada de a$. Si ahora escribimos:

```
LET a$(2 TO 5) = "1234"
```

entonces a$ contendrá "A1234FGHIJKL". Esto ilustra la eficacia de Sinclair en la manipulación de cadenas: todas las cadenas son tratadas como matrices de cadenas de dimensión simple, las matrices pueden llevar o no subíndice, y se puede acceder a los elementos individuales de una matriz (de forma individual o como parte de una subcadena) mediante subíndices. Asimismo, ilustra otra gran diferencia respecto a otras versiones de BASIC. En otros sitios, DIM a$(12) crea 12 variables alfanuméricas separadas denominadas a$(1), a$(2), etc., cada una de las cuales tiene la longitud de la expresión a ella asignada. Si a una variable alfanumérica determinada no se le hubiera asignado nada, entonces su longitud sería 0 y contendría sólo la cadena nula, "".

En otras versiones de BASIC esta forma de manipular las cadenas requiere de las diversas funciones para cadenas, LEFT$, RIGHT$, MID$ y algunas veces INSTR, para posibilitar la manipulación de subcadenas y la partición de cadenas de la forma dicha. Pero esto no es así en el BASIC de Sinclair. Los equivalentes Sinclair de estas funciones para cadenas son:

```
LEFT$(A$,N) = A$(TO N)
```

(que significa los N caracteres más a la izquierda de A$):

```
RIGHT$(A$,N) = A$(LEN A$−N+ TO)
```

(que significa los N caracteres más a la derecha de A$); y

```
MID$(A$,P,N) = A$(P TO P+N−1)
```

(que significa los N caracteres de A$ desde la posición P hacia adelante). Por su parte, la función:

```
LET S = INSTR(A$,"teststring")
```

(que significa hallar la posición de comienzo en A$ de la subcadena cuyo valor sea, por ejemplo, "teststring") se puede reemplazar por:

```
LET Y$ = A$:LET Z$ = "teststring":GOSUB
  9900:LET S = POSN
```

```
9900 LET    ZL = LEN    Z$:LET    SL = LEN
  Y$-ZL+1:LET POSN = 0
9910 FOR K = 1 TO SL
9920 IF  Y$(K  TO  K+ZL-1) = Z$  THEN  LET
  POSN = K:LET K = SL
9930 NEXT K:RETURN
```

Observe cómo la variable alfanumérica Y$ es tratada como una variable subíndice de tipo matriz, aun cuando no se la haya DIMensionado. Dado que en BASIC de Sinclair todas las variables alfanuméricas son variables de tipo matriz, una variable alfanumérica que no esté DIMensionada es implícitamente una matriz de caracteres únicos, de dimensión única y longitud variable; de estar DIMensionada, la longitud de su elemento está determinada por el último número de la sentencia DIM. Mientras que en otras versiones de BASIC DIM y$(8,7) crea una matriz de dos dimensiones, en Sinclair crea una matriz de dimensión única de ocho elementos, cada uno de los cuales tiene una longitud fija de siete caracteres.

La estricta atención que el BASIC de Sinclair presta a la longitud de las variables alfanuméricas DIMensionadas significa que sentencias aparentemente sencillas pueden producir efectos diferentes, según se haya ejecutado o no una sentencia DIM. Si a$ es una variable alfanumérica simple, entonces LET a$ = "" hace que el contenido de a$ sea igual a la cadena nula ("") y la longitud de a$ sea igual a cero. No obstante, si previamente se hubiera ejecutado DIM a$(7), entonces LET a$ = "" hace que el contenido de a$ sea igual a siete espacios, y la longitud de a$, igual a siete (que tendrá siempre, a partir de la sentencia DIM). Además, en dicho caso, aun cuando se hubiera ejecutado LET a$ = "", una condición como:

```
IF a$ = "" THEN PRINT "cadena nula"
```

fracasará, y no se imprimirá nada: a$ es igual a siete espacios y no a la cadena nula.

Si necesita verificar de esta manera los elementos de la matriz de cadenas, entonces probablemente sea mejor reservar con este fin una variable alfanumérica, DIMensionarla a la longitud de la variable de matriz más larga utilizada en el programa, y verificar sus variables de matriz mediante ella, así:

```
100 DIM a$(12,34)
120 DIM b$(7,56)
140 DIM N$(56)
150 REM N$ se utilizará como la cadena vacía
.....
580 IF b$(3) = N$(TO 56) THEN PRINT "vacía"
590 IF a$(11) = N$(TO 34) THEN PRINT "vacía"
.....
```

Aquí N$ sólo se emplea como la cadena vacía y, si no se utilizara de esta manera, entonces las condiciones de las líneas 580-590 habrían de echar mano de literales, de modo que:

```
580 IF b$(3) = "                    " THEN PRINT "vacía"
585 REM 56 espacios entre las comillas
```

Esto es incómodo y es susceptible de error. Una alternativa a la utilización de N$ de esta forma consiste en DIMensionar todas las variables de matriz con un elemento más del que se necesita y utilizar ese último elemento como una cadena vacía para las condiciones de dicha matriz, de modo que la línea 590 podría decir:

Ian McKinnell

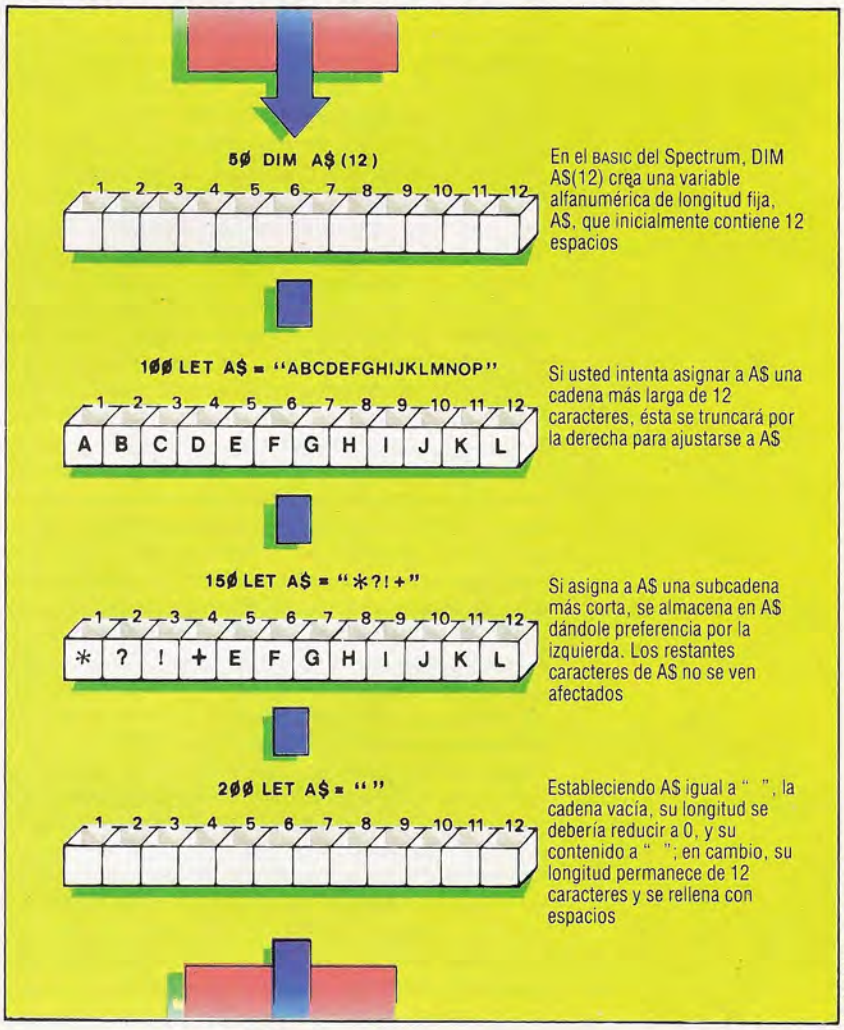
SuperBASIC
El SuperBASIC de Sinclair posee una gama de órdenes considerablemente mejorada respecto al BASIC ZX, pero la novedad más llamativa es el abandono del sistema de entrada de palabras clave tecla a tecla, común al ZX80, al ZX81 y al Spectrum. Originalmente se pensó como una medida de economía para los usuarios (se creyó que pulsar una sola tecla en lugar de digitar una palabra entera resultaría estimulante). El sistema fue provisto de una variedad de "modalidades" diferentes para permitir que la entrada de caracteres únicos se diferenciara de la entrada de palabras clave. Este sistema fue atractivo para aquellos usuarios de Sinclair que nunca antes se habían enfrentado a un teclado, pero resultó frustrante para quienes ya habían utilizado una máquina de escribir

```
590 IF a$(11) = a$(12) THEN PRINT "vacía"
```

dando por sentado que a$(12) no se emplea nunca y que, por consiguiente, sólo contiene espacios.

Por último, observe que en el Sinclair el primer elemento de cualquier matriz posee el subíndice uno, mientras que en algunas otras versiones de BASIC el primer elemento de una matriz tiene el subíndice cero. En el próximo capítulo del curso terminaremos este análisis del BASIC Spectrum.

Cadenas de Procusto
El personaje mitológico griego Procusto era un mesonero que tenía camas de un solo tamaño y estiraba o acortaba a sus huéspedes para que se adaptaran a ellas



Kevin Jones

Primeros conceptos

Este primer capítulo del curso de programación en lenguaje máquina pretende hacer perfectamente comprensibles los fundamentos de la programación de ordenadores

La programación en lenguaje máquina es la llave que deja al descubierto la verdadera potencia del microprocesador. De este modo, el programador tiene en sus manos el control directo de la máquina.

El lenguaje máquina tiene frases como ésta:

```
INSTK: SBC $D9FA, X; Outport valor bandera
```

o ésta:

```
DE23 FD FA D9
```

o ésta:

```
11011110 00100011 11111101 11111010 11011001
```

Algunas veces se escribe así:

```
1240 LET ACC = ACC-FLAG (X)
```

Y otras veces de esta manera:

```
PERFORM FLAG-ADJUST THROUGH LOOP 1
```

Son frases de un mismo código y, puesto que está destinado a una máquina informática, se denomina *lenguaje máquina*. A la máquina, en realidad, no le dice más que se trata de un patrón de niveles de voltaje o una corriente de electricidad.

Normalmente, cuando hablamos de lenguaje máquina nos estamos refiriendo al lenguaje ensamblador y el primer ejemplo que hemos ofrecido en este artículo es una instrucción en lenguaje ensamblador 6502. El que hayamos añadido otros ejemplos más era para demostrar que no existe un lenguaje máquina específico como tal, sino un buen puñado de diferentes maneras de representar una secuencia de acontecimientos electrónicos, y de representarla de modo que nos resulte más o menos fácil de comprender. De modo que lo primero que hay que aprender acerca del lenguaje máquina (o del lenguaje ensamblador, pues de momento no nos va a preocupar su diferencia) es tan sólo que se trata de un lenguaje de programación como otro cualquiera. Pero la programación siempre viene antes que el lenguaje: tanto si usted escribe sus programas en ensamblador IBM, en BASIC Atari o en cualquier otro lenguaje, antes de sentarse al teclado debe haber resuelto el problema de la programación en su cerebro. El lenguaje en el cual usted exprese luego su solución, obviamente incidirá en la forma del programa final. En realidad, se puede escoger entre varios posibles lenguajes porque buscan que la codificación de su programa resulte más fácil, más corta o más legible. Pero la solución es lo primero: el contenido es antes que la forma.

En este caso, ¿por qué llamarlo lenguaje máquina y por qué molestarnos en utilizarlo? Lo llamamos con ese nombre porque el conjunto de sus instrucciones se corresponde exactamente con el de las operaciones "primitivas" o esenciales que puede realizar un microprocesador determinado. Empleamos el código de lenguaje máquina cuando importa dirigir la operación del microprocesador paso a paso, en vez de que un intérprete de lenguaje de programa lo controle de forma más general.

La razón más común por la que se utiliza es la velocidad: si su programa direcciona al procesador de forma más o menos directa, entonces evita el relativamente largo camino de la traducción del programa. En otras palabras, eliminando el intermediario se gana tiempo. Es decir, tiempo de ejecución del programa. La detallada codificación, verificación, depuración, modificación y mantenimiento de un programa en lenguaje máquina es probable que nos pida al menos el doble de tiempo que esas mismas operaciones necesitan en un lenguaje de alto nivel. El carácter oscuro y casi ininteligible del lenguaje máquina estimuló el desarrollo de lenguajes como el COBOL y el BASIC.

Si el conjunto de instrucciones en lenguaje máquina es el de las operaciones del procesador, ¿entonces qué son estas operaciones y qué es lo que hace el procesador? En términos lo más sencillos posible, la unidad central de proceso (CPU: *Central Processing Unit*) de un ordenador es un interruptor que controla el flujo de corriente de un sistema informático a través de los componentes de dicho sistema. Estos componentes son la memoria, la unidad aritmético lógica (ALU: *Arithmetic Logic Unit*) y los dispositivos de entrada-salida. Cuando usted pulsa una tecla está dando entrada a alguna información; en la máquina, sin embargo, simplemente está generando un patrón de voltajes en la unidad de teclado. La CPU conmuta el patrón desde el teclado a parte de la memoria, luego conmuta un patrón correspondiente desde algún lugar de la memoria hasta la pantalla de modo que en ella aparezca un patrón de caracteres. Puede que este proceso le resulte parecido al de una máquina de escribir; pero en ésta hay una conexión mecánica entre pulsar una tecla e imprimir un carácter, mientras que en un ordenador esa conexión sólo existe en virtud de que la CPU conmuta los patrones de voltaje correctos de un lugar a otro. Algunas veces pulsar una tecla no produce en la pantalla un carácter: la pulsación de una tecla puede destruir un asteroide, o guardar un programa, o eliminar un archivo en disco, además de imprimir una letra. La operación depende de cómo y dónde conmute la CPU la corriente eléctrica.

En este análisis esquemático, la CPU está situada en el corazón del sistema y toda la información (o corriente eléctrica) debe pasar a través de ella desde un componente a otro. En realidad, la CPU y el sistema son algo más complicados, pero no es un análisis engañoso. Puede imaginar la CPU como un controlador maestro que establece interruptores menores a través de todo el sistema para controlar

el flujo de electricidad y que, por consiguiente, controla indirectamente el flujo de información, en vez de imaginar que canaliza toda la información físicamente a través de sí misma.

Los efectos de las operaciones de conmutación de la CPU se pueden clasificar, para nuestros fines, como: operaciones aritméticas y operaciones lógicas, operaciones de memoria y operaciones de control. Todas estas operaciones son resultado de conmutar la información a través de distintos caminos del sistema y de la CPU, y para la CPU es como si todas ellas se trataran de la misma cosa.

Las operaciones aritméticas son, en realidad, la configuración más importante de la máquina. La CPU puede sumar dos números entre sí o restarle uno al otro. La resta se consigue representando uno de los números como un número negativo y sumándole ese número negativo al otro número; 7 + 5 = 12 en realidad significa:

(más 7) sumado a (más 5) es igual a (más 12).

7 − 5 = 2 en realidad significa:

(más 7) sumado a (menos 5) es igual a (más 2).

La multiplicación y la división se consideran como sumas o restas repetitivas, de modo que se puede programar a la CPU para que también imite estos procesos. Si la CPU puede hacer frente a las cuatro operaciones aritméticas, entonces puede hacer frente a cualquier proceso matemático. Merece la pena recalcar, no obstante, que todo su potencial matemático descansa en la simple capacidad de sumar dos números.

Para los fines que perseguimos actualmente, las operaciones lógicas se pueden describir como la capacidad de comparar dos números: no sólo en cuanto al tamaño relativo, sino también respecto al patrón de sus dígitos. Es fácil ver que siete es mayor que cinco, porque podemos quitarle cinco a siete y aún tendríamos un resultado positivo. La CPU posee la capacidad de hacer esa clase de comparación, y también puede comparar 189 con 102 y reconocer que ambos números tienen el mismo dígito en la columna de las centenas.

Esencialmente, la CPU puede realizar dos operaciones de memoria: puede copiar información de una posición de memoria en su propia memoria interna, y puede copiar información de su memoria interna a otra posición de memoria. Haciendo estas dos cosas, una después de la otra, puede, por consiguiente, copiar información de cualquier parte de la memoria a cualquier otra parte de ésta. Para que la memoria nos sea útil, la CPU debe ser capaz de hacer estas dos cosas, y estas dos operaciones son todo cuanto necesita para conseguir una administración completa de la memoria.

Las operaciones de control son, en realidad, decisiones acerca de la secuencia en la cual la CPU realiza las otras operaciones que hemos descrito aquí. Por el momento no es importante entenderlas con mayor profundidad: si uno acepta que la CPU puede tomar decisiones relativas a su propia operación, eso ya es suficiente por ahora.

De modo que la CPU puede realizar operaciones aritméticas, comparar números, desplazar la información por la memoria y decidir su propia secuencia de operaciones. Ésta es una lista sencilla de procedimientos y, así y todo, ¡describe y especifica completamente una máquina de informática ideal! Si la CPU puede hacer estas cuatro cosas, haciéndolas en la secuencia correcta puede realizar cualquier trabajo informatizable. La secuencia correcta, por supuesto, es el programa de ordenador para el trabajo determinado y aquí es donde entramos nosotros como programadores. Si la CPU pudiera generar sus propias secuencias de operaciones, no se nos necesitaría a nosotros para nada.

Puede que usted no esté aún muy convencido de que los cuatro tipos de operaciones que hemos descrito sean la idea cabal de un ordenador, así que vamos a tomar un programa en BASIC y reducirlo a las operaciones generales efectuadas. ¿Qué son estas operaciones fundamentales? En cualquier programa se encuentra con variables, que son simplemente los nombres de los lugares de la memoria en donde está almacenada la información. La mayoría de los programas realizan alguna clase de operación aritmética con algunas de estas variables. Una vez efectuada la operación, a menudo se compararán dos informaciones y, como resultado, se ejecutará un conjunto de instrucciones u otro. Finalmente la información suele entrar en el programa del usuario por el teclado, y salir hacia el usuario a través de la pantalla.

Salvo la sentencia relativa a entrada y salida, esta descripción no contiene más que las cuatro operaciones elementales de la CPU expresadas con diferentes palabras. Y, si se acepta por el momento que para la CPU todos los dispositivos de entrada-salida son sólo zonas especiales de la memoria, entonces la imagen del ordenador ideal ejecutando programas reales es completa. Por consiguiente, la ejecución de un programa se puede describir como un flujo de información dirigido hacia adentro, alrededor y hacia afuera del ordenador; usted proporciona alguna información mediante el teclado, esa información la manipula su programa, y en la pantalla aparece otra información.

Si el ordenador idealizado no es más que una CPU y algo de memoria, entonces antes de seguir adelante debemos analizar la memoria del ordenador: ¿qué es y cómo funciona?

Imagínese un circuito eléctrico sencillo que conste de una pila, un interruptor y una bombilla: si se acciona el interruptor se enciende la luz, y permanece encendida hasta que se agota la pila o hasta que se cierra el interruptor. Entonces el estado de la bombilla (ON u OFF: encendida o apagada) es una información, y todo el circuito es un dispositivo de memoria que registra esa información. Supongamos ahora que el interruptor está localizado a la entrada de una fábrica y que la luz está situada en el despacho del director. Cuando el primer empleado llega a la fábrica, acciona el interruptor de la entrada y el director, al ver la luz encendida en su despacho, sabe que alguien ha llegado al trabajo. El director no necesita estar en el despacho cuando la luz se enciende: puede mirar la bombilla en cualquier momento para averiguar si alguien ha llegado. La información de que alguien se ha presentado a trabajar está almacenada en el circuito.

Así es casi exactamente como la información se almacena en la memoria del ordenador: toda información se reduce a la presencia o a la ausencia de electricidad en un circuito. Naturalmente, no se trata tan sólo de esto, de manera que vamos a mejorar el sistema de información de la dirección de la fábrica. Supongamos que tenemos cuatro cir-

cuitos interruptor-bombilla separados (los cuatro interruptores en una fila junto a la puerta y las cuatro bombillas en una fila correspondiente en el despacho), de tal forma que cerrando el interruptor situado más a la izquierda se ilumine la bombilla localizada más a la izquierda, y así sucesivamente. Ahora imaginemos que a cada empleado se le indica que conecte los interruptores de una manera distinta a la de su compañero, de modo que cuando llegue Margarita conectará el primero y el segundo interruptor y dejará abiertos el tercero y el cuarto; que Gerardo cierre el cuarto interruptor y abra los otros tres; que Pablo cierre el primero y el tercero y abra el segundo y el cuarto; y así para los 16 empleados. Las bombillas informarán ahora al director cuál es el empleado que ha llegado al trabajo.

Supongamos que la posición OFF de cada interruptor está etiquetada 0 y que la posición ON está etiquetada 1: más claramente diremos que Margarita tiene que establecer el panel de interruptores en 1100 (los dos primeros interruptores ON, el tercero y el cuarto OFF); a Gerardo le corresponde el patrón 0001 (el cuarto interruptor ON, los otros tres OFF) y Pablo ha de establecerlos en 1010 (el primero y el tercero ON, los otros dos OFF). Si el director entiende el 1 como bombilla encendida y el 0 como apagada, los empleados y él estarán hablando el mismo lenguaje de identificación. Para todos ellos, "0001" significa Gerardo.

¿De cuántas combinaciones diferentes de interruptores disponemos? Cada interruptor puede estar en una de dos posiciones y hay cuatro interruptores, de modo que hay $2 \times 2 \times 2 \times 2 = 16$ posibilidades diferentes, que serían:

0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111,
1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

Observe con qué rapidez hemos pasado de la imagen concreta de las bombillas en una habitación a la región abstracta de patrones de unos y ceros. Sólo con un poco más de abstracción podremos convertir estos patrones en números.

Trate de imaginar y escribir los números a medida que los va contando. Usted puede escribir de cero a nueve muy fácilmente, porque cada uno de estos números tiene un nombre y un símbolo distintos para representarlo. Pero ¿qué es lo que escribe después de nueve? Dispone de un nombre (diez) para ese número, pero no tiene un rasgo nuevo que lo represente. Inconscientemente echa mano de dos símbolos que usted ya utilizó por separado, el 1 y el 0, y así ha compuesto 10, un símbolo arbitrario para el diez. Lo mismo sucede con el once, pues en realidad usted escribe dos unos, el trece, catorce, etc., hasta agotar con el noventa y nueve, 99, las combinaciones por parejas de los únicos diez símbolos de que dispone (excluyendo las que tendrían el cero delante). Para representar cien, no tiene más remedio que pasar de las parejas de símbolos a los tríos: 100 (uno-cero-cero), 110 (uno-diez), etc. Esto parece trivial, pero quizá recuerde todavía lo difícil que le resultó aprenderlo en la escuela. Sin usar aquellas hojas cuadriculadas que le separaban las centenas de las decenas y de las unidades, ahora instintivamente interpreta el número 152 compuesto por una centena, cinco decenas y dos unidades, o lo que es lo mismo: $1 \times 100 + 5 \times 10 + 2 =$ ciento y cincuenta y dos.

¿Cómo escribiría usted los números si sólo le permitieran dos símbolos, el 0 y el 1? Podemos escribir fácilmente el 1, ¿pero cómo podemos representar el número siguiente? Nos hemos quedado sin dígitos exclusivos, de modo que debemos echar mano de los que nos permiten (como hicimos al disponer de diez dígitos) y escribir el siguiente número, que es el dos, como 10. Desde ahora acordamos que el número llamado "dos", en este sistema de tan sólo dos símbolos (binario) se escribirá 10. Seguimos contando y el número siguiente es tres, que lo debemos escribir como 11. ¿Y después, qué? Nos hemos quedado sin combinaciones de dos dígitos, de modo que el número siguiente, cuatro, se debe representar como 100; cinco debe ser 101 (cuatro y uno), seis es 110 (cuatro y dos) y siete es 111 (cuatro y dos y uno). En pp. 54 y 55 tenemos la expresión gráfica de lo que estamos tratando.

Así como un número decimal como 152 significa: $(1\times100) + (5\times10) + (2\times1)$, el número binario 101 significa: $(1\times4) + (0\times2) + (1\times1)$. En vez de emplear columnas de centenas, decenas y unidades para nuestros números, debemos utilizar columnas marcadas con cuatros, doses y unidades. En un número decimal el valor de un dígito se multiplica por diez para cada columna que se desplaza hacia la izquierda; en un número binario, el valor de un dígito se multiplica por dos para cada columna que se desplaza hacia la izquierda.

De modo que el sistema binario es sólo una forma diferente de representar los números, tan arbitraria como la decimal. Si conoce la numeración romana no le resulta difícil entender si escribimos que en *Blancanieves* hay VII enanitos; ¿entiende que se puede escribir 111 enanitos? El número verdadero de enanos no se altera en virtud de la forma en que lo representemos, pero es aconsejable leerlo así "uno uno uno binario" y escribirlo como "111 b", de modo que no se confunda con una representación decimal.

Ahora podemos volver a nuestra analogía original de cómo los empleados de la fábrica conmutan patrones, y decidirnos por un método que haga que éstos resulten más fáciles de utilizar. Lo más sensato que se puede hacer consiste en tratar a estos patrones como números binarios de cuatro dígitos. Esto significa que la señal de Margarita es 1100 binario, que en decimal es 12. La señal de Gerardo es 0001 binario (1 decimal), y la señal de Pablo es 1010 binario (10 decimal). Cuando el director observa el patrón de luces en su despacho, él lo puede leer como un número binario, convertirlo a su equivalente decimal y consultar la lista de empleados para ver a quién le corresponde ese número. De modo que podemos decir que la información se almacena en la corriente eléctrica, y que son los interruptores los que le confieren significado.

Nuestra analogía nos ha proporcionado una imagen sencilla de cómo se representa la información en un ordenador: para el ordenador sólo se trata de patrones de voltaje (es decir, las luces están ON u OFF); pero a nosotros, seres humanos, nos resulta más fácil considerar estos patrones como números binarios. Todo se reduce a una cuestión de representación. Si ahora piensa que "1010" es el código que significa "Pablo", entonces podrá empezar a comprender la relación que guarda todo con el código de lenguaje máquina. En el próximo capítulo veremos cómo se usan los números binarios para representar información en un ordenador personal.

Acelerando
Estos tres breves programas, uno para el ZX Spectrum, otro para el BBC Micro y el tercero para el Commodore 64, demuestran la diferencia en cuanto a velocidad de operación entre el BASIC y el código de lenguaje máquina para visualizar ya sea el juego de caracteres completo (Commodore y BBC) o bloques de color (Spectrum) en la pantalla

Spectrum

```
1 REM*****************************
10 REM*** CODIGO L/M SPECTRUM ***
11 REM* NO LISTAR LINEA 1        *
12 REM* DESPUES DE EJECUTAR PROG*
13 REM*****************************
99 REM*****************************
150 LET PTR=23635: LET SA=PEEK
(PTR)+(256*PEEK (PTR+1))+7
200 BORDER 2
350 DATA 1,0,3,17,0,88,33,0,0,
237,176,201
400 FOR X=0 TO 11
500 READ MC
600 POKE SA+X,MC
700 NEXT X
1000 LET OFFSET=0
1100 FOR X=0 TO 1 STEP 0
1200 POKE SA+7,OFFSET
1300 LET FALSO=USR SA
1400 LET OFFSET=OFFSET+13
1500 IF OFFSET>=256 THEN LET
OFFSET=OFFSET-256*INT(OFFSET/256)
1600 NEXT X
1700 STOP
1799 REM*********************************
1800 REM*GUARDAR PROG ANTES DE EJECUT*
1801 REM*GUARDAR PROG ANTES DE EJECUT*
1802 REM*********************************
```

BBC Micro

```
100 REM*************************************
*********************BBC*********************
*************************************       *
149 REM*************************************
150 REM*          CODIGO L/M BBC            *
151 REM*************************************
200 MODE 4:*TV 254
300 GOSUB 30000
700 FOR P=1920 TO 6079
800 K=K+1:IF K>2679 THEN K=1920
900 ? (HIMEM+P)=(K+47232)
1000 NEXT P
1100 PRINT TAB(13);"ESO ERA BASIC"
1200 INPUT" PULSE RETURN PARA VERSION
CODIGO LENGUAJE MAQUINA ",A$:CLS
1300 FOR L=0 TO 15:FOR B=0 TO 255 STEP L
1400 ? (SA)=LS:? (SA+1)=HS
1500 ? (FA)=LF:? (FA+1)=HF
1600 FALSO=USR(PSTRT)
1700 VDU 30
1800 NEXT B,LP
1900 END
30000 REM*******S/R CARGADORA LM*******
30010 K=1919:PSTRT=PAGE+8:VSTR=HIMEM+1920
30020 HS=INT(VSTR/256):LS=VSTR-256*HS:LF
=LS+56:HF=HS+2:SA=114:FA=116
30100 DATA 50,169,32,197,112,48,4,240,2,
133,112,165,112,32,227,255
30110 DATA 230,114,208,2,230,115,165,116,
197,114,208,7,165,117
30120 DATA 197,115,208,1,96,230,112,169,
128,197,112,208,224
30130 DATA 169,32,133,112,208,218,96,96,96
30150 READ ZZ
30160 FOR BY=PSTRT TO PSTRT+ZZ
30170 READ MC:? (BY)=MC
30180 NEXT BY
30200 RETURN
30299 REM*************************************
30300 REM*    ¡¡¡GUARDAR ANTES DE EJECUTAR!!! *
30301 REM*************************************
30399 REM*************************************
30400 REM*    ¡¡NO LISTAR LINEA 100 DESPUES   *
30401 REM*          DE EJECUTAR PROGRAMA!!    *
30402 REM*************************************
```

Commodore 64

```
99 REM ************************
100 REM*CODIGO L/M COMMODORE *
101 REM************************
200 PRINT CHR$(147) :REM LIMPIAR PANTALLA
300 PRINT "      ESTO NO LLEVARA MUCHO"
400 GOSUB 60000
500 PRINT CHR$(147);CHR$(5) :REM CLS Y BLANCO
600 CC=0
700 FOR P=SM TO FM
800 POKE P,CC:POKE P+OF,CL
900 CC=CC+1: IF CC>255 THEN CC=0
1000 NEXT P
1100 PRINT TAB(13);"ESO ERA BASIC"
1200 INPUT" PULSE RETURN PARA VERSION
CODIGO LENGUAJE MAQUINA ";A$
1300 FOR LP=1 TO 9:FOR B=0 TO 255 STEP LP
1400 POKE SA,LS:POKE SA+1,HS
1500 POKE FA,LF:POKE FA+1,HF
1600 POKE BA,B:POKE CH,0
1700 SYS AA
1800 NEXT B,LP
1900 END
60000 REM******S/R CARGADORA LM******
60010 SM=256*PEEK(648):OF=55296-SM:FM=SM
+999:BD=53280:SC=BD+1:CS=8:CB=6:CL=0
60020 POKE BD,CB:POKE SC,CS
60030 LS=0:HS=PEEK(648):LF=232:HF=HS+3:SA
=251:FA=253:BA=250:CH=2
60100 DATA 850,885,169,0,170,165,250,133,
2,165,2,129,251
60110 DATA 230,251,208,2,230,252,165,253,
197,251,208,7,165,254
60120 DATA 197,252,208,1,96,230,2,208,229,
240,223
60150 READ AA,ZZ
60160 FOR BY=AA TO ZZ
60170 READ MC:POKE BY,MC
60180 NEXT BY
60200 RETURN
60299 REM************************************
60300 REM* GUARDARLO ANTES DE EJECUTARLO *
60301 REM************************************
```

Bill Gates: el impulsor del estándar

En menos de una década, Microsoft se ha convertido en el proveedor de software para microordenadores más influyente del mundo

Cortesía de Microscope

Tony Sleep

La empresa Microsoft, que ahora representa un negocio multimillonario en dólares, es la clásica historia de unos entusiastas a quienes les ha ido bien. Bill Gates, que a los 28 años preside la junta directiva, en 1972 sólo era un aficionado con talento.

En la escuela superior de Seattle (Estados Unidos), donde la asociación de padres y educadores tuvo la buena idea de equipar a los estudiantes con un terminal de tiempo compartido acoplado al popular miniordenador DEC PDP-11, Bill aprendió el funcionamiento de los ordenadores. Fue a la Universidad de Harvard y, después de graduarse, se inició en los negocios en Bellevue en compañía de su condiscípulo y amigo Paul Allen. La firma que crearon se llamó *Traff-O-Data*, y su trabajo consistía en supervisar el flujo del tráfico, contratados por las autoridades públicas de Seattle. Era un momento trascendental en el desarrollo del microordenador: los primeros microprocesadores estaban haciendo su aparición y quienes tenían imaginación y entusiasmo veían un gran futuro para dispositivos como el 4004 de Intel y, posteriormente, el chip 8080. Ya entonces Bill estaba muy familiarizado con el DEC PDP-11 y uno de sus primeros trabajos fue rastrear errores en este ordenador. Pensó que sería una buena idea adaptar su BASIC para utilizarlo con el 8080. No tenía ningún sistema de desarrollo y la primera vez que acopló el código y la máquina fue cuando Bill llevó las cintas a Altair, en Albuquerque (Nuevo México). Funcionó a la primera. Y así nació el MBASIC, que desde entonces ha sido el estándar a superar.

Microsoft fue adquiriendo fama de casa de software con experiencia en dotar de sistemas operativos a nuevos ordenadores, rellenar la caja vacía, por decirlo así, y la IBM se puso en contacto con Gates para solicitar su consejo acerca de cómo especificar y equipar un ordenador personal para un único usuario. Inicialmente, Gates sugirió que Gary Kildall, de la Digital Research, el hombre encumbrado por el éxito del CP/M, era la persona adecuada para ese trabajo. Pero la IBM volvió a dirigirse a la Microsoft. Ésta reescribió el PASCAL, FORTRAN y MBASIC para su adaptación a los 16 bits y también creó el BASIC GW, con sus capacidades ampliadas para música y gráficos.

Al mismo tiempo, Gates comprendió que un sistema operativo para múltiples usuarios, poco cuidado pero poderoso, creado por los Laboratorios Bell, se podía adaptar muy bien a los micros más potentes basados en los nuevos microprocesadores de 16-32 bits, y transformó el Unix en el Xenix. Tanto Tandy como Apple adoptaron el Xenix en 1983 para sus propios modelos de 16-32 bits. Incluso Microsoft colaboró en gran medida en la creación más reciente de Apple: el Mackintosh.

Microsoft también tiene una firma que se está introduciendo en el mercado para aficionados. En 1981 se creó ASCII-Microsoft, con un japonés joven y perspicaz, Kay Nishu, para vender su sistema operativo y su BASIC a los fabricantes orientales de la nueva generación de micros, como el NEC PC 8201 y el Tandy Modelo 100. El estándar MSX común surgió a partir del deseo de los fabricantes japoneses de un modelo común, no sólo en lo que concierne a lenguajes, sino también en cuanto a interfaces para aquellos periféricos personales deseables como son plotters e impresoras en color, lápices ópticos, palancas de mando, mandos de bola, brazos-robot, sintonizadores de FM, etc. Parece ser que también tendremos muy pronto un formato de disco estándar de Microsoft que permitirá la transferencia de datos entre los tres medios operativos principales: MSX, MS-DOS y Xenix. Con su interés por un software de fácil uso, como ilustran productos como las ventanas de pantalla y el ratón, Microsoft tiene un brillante futuro.

Directrices

En 1970, a los 28 años, Shiina Takayoshi creó Sord Corporation (40 millones de dólares de ventas en 1982). Redactó 11 principios para que sirvieran de ayuda en el gobierno de su nueva empresa de informática. He aquí 3 de ellos:

"La empresa tiene una deuda, en primer lugar, con la humanidad."

"La empresa debe esforzarse al máximo por determinar qué productos y servicios son los mejores para la sociedad, y darlos a un costo razonable."

"No debe haber solución de continuidad entre el trabajo y la administración. Todas las personas de la empresa se respetarán mutuamente y cooperarán al beneficio de todos."

Estándar industrial

El BASIC (*Beginners' All-purpose Symbolic Instruction Code:* código simbólico de instrucciones para fines generales destinado al principiante) fue desarrollado en 1965 en el Darmouth College (Estados Unidos) por J. Kemeny y T. Kurtz, adelantándose, por tanto, al microprocesador en siete años al menos. Aunque se han creado muchas versiones de este lenguaje, el MBASIC, que es la de Microsoft, se ha reconocido como el estándar para la industria.

La fama de Microsoft se creó a partir del éxito del MBASIC y se consolidó con la producción de un serio competidor del CP/M de la Digital Research, el MS-DOS, un sistema operativo diseñado para aplicarlo a una amplia gama de microordenadores.

Siguiendo el camino iniciado por Xerox con su sistema terminal Star, que después Apple desarrollaría con el Lisa, en la actualidad Microsoft se ha diversificado ligeramente y ha producido un paquete que combina el software con un dispositivo de hardware necesario para su operación: ventanas MS y el ratón. El ratón de Microsoft, al igual que los otros dos de la competencia, utiliza una disposición parecida y un mando de bola acoplado con dos selectores para desplazar el cursor a través de la pantalla

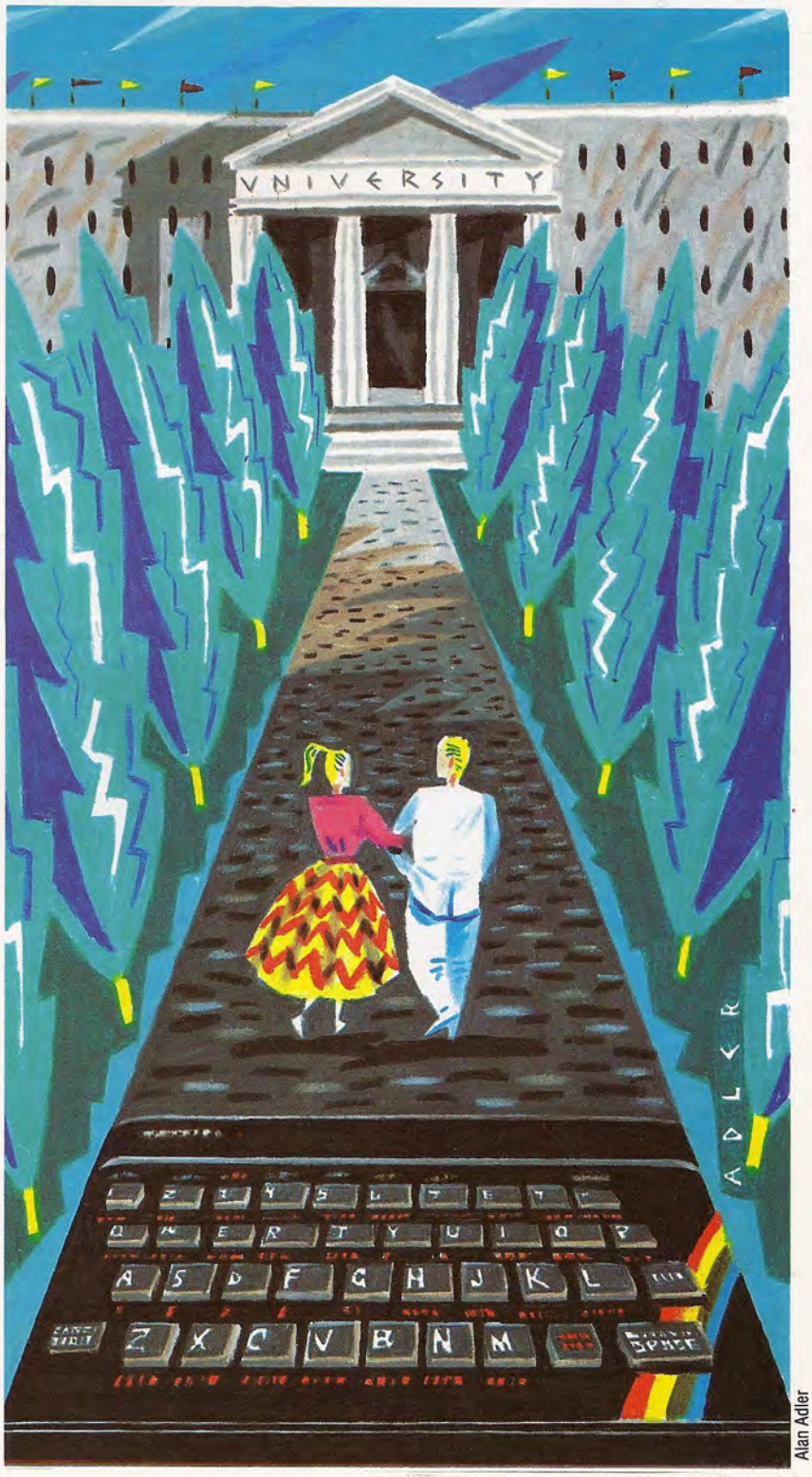
Eficacia educativa

El ordenador puede ser un incomparable auxiliar en la enseñanza, en especial como libro de texto interactivo

En 1980 el gobierno británico lanzó un plan destinado a impartir nociones de informática tanto en establecimientos de enseñanza primaria como secundaria. Conocido como MEP (*Microcomputers in Education Project*: plan para la utilización de microordenadores en la educación), su período de vigencia se estableció en seis años, con un presupuesto global de 21 millones de libras. Quizá sea injusto dividir esta cantidad por 25 000 (el número de escuelas de enseñanza primaria, media y secundaria sólo de Inglaterra) y suponer de este modo una distribución equitativa de los recursos. La máquina que más se recomendó fue el BBC Micro, cuyo precio en el mercado británico es de 400 libras. La segunda opción, el 380Z de Research Machines, tiene un precio bastante más alto que el citado microordenador. Dado que el presupuesto para el proyecto era a todas luces insuficiente, las escuelas se vieron obligadas a recurrir a la autofinanciación.

Hacia 1983, mediado el período de vigencia del plan, el ministro de Tecnología de la Información estuvo en condiciones de proclamar que todas las escuelas secundarias del país (4 553 en Inglaterra) tenían un ordenador, al igual que la mitad de las escuelas primarias. No obstante, la mayoría de los méritos en este sentido se deben atribuir a las asociaciones de padres de alumnos, instituciones benéficas y hasta a los propios escolares, que con gran entusiasmo realizaron considerables esfuerzos para recaudar fondos.

En vez de actualizar el plan de estudios para las escuelas y aumentar su aplicabilidad parecería que la iniciativa del gobierno ha agudizado problemas existentes. La necesidad de equipos costosos ha aumentado la frustración tanto de los maestros como de los alumnos. Ahora existe una conciencia clara de que cada vez se acentúa más la desigualdad de oportunidades entre los niños que pertenecen a ambientes familiares comparativamente más adinerados, quienes disponen de más dinero para contribuir a proyectos como éste, y aquellos niños que pertenecen a familias de menores recursos. Debido al reducido número de microordenadores de que disponen los alumnos de cualquier escuela, es poco probable que el "alumno promedio" tenga acceso a una máquina durante más de 15 minutos por semana, tiempo apenas suficiente para iniciarse en un plan de alfabetización informática y absolutamente insuficiente para explorar las posibles ven-

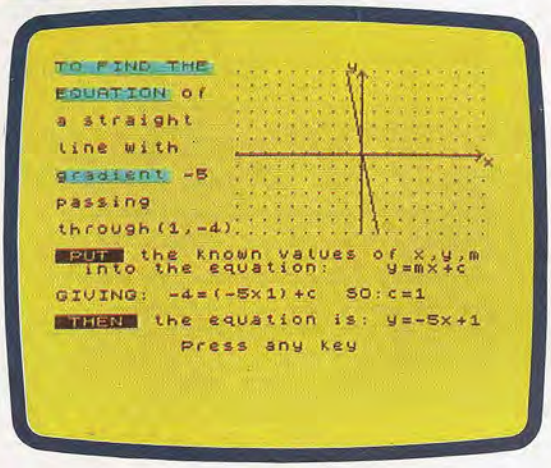


Alan Adler

Una senda segura
La aspiración de un mejor futuro para sus hijos la expresan muchos usuarios de ordenadores personales a través de la compra de software educativo. Para los niños pequeños existe una gran riqueza de material didáctico de iniciación, y para los de mayor edad, que cursan estudios más avanzados, existe un número aún mayor de paquetes de repaso para planes de estudio muy específicos destinados a diversos microordenadores

Paquetes científicos
El campo de acción de los paquetes de software creados para ayudar al estudiante a repasar asignaturas específicas es limitado. Cada uno de ellos intenta cubrir un único aspecto de la materia (matemáticas, o literatura, p. ej.) con el mayor detalle. Nuestro primer ejemplo trata sólo con ecuaciones matemáticas de distintas clases (lineales, de segundo grado y simultáneas) y es para el Spectrum, mientras que el paquete de geometría diseñado para el mismo hardware es de alcance más amplio. Los dos ejemplos restantes plantean problemas relativos a la ley de Ohm y el diseño de amplificadores

"Ecuaciones y desigualdades", Rose Software



tajas de los paquetes de software educativo interactivo.

Dada esta falta de adecuación del sistema escolar, quizá ya no resulte en absoluto sorprendente que un significativo número de usuarios de ordenadores personales incluyan la posibilidad de mejorar las oportunidades educativas de sus hijos como una de las principales razones que los inducen a comprar una máquina.

Pero dejando a un lado el problema del hardware, no cabe duda de que los programas educativos juegan un papel cada vez más importante como medios auxiliares de enseñanza en la clase. Escribir software de repaso, como se ha dado en llamarlo, es un asunto directo, que requiere pocos de los trucos y recursos que se emplean en los programas para juegos basados en pantalla, por ejemplo. Se les considera libros de texto interactivos porque tienen mucho en común con sus equivalentes impresos. El nombre del autor, por ejemplo, puede influir de manera preponderante en las ventas, al igual que ocurre con un libro de texto, y el patrón narrativo establecido en literatura se continúa, asimismo, en el nuevo medio.

La diferencia básica está, no obstante, en la introducción del concepto de interacción. Tradicionalmente, se espera que los estudiantes se basen en dos fuentes de conocimientos: el maestro y el libro de texto. La relación con el maestro es, hasta cierto punto, interactiva, aunque sólo en un pequeño grado, debido a que en cada clase suele haber 30 alumnos o más, y dos horas de instrucción por semana significan que, en el mejor de los casos, cada niño recibe cuatro minutos de atención exclusiva del maestro. No es sorprendente que los educadores hayan explorado las posibilidades de formas más eficaces de que los niños interactúen con el material que están aprendiendo.

Los primeros intentos de cara al aprendizaje mecanizado fueron los laboratorios de idiomas de la década de los sesenta, que se aplicaron a la enseñanza de temas imitativos como son los idiomas extranjeros. Cada estudiante tenía acceso a una grabadora y a un texto pregrabado, y trabajaba con este texto a su propio ritmo. El instructor tenía acceso al canal de audio individual de cada estudiante y podía supervisar o intervenir en la medida de lo necesario. Pero el objetivo consistía en estructurar el curso de instrucción de modo tal que sus intervenciones fueran lo más aisladas posible.

La enseñanza basada en ordenador lleva este proceso a su conclusión lógica y elimina la necesidad de un instructor que intervenga en un momento dado. En muchos sentidos, el paquete de repaso ha de ser un sistema especializado: es decir, tiene que ser totalmente autocontenido, debe evitar las imprecisiones y, además, es necesario que esté diseñado de modo que conduzca al usuario a través del material de una manera fácil y natural. En este caso el sistema informático debería ser lo más sencillo posible de utilizar. Debe partir del supuesto de

"Geometría", Rose Software

que los conocimientos informáticos del usuario son mínimos y, por cierto, para ser más manejable no ha de requerir en absoluto de la pericia propia de un programador o un operador de ordenadores.

Cuando llega el momento de considerar el contenido de software, primero se debe diferenciar entre las ciencias (en las que tratamos con hechos conocidos y cuantificables) y las humanidades (en las que gran parte del análisis es de índole subjetiva y producto, por tanto, del criterio personal). En el caso de los paquetes de repaso basados en ordenador, hay aún otra distinción entre las humanidades y las ciencias. Estas últimas tenderán a estar muy ilustradas (aun cuando debamos depender de microordenadores que posean capacidades para gráficos de resolución relativamente bajas) mientras que las materias de humanidades se verán obligadas a basarse sobre todo en rutinas de manipulación de textos.

Además de los paquetes de repaso y de los paquetes muy especializados en un plan de estudios, en el campo de la educación básica se ha creado gran cantidad de material, en especial las operaciones aritméticas, lectura y ortografía.

Hasta ahora nos hemos venido ciñendo al software de carácter didáctico o bien examinador, ya sea tratando de presentar los hechos de modo que resulten fáciles de memorizar, o bien formulando preguntas tipo test, cuya respuesta correcta está incluida entre otras opciones. Sin embargo, existe un tercer tipo de software, que por lo general el alumno utiliza más en la clase que en su casa: el software de simulación experimental. En este caso las técnicas de programación empleadas son mucho más complejas, ya que se requiere que el programa reproduzca matemáticamente la interacción de fuerzas físicas. La aplicación de métodos de simulación

en la clase está consiguiendo entre los profesores de ciencias tanta popularidad como la que obtuvo con sus equivalentes en la industria, por la sencilla razón de que permite en el ámbito de la clase un nivel de experimentación muchísimo mayor, en especial en aquellas áreas que, en virtud del costo o de los peligros inherentes a la utilización de químicos reactivos, de otra forma resultaría imposible su puesta en práctica.

Consideremos ahora los tres principales grupos por edades para los cuales se producen paquetes educativos, con el fin de comparar las similitudes y las diferencias del software.

Hasta los ocho años

La mayoría de los paquetes de software destinados a los niños más pequeños están relacionados con el desarrollo de la destreza básica en cuanto al reconocimiento de formas y patrones. Muchos utilizan la aritmética simple o la ortografía como los objetos a reconocer o emparejar, reforzando, por consi-

"D.C.", SciCAL Software

guiente, la familiaridad con estos símbolos. Muchos programas que entran en esta categoría se valen de "guiones" del tipo de los de los juegos con la intención de captar la atención del niño con mayor eficacia. Algunos presuponen la presencia de un adulto o de algún niño mayor.

En este grupo de edades gozan de especial aceptación los programas que le enseñan al niño a decir la hora, contar, sumar y restar (un método interesante se sirve de una báscula animada que se inclina hacia uno u otro lado según la carga de cada plato), construir oraciones cortas y deletrear palabras corrientes.

De nueve a catorce años

Para estas edades se abandona un poco el aprendizaje a través del juego y se adopta un enfoque algo más sofisticado, que, en este sentido, reproduce las experiencias del niño en la clase. No es sorprendente que, dado que se espera que el software de este tipo logre que el niño se sienta motivado a utilizarlo, se hayan dedicado considerables esfuerzos en cuanto a incentivos y recompensas. Un método que se ha hecho popular y ha obtenido un éxito razonable le presenta al niño un juego en pantalla después de que él (o ella) ha completado una sección del programa de aprendizaje dentro del tiempo permitido y con un nivel dado de éxito.

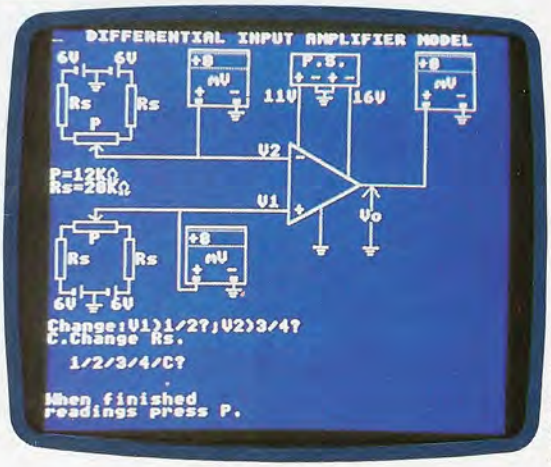
Para los niños entre estas edades los temas más populares siguen siendo la aritmética, la ortografía y el empleo del lenguaje. Pero, además, se encuentran a la venta diversos paquetes de temas históricos o geográficos, algunos que enseñan la teoría básica de la música y cierto número de simulacros sencillos.

A partir de los quince

Es en esta etapa cuando hacen su aparición los paquetes de software educativo de tipo sistema especializado. Existen, por supuesto, paquetes para reforzar las destrezas básicas, pero la mayoría de los fabricantes concentran todos sus esfuerzos en áreas de temas particulares. En este nivel los programas individuales suelen ofrecer un resumen de los conocimientos básicos, proporcionan la teoría y la metodología del tema en cuestión e incluyen una larga relación de preguntas tipo test, con respuestas múltiples, basadas en la asignatura de estudio. En materias como literatura, también ofrecen un análisis de estilo y contenido, de la misma manera que lo haría un profesor.

Se espera que, llegado a esta edad, el alumno haya alcanzado un cierto nivel de automotivación, y se dedica poco esfuerzo, o ninguno, por mantenerlo interesado en el trabajo que tiene entre manos mediante medios artificiales. Esto no quiere

"Ampli.", SciCAL Software



Ian Mckinnell

decir que el tema central se suela presentar de manera convencional. Por el contrario, se propicia universalmente un enfoque inventivo de los gráficos estáticos y animados y el empleo del sonido sintetizado. Para los estudiantes mayores de quince años existe una variedad de paquetes software disponibles, y como muchos de ellos están orientados a un tema muy específico, lo mejor siempre será consultar a un profesional especializado en software educativo para formarse una idea cabal acerca de sus valores relativos.

En un próximo artículo analizaremos con mayor detalle la gama de paquetes de software educativo disponibles en el mercado para los ordenadores personales más populares.

Funciones y estructuras de control

Examinadas las variables y las cadenas en el BASIC del Spectrum, nos falta analizar algunas funciones de este lenguaje, como VAL, WHILE...WEND y REPEAT...UNTIL

Puede que ya haya observado que en el BASIC de Sinclair algunas funciones no requieren encerrar sus argumentos entre paréntesis, a diferencia de sus equivalentes en otras versiones de BASIC, de modo que LEN (X$) se puede escribir ya sea como LEN X$ o LEN(X$). No obstante, es obligado utilizar los paréntesis cuando el significado de una expresión sea dudoso o ambiguo.

La función CODE es el equivalente Sinclair de ASC() y se comporta exactamente de la misma manera. Sin embargo, el juego de caracteres Sinclair es ASCII estándar sólo para los valores entre 32 y 122. De modo que, por ejemplo, si en la mayoría de las versiones PRINT CHR$ (7) da un sonido, el Sinclair envía un mensaje de error.

La función VAL es del BASIC común, pero en Sinclair una sentencia con VAL("a45") rompería el programa, porque el argumento de la función es no numérico. En la mayoría de las otras versiones esto sencillamente daría el valor cero. Si tal arbitrariedad constituyera para usted un problema, escriba una subrutina para sustituir la función VAL, o verifique el valor CODE del primer carácter del argumento de VAL: si CODE A$ (1) < 48 o CODE A$(1) > 57 entonces A$ es no numérica y no puede ser el argumento de la función VAL.

La VAL de Sinclair, no obstante, puede evaluar expresiones numéricas; de modo que:

```
LET A$ = 6*12":PRINT VAL A$
```

daría 72, el valor de la expresión "6*12". En la mayoría de los BASIC la función VAL no es tan poderosa y, en este caso, daría el valor 6.

Esta capacidad de evaluar expresiones se puede utilizar de muchas maneras. El siguiente programa para dibujar un diagrama de barras es un sencillo ejemplo:

```
100 DIM S$(31)
200 LET S$ = "******************************
    **"
300 INPUT "DE ENTRADA A UNA FUNCION DE X";F$
400 PRINT "Y = ";F$:PRINT
500 FOR X = 1 TO 10
600 PRINT S$(TO INT(VAL F$))
700 NEXT X
800 PRINT "====================
    ==========="
900 PRINT "00000000011111111122222222223"
950 PRINT "12345678901234567890123456789O"
```

Cuando ejecute este programa, podría digitar una expresión algebraica con variable X como la siguiente: 2*X + 3/X y vería en la pantalla un diagrama de barras de dicha función. En este programa el eje *y* es horizontal, el eje *x* vertical y los gráficos son del tamaño de un pixel. No se requerirían muchas más instrucciones para dar una escala a los valores de *x* e *y*, inmediatamente después de dar entrada a la función de *x*, y luego hacer ajustes de escala, imprimir ejes y crear gráficos en alta resolución. En resultado sería un paquete gráfico muy llamativo; sólo que el fragmento anterior se ha diseñado simplemente para clarificar aún más la gran utilidad del poder de VAL para aceptar expresiones de argumento.

En este sentido, GOSUB y GOTO son iguales que VAL. También aquí se admiten expresiones cuando la mayoría de las versiones de BASIC exigen que sus argumentos sean números de línea concretos. Se consiguen varias ventajas. Puede, por ejemplo, ponerles nombre a sus subrutinas, definir variables con los mismos nombres y valores apropiados, y después ir hacia las subrutinas (GOSUB) por su nombre. He aquí un ejemplo de esta capacidad:

```
100 LET INIC = 1000
200 LET RESULT = 2000
300 LET CALCULO = 3000
400 GOSUB INIC
500 GOSUB CALCULO
600 GOSUB RESULT
700 STOP
1000 REM******* S/R INIC********
.....
2000 REM***** S/R RESULT********
.....
3000 REM**** S/R CALCULO******
```

que casi hace que su programa se documente a sí mismo. Si reemplazara GOSUB INIC por GOSUB (VAR1 + N*VAR2), o por cualquier expresión numérica válida, se evaluaría la expresión y se trataría al resultado como un número de línea. Además, en Sinclair, si el argumento de GOTO o GOSUB es un número de línea que no existe, entonces el control pasa al siguiente número de línea válido. Si encuentra, por ejemplo, GOTO 17, y la línea 17 no existe pero sí existe la línea 18, entonces el control pasará a la línea 18 y no se indicará ningún error, como ocurriría en la mayoría de las versiones de BASIC.

La capacidad del BASIC de Sinclair para manipular estos "saltos calculados" compensa la carencia de estructuras ON...GOTO y ON...GOSUB. En otro BASIC se podría escribir:

```
2360 ON D GOSUB 100,200,300,400,500
```

para dar a entender que si el valor de D es 1, entonces el programa saltará a la línea 100 (GOSUB 100), si D = 2 entonces saltará a la línea 200 (GOSUB 200) y así sucesivamente. En el BASIC de Sinclair estaría capacitado para escribir:

```
2360 GOSUB (100*D)
```

para producir exactamente el mismo resultado. Sin embargo, en ambas versiones es imprescindible prever lo que sucedería si el valor de D no fuera entero, si fuera menor que uno, o mayor que cinco.

El Spectrum no admite las estructuras de control REPEAT...UNTIL ni WHILE...WEND. Éstas, no obstante, se pueden simular de diversas formas. La estructura REPEAT (repetir) es un bucle que empieza en la palabra REPEAT y termina en la palabra UNTIL (hasta), que es seguida por una sentencia condicionada. Si esta condición es verdadera, entonces el bucle se acaba y el control pasa a la sentencia que sigue a UNTIL. En el Spectrum, esta estructura de control se puede imitar de esta manera:

```
100 DATA "A","B","C","D","E","*","F","G","*"
200 FOR L = 1 TO 1
300 READ X$
400 PRINT X$
500 IF X$ <> "*" THEN LET L = 0
550 NEXT L
600 PRINT"FIN DE DATOS"
```

El problema de la estructura REPEAT...UNTIL es que, debido a que la condición de salida se verifica al final del bucle, éste siempre se ejecutará al menos una vez. Si no se desea que resulte de esta forma, entonces se debe utilizar la estructura WHILE...WEND.

Esta estructura es un bucle que empieza en la sentencia WHILE (mientras), a la que sigue una condición y que termina en la sentencia W(HILE)END (fin de WHILE). En la medida en que la condición se cumpla, las sentencias del programa entre WHILE y WEND se ejecutarán repetidamente. Cuando la condición no se cumpla, el control pasa de la sentencia WHILE a la sentencia que sigue a WEND (saltándose las sentencias entre ambas). El programa que dimos, por consiguiente, sería así:

```
100 DATA "A","B","C","D","E","*","F","G","*"
200 WHILE X$ <> "*" DO
300 READ X$
400 PRINT X$
500 WEND
600 PRINT"FIN DE DATOS"
```

En las máquinas Sinclair esto se podría sustituir por una estructura IF...THEN...GOTO, pero otra alternativa es:

```
50 LET X$ = ""
100 DATA "A","B","C","D","E","*","F","G","*"
200 LET TEST = (X$ <> "*")
250 FOR L = 1 TO TEST
300 READ X$
400 PRINT X$
500 LET L = (X$ = "*")
550 NEXT L
600 PRINT "FIN DE DATOS"
```

Hemos examinado las principales variantes del BASIC de Sinclair. Existen otras pequeñas diferencias, algunas de las cuales son trabas más que genuinas modificaciones, pero con cuidado y con el manual podrá hacerles frente.

Mecanografía visual

El tratamiento de textos es la aplicación reina de la microinformática profesional; paquetes de software como el WordStar están entre los más vendidos en todo el mundo

En su forma más simple, un procesador de textos no es más que una perfecta máquina de escribir. La pantalla del monitor sustituye al papel, y la tecla DELETE actúa como una goma de borrar. Para el usuario de gestión, lo más deseable de un sistema para tratamiento de textos en su capacidad para guardar documentos (en disco o en cinta de cassette), permitiéndole luego que esos documentos (que pueden ser líneas individuales, oraciones, párrafos o páginas enteras) se incorporen a un texto recién dictado. Esto es de particular utilidad en la producción de especificaciones, contratos u otra clase de documentos que emplean el mismo texto una y otra vez.

Cuando ya se conoce cómo crear, modificar y manipular un documento, almacenarlo y recuperarlo, el siguiente aspecto a tratar es cómo reacomodar el texto en la página. Existen tres formas convencionales de disponer el texto: alineado sobre la izquierda (cada línea deja el mismo margen izquierdo), alineado sobre la derecha, o justificado. En el texto impreso justificado, los espacios entre palabras y letras están dispuestos de tal modo que cada una de las líneas poseen la misma longitud. Sin embargo, la mayoría de las impresoras para microordenadores que hay en el mercado sólo varían los espacios en blanco entre palabras cuando componen texto justificado.

Igualmente un paquete para tratamiento de textos debería ser capaz de buscar a través de todo el documento una palabra o una frase determinada y sustituirla por otra. Esto es particularmente útil cuando se ha escrito de manera reiterada con algún error de ortografía una misma palabra, o cuando se desea enviar correspondencia personalizada a alguien determinado. En términos técnicos, WordStar es un procesador y editor de textos activado por menú, con órdenes de formato transparentes, que funciona en un medio CP/M y requiere la utilización de la tecla Control para permitir que la máquina distinga entre una orden y un carácter. Ello significa que el operador puede elegir tener a su disposición un menú de opciones que esté visualizado constantemente en la pantalla, en este caso en una sección reservada de diez líneas en la parte superior. Las órdenes que afectan a la forma en que aparece el texto (ya sea en el momento de entrada o durante la impresión) se visualizan en la pantalla dentro del cuerpo principal, pero la impresora no las reproduce; la responsabilidad de la manipulación de archivos y "administración" la asume el sistema operativo.

Una vez cargado, WordStar presenta al usuario un "menú" que permite seleccionar la función requerida. El usuario puede crear o editar un archivo, ya sea como documento o como "no documento" (que se utiliza, p. ej., en los programas en lenguaje ensamblador que se van a compilar, o para el texto que se vaya a utilizar con otro paquete de software); puede también mezclar archivos entre sí (imprimiéndolos o no); cambiar la unidad de disco en uso de A a B, o viceversa; eliminar, copiar o darle un nombre nuevo a un archivo; salir al sistema operativo, o establecer el nivel de ayuda requerido.

Cada opción se selecciona mediante una sola pulsación de tecla (no se necesita la tecla Control), lo que conduce al usuario hasta un menú más pequeño, específico para la función en uso. La última de estas opciones (para determinar el nivel de ayuda) le permite al usuario seleccionar en qué medida se visualizará en la pantalla el menú de órdenes disponibles. Cuanto más bajo sea el nivel de ayuda requerido, menos serán las líneas reservadas en la

El menú de archivos

El menú de ayuda

El menú de archivos
Para empezar, WordStar presenta un menú que revela la gama de opciones de sus funciones básicas. El usuario puede crear o editar ya sea un archivo de documentos utilizando todos los parámetros de formateo estándar, como traslado de palabras, justificación, etc., o un archivo de no documentos, o copiar, dar un nombre nuevo o borrar un archivo. En resumen, esta sección del programa se ocupa de las rutinas de administración. Estas fotografías corresponden al Osborne-1, que visualiza sólo 52 columnas de texto de las 80 existentes, cortando por consiguiente sobre el lado derecho

El menú de ayuda
Este menú, al que se accede mediante el código de Control (J), le ofrece al usuario una serie de explicaciones en pantalla bastante comprensibles acerca de la forma en que funcionan ciertas secciones del paquete. Coloquialmente se lo conoce como menú de ayuda. El código de control J seguido de uno de los caracteres subsidiarios (H, B, F, D, etc.) ofrece una explicación más extensa de estas secciones del WordStar

pantalla para el menú, y más líneas quedarán reservadas para la edición de texto. La mayoría de las órdenes de tecla con iniciales son siglas, pues se tratan de un único carácter. Por ejemplo, D crea un archivo para "documento" y N crea un archivo para "no documento".

Lo primero que solicita el sistema es un nombre para el archivo que se va a crear. Una vez dado un nombre al archivo, el sistema pasa al menú de trabajo, que consta de indicaciones para detalles como movimiento del cursor, inserción y eliminación, y recordatorios de los códigos de control de un único carácter utilizados para dar entrada a otras bifurcaciones del menú principal (el menú de formato, p. ej.). El WordStar no da por sentado que la máquina en la cual se va a ejecutar (el ordenador "objeto") posee teclas para control del cursor.

El movimiento de un solo espacio se define mediante la tecla Control en conjunción con las teclas E, S, D y X (las máquinas que poseen teclas para control del cursor normalmente tienen instalado el WordStar, de modo que las funciones se duplican). El WordStar es bastante eficaz en su control del cursor, acomodando el movimiento de éste por carácter, palabra, línea o párrafo en cualquier dirección. Para la supresión rige casi el mismo principio. Los caracteres individuales se pueden borrar tanto en la posición del cursor como en la posición inmediatamente a su derecha; y se pueden suprimir palabras enteras a la derecha de la posición del cursor o líneas enteras (independientemente de la posición del cursor). También se puede "enrollar" la pantalla utilizando las funciones de control, sin

El menú de órdenes rápidas

El menú de bloques

Liz Heaney

El menú de órdenes rápidas
A los menús secundarios del WordStar se accede mediante un código de control de un solo carácter, que hasta cierto punto es acrónimo. El menú Q, por ejemplo, equivale a decir en forma abreviada el menú Quick (rápido). El menú Q proporciona un resumen de las teclas de control del cursor, así como las funciones de buscar y sustituir del paquete

El menú de bloques
Algunas de las siglas de un solo carácter son algo arbitrarias; K, por ejemplo, alude a *blocK* (bloque). Controla cómo se guardan y utilizan los archivos, el movimiento de bloques de texto y algunas estructuras de acceso a los archivos. Observe que reproduce, asimismo, algunas de las rutinas de administración estándar del paquete

afectar el posicionamiento del cursor. Esto es muy útil cuando se quiere revisar el texto.

Por último, el usuario puede decidir si insertar un carácter en la posición del cursor o reemplazar el que está escrito por otro que desea introducir desde el teclado. Esta función se realiza mediante la presión de la tecla apropiada (V, en este caso), que cambia el estado de inserción a sustitución, y así permanece hasta que se vuelve a pulsar la tecla V (junto con la tecla Control), para invertirlo.

El paquete se encarga automáticamente de la paginación y del traslado de palabras al final de las líneas. La paginación define el número de líneas por página y el traslado de palabras es el desplazamiento automático de palabras a la línea siguiente.

Todas las funciones que hemos descrito son las necesarias normalmente para crear y editar documentos, pero el WordStar posee otras configuraciones. Puede centrar líneas en la página; posibilitar golpeteos múltiples para obtener caracteres en negrita; poner caracteres sobrescritos y subscritos; sobreimprimir la línea anterior, o subrayar y tachar caracteres individuales. Todas estas funciones son utilizables en el momento de dar entrada al texto o durante la edición.

A la función "buscar y sustituir" se accede a través del menú Q, que también contiene el posicionamiento extra del cursor y las funciones de borrado. El usuario recibe aviso de que dé entrada a la serie de caracteres que componen la palabra o la frase a buscar. El usuario puede mencionar una instancia o bien todas las instancias, y decidir si la serie localizada sencillamente la debe señalar el cursor, o si se ha de sustituir y, de ser de esta forma, si esa sustitución ha de efectuarse sin que medie ninguna otra instrucción.

Igualmente, se pueden desplazar de un lugar del documento a otro bloques marcados de texto mediante diversas funciones del menú K, que de otro modo se ocupa de manipular el archivo. J, O y P, los otros tres menús subsidiarios del WordStar, se encargan, respectivamente, de ayudar al usuario a crear o editar un documento, del formateo y ajuste de los documentos, y de la forma en que aparecerán al imprimirse.

En líneas generales, el WordStar es un paquete de software para microordenadores profesional y amplio. Son muy pocas las funciones que no puede realizar satisfactoriamente, y está definido de tal forma que resulta accesible para una amplia variedad de máquinas. A primera vista podría parecer complicado y diferente, pero gracias a sus exhaustivos menús hasta los principiantes pueden utilizarlo.

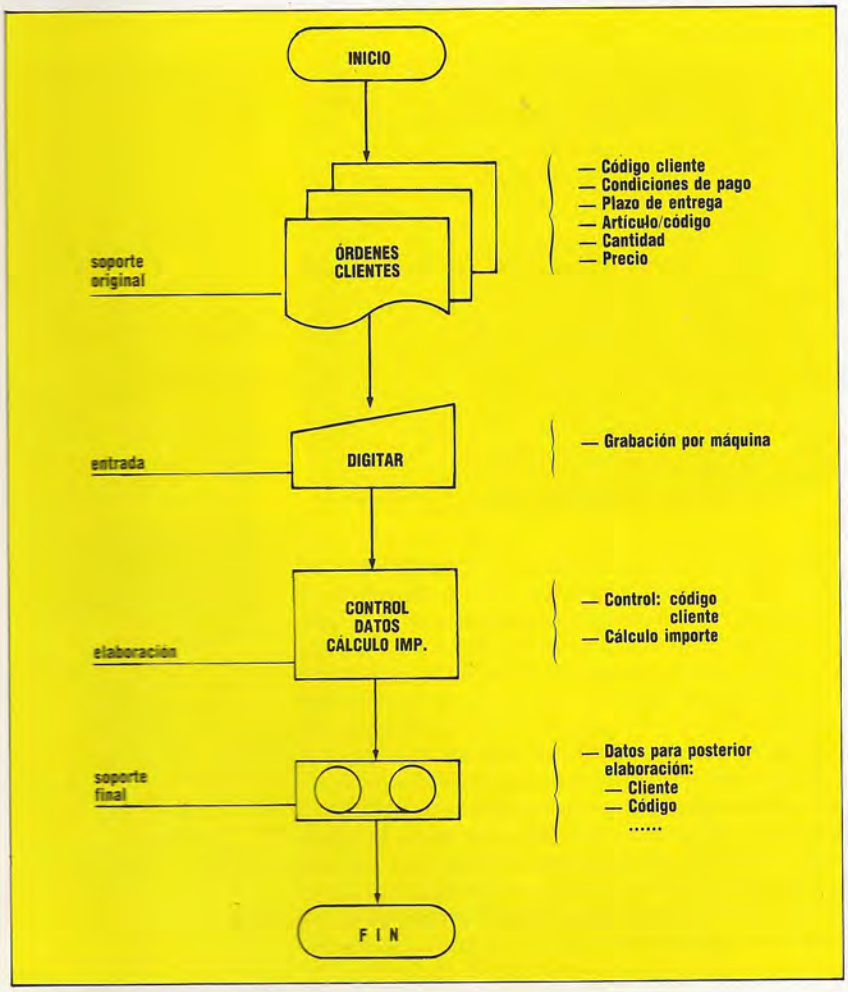
Un importante número de los microordenadores que admiten el WordStar poseen teclas de función programable como estándar. El Osborne-1, por ejemplo, con el cual hemos visualizado nuestros ejemplos, le ofrece al usuario 10 de estas teclas y reserva 256 bytes para sus contenidos. Estas funciones (a las que se accede mediante la tecla Control y una tecla numérica que se pulsa simultáneamente) pueden contener una serie de caracteres de control, o una palabra o frase que se emplee comúnmente, y posibilitar que esa serie de caracteres se incorpore mediante apenas dos pulsaciones de teclas.

Diagramas de flujo

Los diagramas de flujo constituyen un punto de referencia para el analista que acaba de concluir un primer estudio global del problema

Ya hemos presentado los diagramas como la secuencia lógica de operaciones a seguir que forman un programa. Uno de los propósitos confiados al análisis es planificar la combinación de los diferentes componentes del sistema, para alcanzar, en el menor tiempo posible, los mejores resultados. Debemos analizar el modo cómo fluyen los datos a través de los citados componentes, y, tras individualizar esta corriente de datos por medio del sistema, se generaliza la secuencia de las operaciones necesarias para poder codificar los símbolos. Está claro que esta parte del análisis no permite por sí sola confeccionar un programa.

La diagramación, aplicada a la automatización de la información, es en sí, como término, demasiado genérica. En cuanto a los diagramas, empecemos distinguiendo dos tipos básicos: *diagrama de flujo*, llamado también *flow-chart* de sistema, y *diagrama de bloques*, o *flow-chart* de programa. Aun tratándose de la resolución de un mismo problema, en realidad cada uno cumple una función específica, puesto que ambos responden a diferentes exigencias, aunque complementarias entre sí.



Un diagrama de flujo busca la representación gráfica del paso de los datos, explicando, a grandes rasgos, la resolución y su proceso. Define igualmente la configuración necesaria para resolver el problema, mencionando sintéticamente las operaciones, pero sin detallar los movimientos de cada frase, con lo que, a cambio de una visión global, nos hallamos faltos de información específica para poder transcribir y crear el programa.

Pero ¿para qué sirve un diagrama de flujo? Podemos resumirlo en estos puntos: describe todas las fases de un proceso; asigna y define aquellos componentes del sistema que se utilizarán en cada una de las fases; relaciona las fases y sus componentes.

Supongamos que una pequeña empresa tiene la intención de automatizar su sistema de pedidos y sustituir sus archivos tradicionales por uno informático. La persona encargada de esta tarea (el analista) recibe una serie de explicaciones acerca de la manera en que se había desarrollado el trabajo hasta la fecha: "Los clientes hacen los pedidos que nosotros recogemos en unos albaranes. Éstos contienen datos tales como número de código, nombre del cliente, artículo, número de unidades, etc. Buscamos después en los archivos la ficha del cliente y una vez cotejados los datos, se le aplican las condiciones de pago, entrega y otros extremos que en ella figuren. Enseguida calculamos el importe del pedido. Finalmente archivamos, corregida, la ficha".

Nuestro analista, que bien pudiera ser usted, dando por comprendido el método expuesto y habiéndose cerciorado de que la empresa no desea introducir variaciones de ningún tipo, estudió el problema llegando al siguiente planteamiento global. Según éste, los pasos esenciales a desarrollar han de ser:

Recibir el soporte original formado por las órdenes de pedido

Introducir los datos en el ordenador

Realizar las pertinentes comprobaciones y una vez hechos los controles, calcular a cuánto asciende el total del pedido. Este paso se realizará íntegramente de forma automática

Proceder a la grabación del registro correspondiente al cliente, debidamente actualizado en el soporte (magnético) final

Éste fue el planteamiento que pudo entregar usted a la figurada empresa con el diagrama que aparece aquí al lado.

Una buena imagen

La superior calidad de imagen que proporciona un monitor respecto a un televisor puede ser decisiva para un ordenador que se desea emplear en gestión empresarial o en la enseñanza

Entre los periféricos más importantes se encuentra la pantalla. Es frecuente el uso de un televisor o un aparato de bajo coste en blanco y negro adquirido especialmente pero, aunque la calidad obtenida sea aceptable, no es todo lo buena que debiera ser.

Esto se debe a que la señal, al pasar de la memoria de pantalla del interior del ordenador a la pantalla, debe someterse a varias etapas de codificación y decodificación. Con independencia de la bondad del sistema de circuitos usado, inevitablemente será poco perfecto, con una imagen final a menudo borrosa y difícil de leer, afectada por un desagradable rehílo lumínico, que se conoce como "hormigueo de puntos".

El secreto en una visualización de calidad está, pues, en eliminar estas distorsiones de señales; y dado que el elemento que las produce es el sistema de circuitos de modulación y demodulación, se trata sencillamente de no incluir dichos sistemas en el proceso.

En realidad esta tarea es la que realiza un monitor, que no consiste más que en un tubo catódico sin los decodificadores de televisión, un sistema simplificado que tiene capacidad para generar imágenes más nítidas, más brillantes y, por lo general, más estables.

Puesto que no lleva decodificadores de televisión, el monitor no funcionará si estuviera conectado a la salida TV de su ordenador. Necesita una salida "video". Puede que en su ordenador no se llame de este modo, pero de todas formas el detalle más importante es que debe tratarse de una salida que no pase a través del modulador y para comprobar esta característica usted debe consultar el manual de su ordenador.

El proceso que genera la imagen en una pantalla de televisión consiste en asegurarse de que todo suceda en el momento correcto. El problema radica en el hecho de que el barrido del haz a través de la superficie del tubo se genera dentro del monitor y, por lo tanto, es inaccesible para el dispositivo activador del propio ordenador.

Estando encendido pero sin conectar a una entrada, un monitor explorará el haz sobre la pantalla entera 50 veces por segundo, produciendo un campo perfectamente iluminado. Convertir éste en una imagen implica encender y apagar este haz exactamente en el mismo lugar cada 1/50 de segundo; cualquier inestabilidad que se produzca durante este proceso se traducirá en un cabrilleo molesto, que hace que mirar la pantalla resulte, en el mejor de los casos, fatigoso y, en el peor, absolutamente inútil.

Todo el proceso depende de *impulsos de sincronización* que, junto con la señal de brillo, son producidos directamente por el ordenador y salen hacia el monitor.

Hay dos tipos de impulsos de sincronización: uno para cada línea de la imagen y otro para cada imagen completa. Al final de cada ciclo completo se le envía al monitor un impulso corto que le indica que ahora el fotograma está completo y que el haz de electrones (y, por tanto, el punto que produce) ha de regresar a la esquina superior izquierda del fotograma para repetir el ciclo.

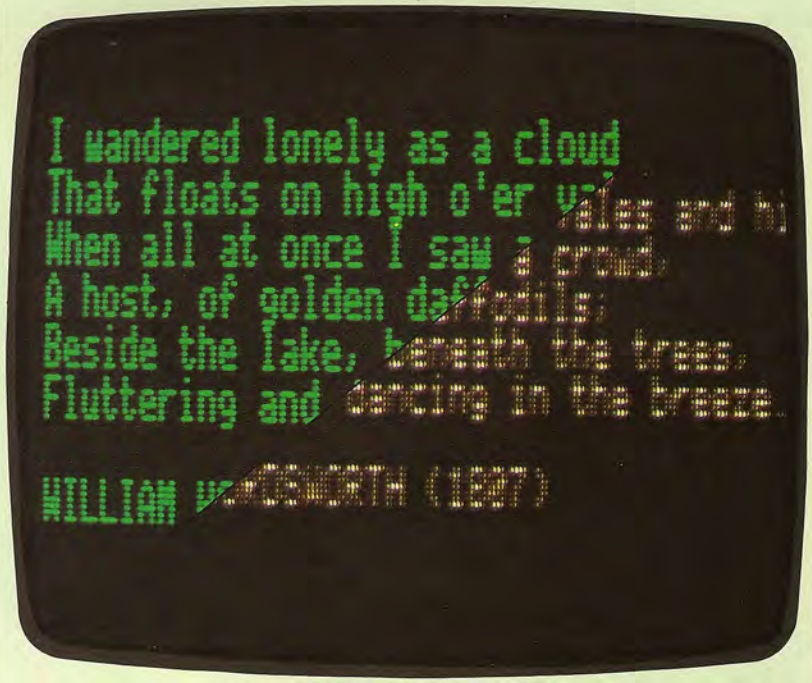
Algo similar se produce al final de cada línea. El monitor recibe un impulso que le indica que una línea está completa y que el haz de electrones debe regresar al lado izquierdo de la pantalla para dar comienzo a la línea siguiente.

Hay muchos modelos de monitores, y sólo dos clases: en color y monocromos, que a su vez se dividen según los diferentes tipos de señales que aceptan.

Los monitores monocromáticos son bastante sencillos y el único tipo importante es el que acepta

Especialista en imágenes

Esta pantalla ilustra la diferencia entre la calidad de imagen de un monitor monocromático (en este caso el Monitor III de Apple) y un buen televisor doméstico utilizado para visualizar la salida de un paquete para tratamiento de textos



Ian Mckinnell

una señal denominada *compuesta*, así llamada porque los diversos impulsos de sincronización se combinan con el nivel de brillo en una señal, que el monitor clasifica después para la producción de la imagen.

Monitores en color

Algunos monitores en color utilizan un esquema similar, pero en virtud de la mayor complejidad de la señal están más emparentados con los aparatos de televisión y operan casi con los mismos sistemas. Los tipos principales son PAL, SECAM y NTSC, nombres de los sistemas de color que se utilizan en las emisoras de televisión de diferentes partes del mundo. Representan métodos de codificar los tres colores "aditivos" (rojo, verde y azul) y los dos impulsos de sincronización.

En algunos casos, las diversas señales se pueden enviar al monitor por separado. A pesar de que en los esquemas utilizados existen sutiles variaciones, genéricamente se conocen como RGB por las pistolas de color (*Red, Green, Blue*: rojo, verde, azul). El más simple es el TTL (*Transistor-Transistor logic*: lógica transistor-transistor), en el cual los colores sólo poseen dos estados, encendido o apagado, de modo que en diversas combinaciones pueden producir los ocho colores familiares que se aprecian en los sistemas de videotext como el Prestel y el Ceefax.

Es posible obtener más colores si se dota de intensidad a cada color aditivo; aunque tal intensidad suele producirse por pasos discretos (digitales), el

Anchura de banda

La anchura de banda es un extremo importante a la hora de elegir un monitor. Mide la pequeñez con que el sistema de circuitos puede producir un punto. Según esto, cuanto más pequeño sea el punto, mejor será; pero la conclusión sobre qué anchura de banda es mejor para una máquina determinada se basa también en otras consideraciones.

Antes de nada, usted debe calcular cuál es la anchura de banda mínima absoluta para su máquina. Ésta se calcula multiplicando el máximo número posible de líneas de caracteres en pantalla por el número de los caracteres de cada línea, lo que da el número total de caracteres. Esta cifra se multiplica, a su vez, por la anchura en puntos de la matriz de caracteres, y después por la profundidad de la matriz.

El resultado obtenido es el número máximo de puntos en pantalla, que por lo general se sitúa entre los 10 000 y 1 500 000. En una pantalla normal de 80 × 24 con una matriz de 7 × 9, la cifra es de 110 960. Todos estos puntos los ilumina el barrido en la pantalla a la velocidad de 50 veces por segundo (60 en Estados Unidos y otros países), lo que significa que el número de puntos de la pantalla se debe multiplicar por esta cifra. El resultado generalmente estará comprendido entre 500 000 y 75 000 000, que es el número de veces que el sistema de circuito de control debe poder encender y apagar el haz por segundo.

Como se trata de una frecuencia se expresa en hertzios (Hz) y, al igual que casi todas las magnitudes medidas por millones, las unidades son megahertzios o megas, para abreviarlo.

Los monitores se venden con anchuras de banda de 5, 7, 10, 12 y 15 MHz e incluso es posible que de 20 MHz, si bien el precio aumenta notablemente a medida que la cifra se eleva. Se pueden construir monitores que operen a 100 MHz, pero su precio sería excesivo.

Por lo general, a mayor anchura de banda del monitor, más nítida será la imagen, si bien una resolución excesivamente alta puede llegar a ser tan incómoda como una demasiado baja, en especial para los gráficos en color.

Si la anchura de banda del monitor es muy pequeña para el ordenador, las letras serán desdibujadas y difíciles de leer, como si la señal se enturbiara. Si, por el contrario, es demasiado grande, los puntos individuales se separarán con excesiva claridad, dividiendo la imagen.

Monitores monocromáticos
De todos los monitores, los más sencillos y menos caros son los modelos monocromos. A un precio razonable, estas unidades le ofrecen al usuario una opción de colores de fósforo y una definición razonablemente alta. Son ideales para tratamiento de textos y otras aplicaciones de gestión, pero también se utilizan para visualizar gráficos sencillos

Monitores a tres colores
El CUB de Microvitec es un excelente ejemplo de los monitores en color menos caros que están al alcance del usuario de un ordenador personal. Si bien su resolución no es de las más altas, es bastante adecuado para diversas aplicaciones, incluyendo la mayoría de los gráficos

Versatilidad japonesa
Con su capacidad para diferenciar la naturaleza de la señal recibida, el TM90PSN de JVC es especialmente valioso para quienes desean que su monitor tenga aplicaciones diferentes. Además de su empleo como dispositivo de salida de ordenador, también puede servir como aparato de video o reproductor de videodisco

Televisores domésticos
La calidad de imagen de los televisores domésticos ha ido mejorando rápidamente con los adelantos en la tecnología de la construcción de tubos de rayos catódicos y la alineación de cañones de electrones. Sony, por ejemplo, ha tomado buena nota del doble servicio a que se destinan sus aparatos al instalar el conector de la antena en la parte delantera de los mismos

Chris Stevens

resultado se conoce como monitor *analógico* o *lineal*.

Los monitores en blanco y negro de nueve pulgadas suelen salir por la mitad del precio de uno con pantalla verde fósforo de 12 pulgadas. Los monitores en color son más caros, debido al mayor coste del tubo. Los monitores estándar, como el Microvitec y el Kaga, doblan el precio a los anteriores, dependiendo en gran medida de las dimensiones de la pantalla y de la anchura de banda. La mayoría de los monitores en color se pueden hallar tanto en versión TTL como analógica, con poca diferencia en cuanto se refiere al precio.

El TM90PSN es un monitor interesante y flexible que produce la firma JVC. Es más pequeño que la mayoría, con una pantalla de 10 pulgadas y, aunque la anchura de banda no es particularmente alta (lo cual limita la máxima resolución), puede aceptar casi cualquier tipo de entrada, desde la compuesta monocromática normal, pasando por la TTL y la RGB analógica, hasta cualquiera de las cuatro señales a color compuestas, PAL, SECAM, NTSC 3,58 y NTSC 4,43. Posee en su interior un mecanismo selector que verifica la señal entrante y automáticamente conmuta el monitor a la modalidad adecuada.

Este tipo de monitor se irá haciendo cada vez más común, y puede servir como salida con fines generales para ordenadores, aparatos de video, reproductores de videodiscos y otros tipos de máquinas.

Persistencia y color

Un factor a tener en cuenta en la elección de un monitor monocromático es el tipo de fósforo utilizado. Ésta es la sustancia polvorienta que reviste la cara interior de la pantalla y tiene la propiedad de ponerse incandescente al ser alcanzada por un haz de electrones, produciendo de este modo la imagen.

Las principales consideraciones atañen al color y a la "persistencia". La primera se explica por sí misma, pero la persistencia se suele comprender menos. Se trata de una medida del tiempo que dura el fósforo incandescente después de que el haz ha pasado por él. Rara vez se facilita una cifra exacta, pues por lo general basta con indicar persistencia *larga* o *corta*.

Los fósforos que se utilizan en los televisores y virtualmente en todos los monitores en color tienen una persistencia corta, al igual que la mayoría de los monitores monocromáticos, pero en muchas aplicaciones es preferible que la imagen continúe brillando durante una fracción de segundo, porque ello reduce el "pestañeo" de la pantalla, una de las principales causas de la fatiga visual.

El ejemplo más conocido del fósforo de persistencia larga es el de las pantallas de radar, en el cual el barrido radial del haz deja una larga huella de incandescencia, confiriéndole a la pantalla una especie de "memoria" sin necesidad de una complicada electrónica.

No obstante, si el sistema emplea un lápiz óptico, la persistencia larga es un riesgo; incluso lo haría inutilizable, debido a que su operatividad depende de que la imagen desaparezca con la rapidez suficiente como para que el ordenador localice el lápiz. Si el fósforo continúa incandescente, el ordenador verá luz, independientemente de dónde señale el lápiz y sin considerar tampoco si el haz está en realidad explorando ese punto o no. Por lo tanto el ordenador no será capaz de calcular la posición del lápiz óptico valiéndose de su conocimiento de la posición explorada en un momento dado.

Existe toda una gama de tipos y colores de fósforo, según la utilización a que se destine el monitor y la preferencia del comprador. Se puede producir casi cualquier color. El fósforo blanco de persistencia corta es barato y fácil de conseguir, pero uno de los tonos de verde es casi tan común como el blanco y es mucho más descansado para la vista, al igual que el ámbar. El azul se emplea en muchos terminales de unidad principal como los utilizados en las agencias de viajes y en los mostradores de las líneas aéreas, mientras que el rojo se utiliza en las salas de radar y otros lugares en los cuales la visión nocturna no debe ser perjudicada.

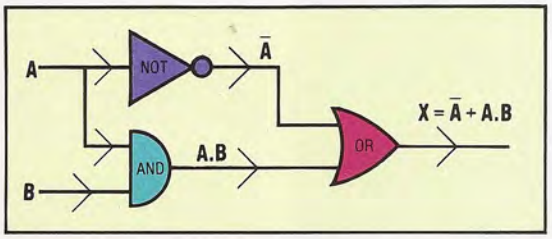
Los bloques lógicos de la suma

Veamos cómo los bloques lógicos ADN, OR y NOT se convierten en circuitos lógicos capaces de sumar

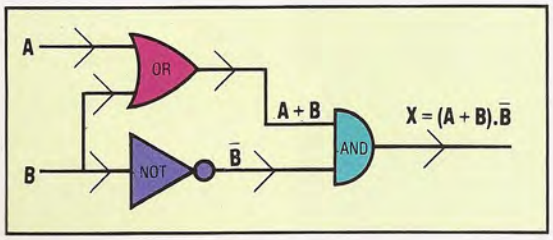
Al sistema que utiliza la notación algebraica para describir las relaciones lógicas se denomina *álgebra booleana*, en honor del lógico y matemático británico George Boole (1815-1864).

El álgebra booleana es de gran valor para el diseño de circuitos de ordenador porque permite la simplificación matemática de los circuitos lógicos. Esto significa que para realizar una función determinada se requieren menos puertas lógicas, lo que a su vez aumenta la velocidad de operación de la máquina.

Ya nos hemos encontrado con la notación booleana para la salida de las tres puertas lógicas básicas: AND (A.B), OR (A + B) y NOT ($\bar{A}$). Utilizando estas tres expresiones se pueden representar circuitos más complejos. Por ejemplo, la expresión booleana $X = \bar{A} + A.B$ representa el siguiente circuito:



Es importante observar que el orden en que se efectúan las operaciones AND y OR es decisivo. Una sencilla regla dice que AND tiene prioridad sobre OR (así como sobre NOT). Si se desea invertir el orden de prioridad, se deben utilizar paréntesis, como en este ejemplo: $X = (A + B).\bar{B}$. Para esta expresión, las dos entradas se someten a OR antes de someterse a AND con el negativo de $\bar{B}$. Éste es el diagrama del circuito:



Al dibujar un circuito lógico a partir de su expresión booleana suele ser mejor comenzar por la salida e ir trabajando hacia atrás hasta las entradas. Con este método se consiguen mejores trazados.

OR inclusivo y OR exclusivo

En inglés la partícula OR (como en castellano, o) tiene dos posibles connotaciones. De la primera ya hemos hablado: Éste OR (o) ése OR (o) ambos.

La segunda acepción tiene importantes consecuencias para el diseño de circuitos lógicos: Éste OR (o) ése, pero no ambos.

Por ejemplo, si le dicen que un amigo está siguiendo un partido de fútbol por televisión OR (o) por radio, puede entender que lo está siguiendo por ambos medios a la vez (caso de uso *inclusivo*). Por otra parte, su amigo puede ser alto OR (o) puede ser bajo, pero no ambas cosas (uso *exclusivo*).

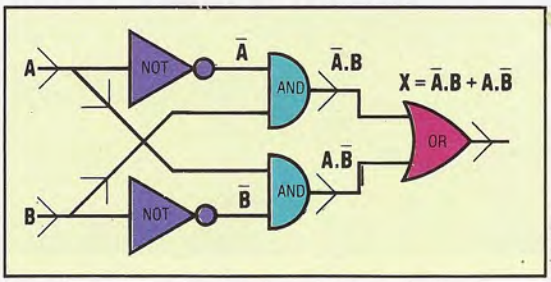
En los circuitos lógicos, la operación excluyente de OR se indica por XOR, y puede construirse a partir de juegos de puertas AND, OR y NOT. La tabla de verdad para XOR es:

ENTRADAS		SALIDAS
A	B	$A \veebar B$
0	0	0
0	1	1
1	0	1
1	1	0

Como se puede ver en las filas segunda y tercera, la salida será VERDAD sólo si enfrentamos:

NOT(A) AND B
OR
A AND NOT(B)

A lo que es posible darle la siguiente expresión booleana: $X = \bar{A}.B + A.\bar{B}$. Un circuito para producir la operación *OR exclusivo* sería:



Liz Dixon

Lo que nos da un circuito de cinco puertas. Más adelante veremos cómo se puede convertir este circuito en uno de sólo cuatro puertas.

Puertas lógicas y aritmética

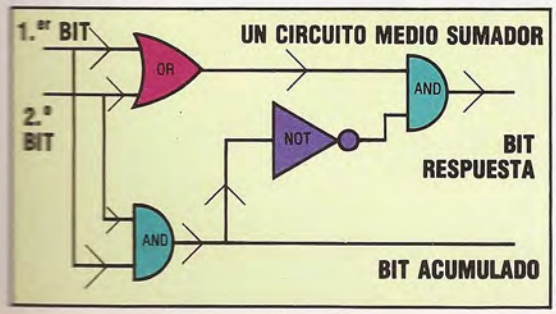
Aunque la mayoría de los ordenadores personales pueden efectuar la gama completa de operaciones aritméticas, sólo la suma se efectúa directamente mediante circuitos lógicos. Las restantes —resta, multiplicación y división— se realizan combinando hardware de circuitos "sumadores" con software para controlar el movimiento de los patrones de bits. Pero antes de que podamos analizar los circuitos que realizarán la suma binaria, hemos de echar un vistazo al proceso en sí mismo. Sea la siguiente suma binaria 101 + 111 (véase p. 55):

	8	4	2	1
	0	**1**	**0**	**1**
	0	**1**	**1**	**1**
	1	**1**	**0**	**0**
RESULTADO ACUMULADO	1	1	1	

Examinando una columna por sí misma (sea la columna de los doses) podemos listar sus posibles entradas y salidas. Las *entradas* son: los dos bits a sumar y el bit acumulado de la columna anterior. Las *salidas* son: el bit colocado como resultado en la columna de los doses, y el bit a llevar a la columna siguiente. El dispositivo capaz de aceptar estas entradas y producir las salidas correctas se llama *sumador*. Este dispositivo es bastante complejo, de modo que empecemos por una versión ligeramente simplificada, denominada *medio sumador*. Un circuito medio sumador ignora el hecho de que puede haber un uno acumulado de la columna anterior. Así se reduce el problema a un circuito con dos entradas y dos salidas. Ahora podemos elaborar una tabla de verdad para un circuito medio sumador, que adquiere la siguiente forma:

ENTRADAS		SALIDAS	
1.er BIT	2.º BIT	BIT ACUM.	BIT RESULT.
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Es fácil ver que el bit acumulado será 1 si el primer bit AND (y) el segundo bit son los dos 1. El bit resultado se forma operando con OR (o) las entradas, excepto en el caso en que el bit acumulado sea 1. Podemos decir que la salida del bit resultado es 1 si "el primer bit es 1 OR el segundo bit es 1 AND el bit acumulado es NOT 1". El circuito que incluimos a continuación producirá las salidas deseadas:



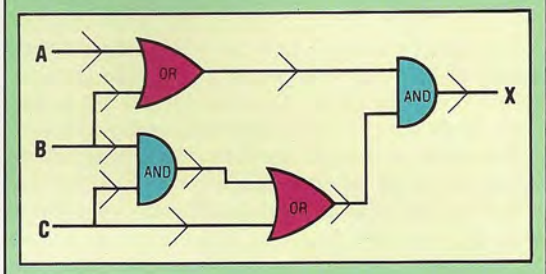
EJERCICIO 2

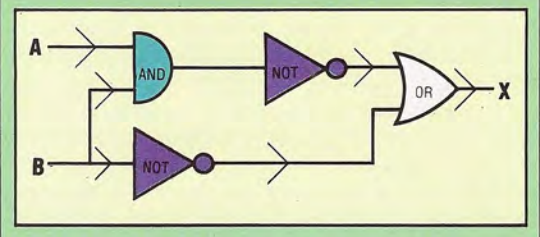
1) Dibujar los circuitos lógicos para las siguientes expresiones booleanas:

a) $X = (A + B).C$
b) $X = \overline{A}.B + (A + C)$
c) $X = \overline{A}.B + (A + B)$
d) $X = \overline{A}.\overline{B}.(A + B)$

2) Escribir las expresiones booleanas utilizando A y B como entradas para el bit acumulado y el bit respuesta de un circuito medio sumador.

3) Escribir las expresiones booleanas para estos circuitos:





Respuestas al ejercicio 1 de la página 489

1)

APROBADO EXAMEN CONDUCIR	ACOMPAÑADO CONDUCTOR CUALIF.	PODRÍA CONDUCIR
FALSO	FALSO	FALSO
FALSO	VERDADERO	VERDADERO
VERDADERO	FALSO	VERDADERO
VERDADERO	VERDADERO	VERDADERO

2)

DISPONE GRABADORA CASSETTE	DISPONE UNIDAD DISCO	ESCRITO PARA OTRA MÁQUINA	PROGRAMA SE PUEDE CARGAR
FALSO	FALSO	FALSO	FALSO
FALSO	FALSO	VERDAD.	FALSO
FALSO	VERDAD.	FALSO	VERDAD.
FALSO	VERDAD.	VERDAD.	FALSO
VERDAD.	FALSO	FALSO	VERDAD.
VERDAD.	FALSO	VERDAD.	FALSO
VERDAD.	VERDAD.	FALSO	VERDAD.
VERDAD.	VERDAD.	VERDAD.	FALSO

3)

A	B	P	Q	C
0	0	1	0	0
0	1	1	1	1
1	0	0	1	0
1	1	0	1	0

Rizar el rizo

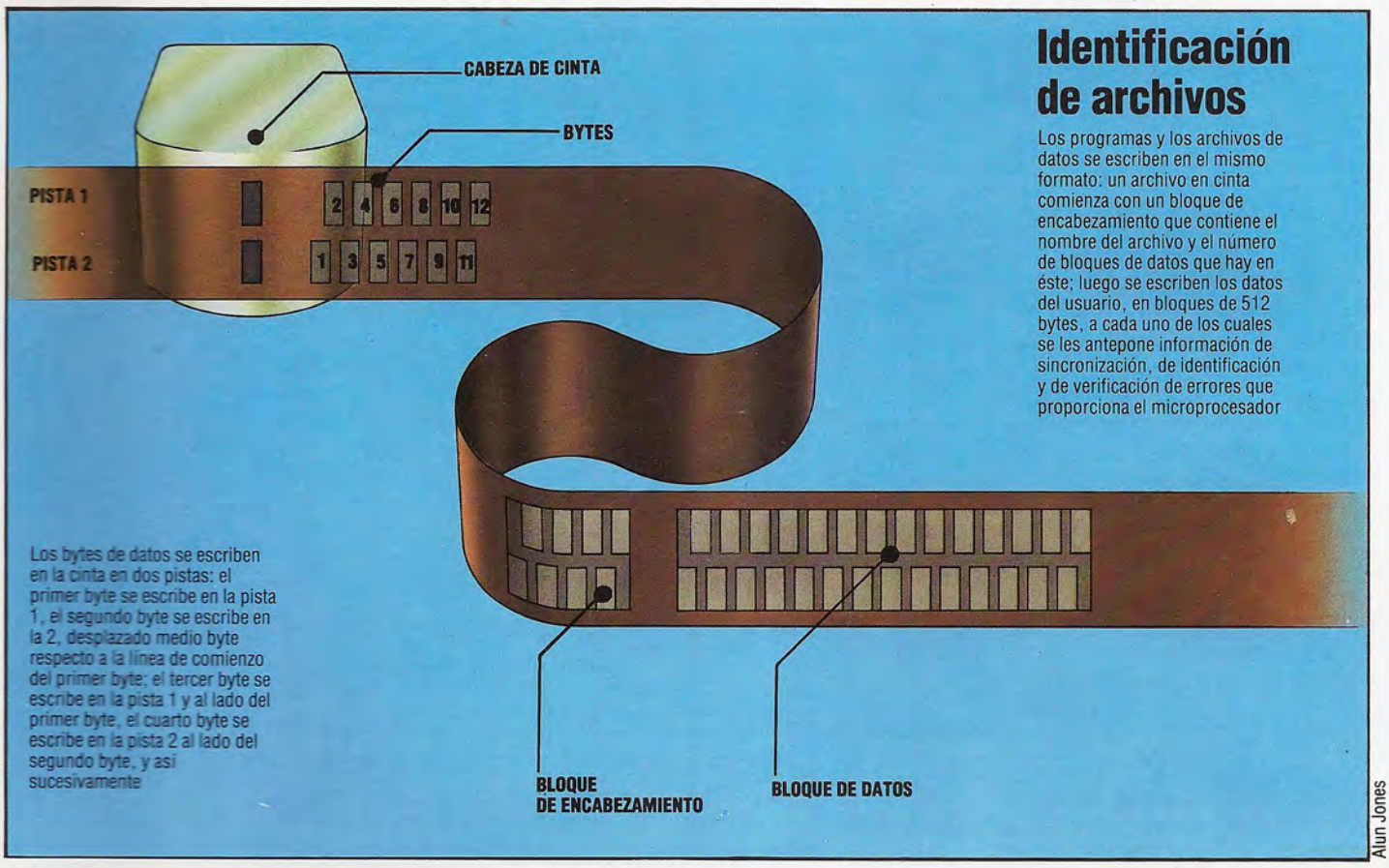
Con el Spectrum se llegó al ordenador personal potente, pequeño y barato: faltaba sólo el microdrive, el dispositivo de acceso rápido

El sistema del microdrive utiliza un bucle continuo de 200 pulgadas de cinta de video magnética de 2 mm que rota en el interior de un pequeño cartucho una vez cada siete segundos. A pesar de que en cuanto a acceso e instrucciones es bastante similar a una unidad de disco flexible estándar, en realidad es una "cinta flexible". Los datos se graban digitalmente, al contrario que en el método de cinta de audio, que graba tonos. La videocinta tiene un formato de dos pistas en las cuales se graban secuencialmente los bits según un patrón en zigzag que se superpone. Este sistema, que requiere dos cabezas de lectura-escritura, permite almacenar los datos al doble de la densidad y velocidad que en un sistema de pista única. Las pistas, además, se formatean en bloques de 512 bytes. Cada bloque contiene una descripción de los datos contenidos en el mismo y está precedido por un *encabezamiento* compuesto por 26 bytes de datos de identificación.

Al conjunto de un bloque y su correspondiente encabezamiento se los denomina *sector*. El bucle de cinta de 200 pulgadas puede almacenar casi 200 sectores, dando una densidad de almacenamiento de datos de aproximadamente 500 bytes por pulgada. Un archivo identificado por su nombre se almacena en un solo sector si su longitud es de menos de 512 bytes. Si se superpone con otro sector, o con varios, sin acabar de rellenar el último sector, el espacio inutilizado de éste se pierde hasta que el archivo se borre. De modo, entonces, que a pesar de que cada microdrive teóricamente puede almacenar 100 Kbytes de datos, en realidad su capacidad es de 85 a 90 Kbytes. El tiempo de acceso promedio para hallar y cargar un programa en la memoria del Spectrum es de entre 10 y 15 segundos.

La interface ZX

El microdrive se conecta al Spectrum a través de una interface 1 ZX, que se acopla en el conector marginal para periféricos del micro. Se pueden conectar hasta ocho microdrives en la disposición de "cadena margarita". Además de proporcionar una conexión para microdrives, la interface 1 sirve como interface RS232 estándar, como interface para red de área local y como conector para una impresora ZX. También contiene instrucciones en ROM que amplían el BASIC de Sinclair al objeto de



Identificación de archivos

Los programas y los archivos de datos se escriben en el mismo formato: un archivo en cinta comienza con un bloque de encabezamiento que contiene el nombre del archivo y el número de bloques de datos que hay en éste; luego se escriben los datos del usuario, en bloques de 512 bytes, a cada uno de los cuales se les antepone información de sincronización, de identificación y de verificación de errores que proporciona el microprocesador

Los bytes de datos se escriben en la cinta en dos pistas: el primer byte se escribe en la pista 1, el segundo byte se escribe en la 2, desplazado medio byte respecto a la línea de comienzo del primer byte; el tercer byte se escribe en la pista 1 y al lado del primer byte, el cuarto byte se escribe en la pista 2 al lado del segundo byte, y así sucesivamente

Alun Jones

incluir órdenes para manipulación de datos adecuados al microdrive y otras interfaces.

El microdrive ZX y la interface 1 son eficaces accesorios para el sistema Spectrum. Las facilidades que ofrecen reproducen con mucha fidelidad la operación de un sistema de disco flexible convencional. Sin embargo, existen las limitaciones impuestas por la incierta fiabilidad a largo plazo de los cartuchos de cinta y la falta de una auténtica facilidad para archivos de acceso directo. Esto afecta la viabilidad del microdrive como sistema de almacenamiento para aplicaciones serias de gestión empresarial, porque la mayoría del software comercial requiere la localización, el almacenamiento y la recuperación de pequeñas cantidades de datos muchas veces durante su utilización. El advenimiento del ordenador de gestión Sinclair QL, que se suministra con dos microdrives incorporados, bien podría ser el origen de una ampliación de las capacidades del sistema.

Ian Mckinnell

Cinta flexible
Combinando el bajo precio y la simplicidad del almacenamiento en cinta con la velocidad del disco, el microdrive se ha llegado a conocer como una cinta flexible. Se conecta con el Spectrum a través de la interface 1.
Conectores de sonido para acceso a cassette y una puerta RS232 son también configuraciones de la interface 1

Órdenes del microdrive

Las órdenes que se utilizan con el microdrive son:

FORMAT CAT SAVE* VERIFY* LOAD*
MERGE* ERASE OPEN# PRINT# INPUT#
INKEY$# CLOSE# MOVE

En todos los casos, M selecciona el microdrive, N es el número del microdrive al que acceder (1-8) y S es el número de flujo asignado (4-15).

Format
Formatea la configuración de los datos en cinta, da nombre al cartucho y borra todos los datos grabados previamente. Se puede construir de la siguiente manera:

```
FORMAT "M";N;"NAME" o
FORMAT M$;N;C$
```

donde NAME es el nombre seleccionado para el cartucho (1-10 caracteres), M$ (ya sea M o m) es lo mismo que M y C$ (1-10 caracteres) es como NAME.

Cat
Esta orden carga a la pantalla o flujo especificado un catálogo de todos los archivos contenidos en cartucho en una unidad determinada. Se obtiene con:

```
CAT N o
CAT# S;N
```

El catálogo contiene el nombre del cartucho, hasta 50 nombres de archivo y el espacio libre en el cartucho expresado en Kbytes.

Save*
Crea archivos de programas que pueden ser programas, cadenas o datos con nombre, y se construye de una de las siguientes maneras:

1. SAVE* "M";N;"NOMARCHI"
2. SAVE* "M";N;"NOMARCHI"SCREEN$
3. SAVE* "M";N;"NOMARCHI"DATA A()
4. SAVE* "M";N;"NOMARCHI"LINE X

creando lo siguiente:

1. Un archivo
2. Un archivo que conste de SCREEN$
3. Un archivo que conste de datos A()
4. Un archivo que se ejecutará (RUN) desde la línea X al cargarlo (LOAD).

Verify*, Merge* y Erase*
Estas órdenes se construyen de la misma forma que 1. SAVE* anterior. VERIFY* compara el archivo "NOMARCHI" con el contenido corriente de la memoria para el usuario y genera un mensaje de error si son diferentes. MERGE* une "NOMARCHI" con el contenido corriente de la memoria para el usuario, y ERASE* borra "NOMARCHI".

Load*
LOAD* se puede construir de la misma forma que 1. SAVE* y 2. SAVE*. Al ejecutarla, LOAD* copia el contenido del archivo especificado en la memoria para el usuario.

Open#, Print#, INPUT#, Inkey$#, Close# y Move
Estas órdenes están relacionadas con la manipulación de archivos de datos. Estos se almacenan secuencialmente, pero se pueden manipular para que imiten algunas de las propiedades de los archivos de acceso directo permitiendo leer un archivo de datos entero y extrayendo los datos requeridos después de cargarlo. Del mismo modo, ese archivo se puede modificar y realmacenar. La manipulación de archivos de datos se organiza abriendo y cerrando (OPEN y CLOSE) flujos para establecer canales de datos haciendo referencia al número de flujo especificado. Por ejemplo:

```
OPEN#S;"M";N;"NOMARCHI"
```

conecta el flujo S con el archivo "NOMARCHI" en el microdrive N. Se puede, entonces, escribir en "NOMARCHI" utilizando PRINT#S y leerlo empleando INPUT#S o INKEY$#S. MOVE se puede utilizar para transferir archivos de datos dentro de un cartucho, de un microdrive a otro, o a cualquier dispositivo al cual se pueda acceder a través de un número de flujo. Cuando los canales de flujo ya no se necesitan, se desconectan mediante CLOSE#S.

Wafer del microdrive
La cinta del wafer es una videocinta de 2 mm, utilizada en virtud de su fortaleza y su elevada densidad de almacenamiento. La cinta forma un bucle de 8 metros y está revestida, de modo que se desliza con facilidad desde el carrete de cinta central a la cinta rebobinada. Al estar en su lugar un apéndice protector de escritura permite leer datos del wafer o escribirlos en él; si se quita el apéndice, el wafer sólo se puede leer, protegiendo, por tanto, la información almacenada

Páginas de memoria

El ordenador, para localizar de manera expedita cualquier byte, divide la memoria en "páginas", como si se tratara de un libro

En el primer capítulo de esta serie sobre el lenguaje máquina vimos por analogía la forma en que los ordenadores almacenan la información con la corriente eléctrica. Utilizamos el ejemplo de una fábrica en la que cada trabajador tenía un modelo propio de interruptor, que encendía cuatro bombillas en el despacho del director, lo que permitía identificar quién era el empleado que había llegado al trabajo.

En nuestro ejemplo descubrimos que utilizando cuatro interruptores y bombillas podíamos representar los números entre 0 y 15. En otras palabras, sólo había 16 modelos posibles. Pero, en cambio, si hubiéramos empleado ocho interruptores y ocho bombillas, habríamos dispuesto de 256 modelos exclusivos y, por consiguiente, hubiésemos podido contar desde 0 hasta 255 (variaciones de 2 elementos, OFF y ON, en grupos de 8 cada vez = $= 2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$).

Pues bien, un ordenador personal tiene la memoria dispuesta en bancos individuales de ocho interruptores y cada uno de estos bancos se denomina *byte*. En general, la CPU manipula la información de un byte por vez, lo que de hecho significa que sólo puede sumar, comparar y almacenar números entre 0 y 255. Podría parecer que esto limita sus capacidades aritméticas, pero no es así. Piense, por ejemplo, cómo hace una suma cualquiera, 63951 + 48770 = ?, y verá que en realidad manipula los dígitos uno a uno. Del mismo modo, la CPU puede efectuar sumas de números muy grandes utilizando un byte cada vez.

Debido a que posee ocho interruptores, un byte es un lugar donde se puede almacenar un número binario de ocho dígitos. Cada una de estas posiciones de dígitos binarios se denomina *bit*. El bit de un byte está o encendido (ON) o apagado (OFF) y un dígito binario o es un 1 o es un 0.

Suele ser importante individualizar cada uno de los bits de un byte, de modo que por convenio los bits se numeran de 0 a 7 de derecha a izquierda del byte. Si un byte contiene el número binario 00000001, decimos entonces que el bit 0 es 1, o que el bit 0 está ON, o que el bit 0 está ACTIVADO; todos los otros bits son 0, o están OFF, o están DESACTIVADOS. De modo que, por cuanto concierne al número binario 01001000: el bit 3 está ACTIVADO, al igual que el bit 6, el bit 4 está OFF, el bit 7 es 0, el bit 0 está DESACTIVADO, etc. En un byte, también se suele decir que el bit 0 es el bit menos significativo (LSB: *Least Significant Bit*) y el bit 7 el bit más significativo (MSB: *Most Significant Bit*).

Podemos, pues, concebir la memoria del ordenador como una larga tira de papel cuadriculado, de ocho cuadrados de ancho y miles de cuadrados de largo: cada fila de ocho cuadrados es un byte, y esos mismos cuadrados son los bits de ese byte. La memoria no sirve de nada si no se pueden posicionar datos en ella, de modo que cada uno de los bytes posee una etiqueta de identificación que se denomina *dirección*; la dirección de un byte no está escrita en ningún lugar del papel (o del byte), sino que sencillamente es el número del byte en la memoria, contando a partir del principio de la misma. El primer byte, por lo tanto, recibe la dirección 0, el byte siguiente la dirección 1, el siguiente la dirección 2, y así sucesivamente. Si se desea escribir algo en el byte 43, contaremos los bytes, empezando por la parte inferior de la memoria (por el byte 0), hasta llegar al byte 43.

Una vez alcanzado, no habrá nada que identifique a ese byte como el byte 43, excepto su posición: se ha ido contando hacia arriba a partir del byte 0 y así llegamos al 43, de modo que ése ha de ser el byte 43. Los bytes de la memoria son en realidad minúsculos bancos de dispositivos de ocho transistores grabados en los chips del interior de la máquina y son idénticos entre sí en todo, excepto en su posición física.

Sin embargo, este método tiene un inconveniente. Este sistema de direccionamiento de memoria sería apropiado si sólo hubiera unos pocos centenares de bytes. La CPU puede contar de 0 a 100 en fracciones de milisegundo; pero los ordenadores poseen millares de bytes, y contar de 0 a 20 000 ocupa un tiempo apreciable, aun para un microprocesador. El ordenador tiene una forma de superar este problema y consiste en dividir la memoria en páginas, al igual que sucede con los libros.

Si seguimos pensando en la memoria de un ordenador como una tira de papel cuadriculado de millares de cuadrados de longitud y ocho cuadrados de anchura, podemos imaginariamente cortar esa tira cada 100 bytes (o sea, cortar por la línea que divide el byte 99 y el byte 100, cortar de nuevo por la que separa el byte 199 y el byte 200, por el byte 299 y el byte 300, y así sucesivamente). Ahora cada una de las tiras de papel entre los cortes es una página de 100 bytes. La página 0 empieza en el byte 0 y sigue hasta el byte 99; la página 1 empieza en el byte 100 y sigue hasta el byte 199; la página 2 comprende desde el byte 200 hasta el byte 299, etc. Según esto, para hallar ahora cualquier byte, supongamos el byte 3 518, no necesitamos contar 3 518 bytes desde el principio de la memoria, porque la dirección nos indica que este byte debe estar en la página 35. En consecuencia, sólo necesitamos contar 35 páginas desde la parte inferior de la memoria y después contar los bytes desde la parte inferior de esa página hasta llegar al byte 18 de la misma, que debe ser el byte 3 518. Pruebe, para verlo mejor, con una tira de papel cuadriculado.

Este sistema de memoria con paginación es conveniente porque podemos tomar la dirección de cualquier byte y dividirla en dos partes: los dígitos

desde la columna de las centenas hacia la izquierda corresponden al número de página del byte, y los dígitos desde la columna de las decenas hacia la derecha representa el número de bytes contados desde la parte inferior de esa página. En el ejemplo anterior separamos la dirección 3 518 en dos números: el número de página 35 y el número de byte 18 de esa página. Decimos que 18 es un *offset* (desplazamiento) o un *page offset* (desplazamiento de página), porque es el número en razón del cual uno ha de desplazar (o incrementar) la dirección del byte inferior hasta el byte en cuestión.

El ordenador, no obstante, no cuenta en decimal, como contamos nosotros, sino que cuenta en binario. El sistema de paginación consiste en ser capaz de hallar la página y el desplazamiento sencillamente mediante la inspección de la dirección del byte. Ya vimos que el decimal 99 se representa 01100011 en binario, y el decimal 100 es el binario 01100100; el decimal 199 es 11000111, y el decimal 200 es el binario 11001000. A partir de estos ejemplos podemos apreciar que no existe una forma sencilla de saber qué página es, con sólo observar los números binarios, como hacemos tan fácilmente con los equivalentes decimales.

Hemos elegido 100 como tamaño de la página porque se trata de un número significativo en el sistema decimal (es una potencia de 10). Sin embargo, si hemos de contar en binario, entonces debemos elegir un tamaño de página que sea adecuado para *ese* sistema. El tamaño de página que utilizan nuestros ordenadores es 256, de modo que la página 0 empieza con el byte 0 y continúa hasta el byte 255; la página 1 empieza con el byte 256 y continúa hasta el byte 511, etc. Para ver la conveniencia de esta nueva numeración de las páginas debemos escribir las direcciones en binario:

```
Página 0: byte 00000000  – byte 11111111
Página 1: byte 100000000 – byte 111111111
```

Como apreciará, podemos contar en binario de 0 a 255 en números de ocho dígitos; el número siguiente (256) requiere nueve bits, y con nueve bits podemos contar hasta 511. El número siguiente (512) requiere diez bits, y con diez bits podemos contar hasta 1023; y así sucesivamente. Ahora vemos que si el tamaño de la página es 256 y contamos en binario, un desplazamiento sobre la página sólo puede ser indicado por los ocho bits situados a la derecha, pues el número de página viene dado por los bits situados a partir del bit 8 hacia la izquierda.

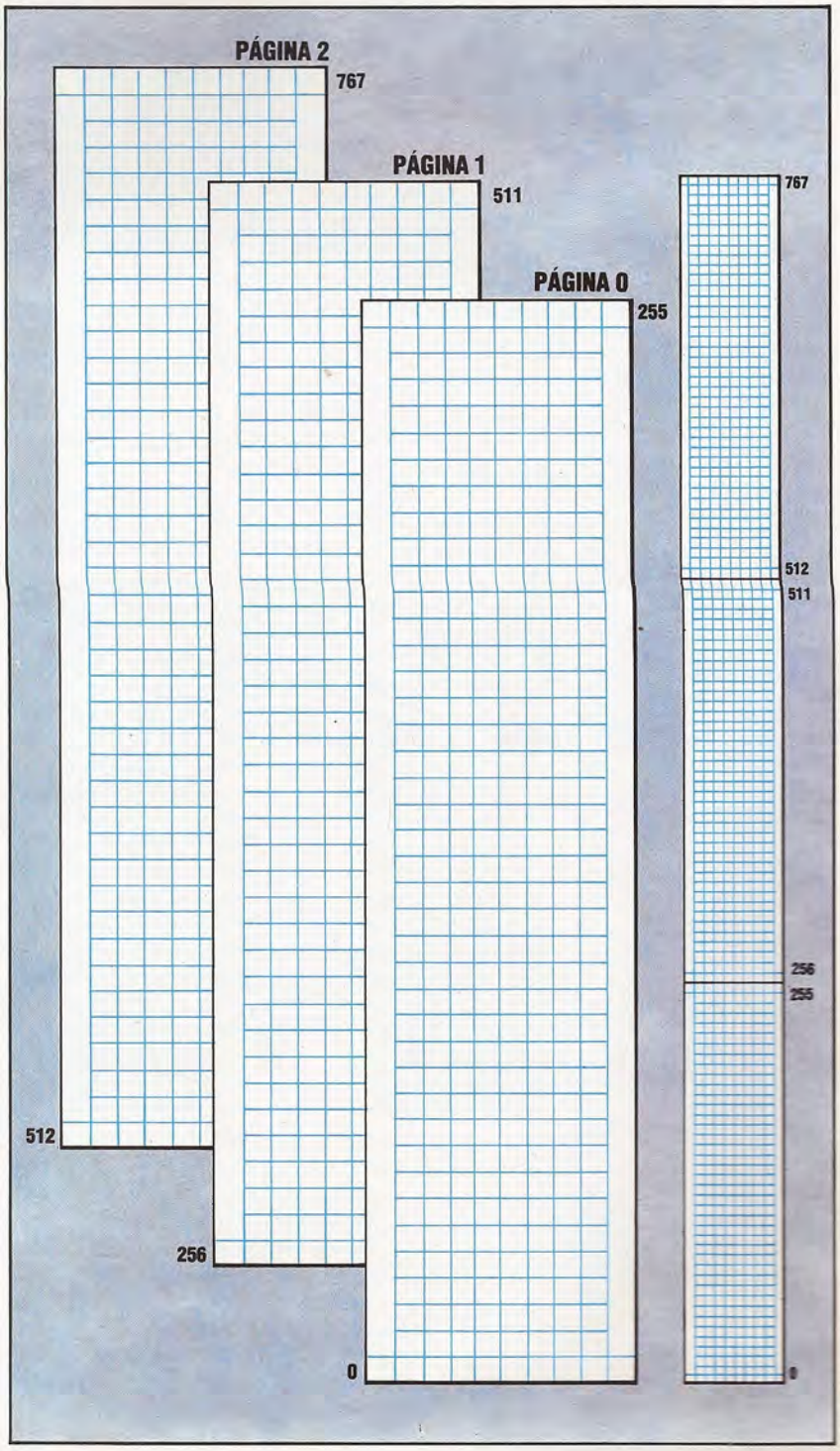
Esto puede resultar sorprendente dado que antes hemos afirmado que la CPU sólo puede manipular bytes individuales, y un byte sólo contiene ocho bits. Por consiguiente, puede que se esté preguntando qué sentido tiene hablar de números de nueve o diez bits. La respuesta es que todas las direcciones de la memoria se tratan como números de dos bytes y la CPU se ocupa de ellas a un byte por vez. Si volvemos a escribir los límites de páginas en forma de números de dos bytes veremos cómo este sistema nos resulta más claro:

```
Página  0 empieza en  00000000 00000000
          termina en  00000000 11111111
Página  1 empieza en  00000001 00000000
          termina en  00000001 11111111
Página 10 empieza en  00000010 00000000
          termina en  00000010 11111111
Página 11 empieza en  00000011 00000000
          termina en  00000011 11111111
```

y así sucesivamente.

Ahora podemos apreciar que cuando la CPU recoge información de un byte de la memoria, o cuando coloca información en él, dicho byte se puede identificar mediante una dirección de dos bytes. El primero de los dos bytes, o el situado más hacia la izquierda, da el número de página, mientras que el segundo byte, o el situado más a la derecha, da el desplazamiento.

Al final de esta sección proporcionamos programas que convierten decimales a binarios y a hexadecimales.



Direccionamiento por páginas
El direccionamiento por páginas divide a la memoria en bloques o páginas imaginarias de 256 bytes. Todas las direcciones se expresan así como números de dos bytes: un byte proporciona el número de página y el otro proporciona el desplazamiento desde el principio de esa página

Kevin Jones

Procesador de números

Los tres programas que presentamos aquí, para el BBC Micro, Spectrum y los Commodore, reciben números decimales y devuelven sus equivalentes binarios y hexadecimales

Commodore 64

```
10 REM********COMMODORE**********
40 S$="=              ":X$=
   "0123456789ABCDEF"
50 REM  S$ CONTIENE 9 ESPACIOS
60 PRINT CHR$(147)     :REM LIMPIAR
   PANTALLA
70 PRINT "VISUALIZAR NUMEROS
   DECIMALES"
80 PRINT "Y SUS EQUIVALENTES
   BINARIOS"
90 PRINT:PRINT "    *******DIGITE
   EL 0 PARA SALIR********":PRINT
100 FOR K=1 TO 1
110 FOR L=1 TO 1
120 INPUT"DIGITE CUALQUIER NUMERO
   ENTERO POSITIVO";A$
130 NU=VAL(A$)
140 IF NU=0 THEN PRINT"SALIDA
   PROGRAMA":STOP
150 IF INT(NU)<>ABS(NU) THEN L=0
160 IF NU>65535 THEN PRINT NU;"ES
   DEMASIADO GRANDE":L=0
170 NEXT L
200 NM=NU:H$="":GOSUB 2000
210 PRINT NU;TAB(5);N$;
220 IF RIGTH$(A$,1)="+" THEN
   GOSUB 4000
230 PRINT H$:PRINT:PRINT
240 K=0:NEXT K
250 END
300 END
1000 REM******S/R BYTE
   BINARIO******
1010 B$=""
1020 FOR D=8 TO 1 STEP-1
1030 N1=INT(N/2)
1040 R=N-2*N1
1050 B$=MID$(STR$(R),2)+B$
1060 N=N1
1070 NEXT D
1080 RETURN
2000 REM****S/R CONVERSION
   BINARIO***
2010 IF NM<256 THEN N=NM: GOSUB
   1000:N$=S$+B$:RETURN
2020 HI=INT(NM/256):LO=NM-256*HI
2030 N=HI:GOSUB 1000:N$="= "+B$
2040 N=LO:GOSUB 1000:N$=N$+" "+B$
2050 RETURN
3000 REM****S/R BYTE
   HEXA**********
3010 HB=INT(N/16):LB=N-HB*16
3020 B$=MID$(X$,HB+1,1)+MID$
   (X$,LB+1,1)
3030 RETURN
4000 REM****S/R CONVERSION
   HEXA******
4010 IF NM<256 THEN N=NM:GOSUB
   3000:H$="= "+B$: RETURN
4020 HI=INT(NM/256):LO=NM-256*HI
4030 N=HI:GOSUB 3000:H$="= "+B$
4040 N=LO:GOSUB 3000:H$=H$+" "+B$
4050 RETURN
```

BBC Micro

Copiar el listado del Commodore con las siguientes modificaciones:

```
60 CLS:@%=5
210 PRINT TAB(0);NU;TAB(5);N$;
1050 B$=STR$(R)+B$
```

Este programa no hace uso de las facilidades para representación de números del BBC para mantener la compatibilidad de formato con las otras máquinas: quizá usted sea capaz de volver a escribirlo en una versión más corta.

Spectrum

```
10 REM********SPECTRUM**********
40 LET S$="=              ":LET X$=
   "0123456789ABCDEF"
50 REM   S$ CONTIENE 9 ESPACIOS
60 CLS
70 PRINT "VISUALIZAR NUMEROS
   DECIMALES"
80 PRINT "Y SUS EQUIVALENTES
   BINARIOS"
90 PRINT:PRINT "********DIGITE EL
   0 PARA SALIR*********":PRINT
100 FOR K=1 TO 1
110 FOR L=1 TO 1
120 INPUT"DIGITE CUALQUIER NUMERO
   ENTERO POSITIVO";A$
130 LET NU=VAL(A$)
140 IF NU=0 THEN PRINT "SALIDA
   PROGRAMA":STOP
150 IF INT(NU)<>ABS(NU) THEN LET L=0
160 IF NU>65535 THEN PRINT NU;"ES
   DEMASIADO GRANDE":LET L=0
170 NEXT L
200 LET NM=NU:LET H$="":GOSUB 2000
210 PRINT NU;TAB(5);N$;
220 IF A$(LEN A$)="+" THEN GOSUB
   4000
230 PRINT H$:PRINT:PRINT
240 LET K=0:NEXT K
300 STOP
1000 REM**S/R BYTE BINARIO**
1010 LET B$=""
1020 FOR D=8 TO 1 STEP-1
1030 LET N1=INT(N/2)
1040 LET R=N-2*N1
1050 LET B$=STR$(R)+B$
1060 LET N=N1
1070 NEXT D
1080 RETURN
2000 REM**S/R CONVERSION BINARIO**
2010 IF NM<256 THEN LET N=NM:GOSUB
   1000:LET N$=S$+B$:RETURN
2020 LET HI=INT(NM/256):LET
   LO=NM-256*HI
2030 LET N=HI:GOSUB 1000:LET
   N$="= "+B$
2040 LET N=LO:GOSUB 1000:LET
   N$=N$+" "+B$
2050 RETURN
3000 REM****S/R BYTE HEXA******
3010 LET HB=INT(N/16):LET
   LB=N-HB*16
3020 LET B$=X$(HB+1)+X$(LB+1)
3030 RETURN
4000 REM***S/R CONVERSION HEXA****
4010 IF NM<256 THEN LET N=NM:GOSUB
   3000:LET H$="=  "+B$:RETURN
4020 LET HI=INT(NM/256):LET
   LO=NM-256*HI
4030 LET N=HI:GOSUB 3000:LET
   H$="= "+B$
4040 LET N=LO:GOSUB 3000:LET
   H$=H$+" "+B$
4050 RETURN
```

Si se da entrada a un numero con un "+" al final, por ejemplo: 6435+, entonces, además de su representación decimal y binaria, obtendrá también su representación hexadecimal

La reina del juego

Al igual que otras grandes firmas, Atari surgió al cristalizar una idea: Nolan Bushnell, al inventar el juego llamado "Pong", no imaginó que aquello era apenas el principio...

El sencillo método de Nolan Bushnell de poner el control de lo que aparecía en la pantalla en las manos de quien lo estaba contemplando, habría de transformar el concepto popular del ocio y cautivar a millones de jóvenes.

Bushnell y sus dos socios, Ted Dabney y Larry Bryan, aportaron cada uno 100 libras esterlinas para lanzar el Pong. El juego hizo su primera aparición en Sunnyvale, California, en 1972, y pronto hubo pruebas de que el invento iba a ser algo sensacional y rentable. La hegemonía de Atari sobre el mercado de videojuegos comenzó con una astuta decisión tomada al poco tiempo: la de comprar los derechos a Bushnell.

Atari se mantuvo a la cabeza del mercado durante la mayor parte de la década de los setenta, hasta que el gusto del público pasó de las máquinas de juegos recreativos a los micros personales. Comercializar juegos es como comercializar discos: uno debe descubrir sus potenciales estrellas "pop" y promocionarlas. Es bastante normal, en consecuencia, que Atari sea propiedad de la multinacional Warner Comunications International, conocida entre nosotros por sus intereses en las industrias del cine y discográfica. Y aunque los negocios de Atari en máquinas recreativas tragaperras reportaron abundantes beneficios a finales de los setenta, en 1983 los ingresos disminuyeron un 25 %, lo que ocasionó cuantiosas pérdidas a la empresa madre.

El juego *Space invaders* (Invasores del espacio) de Taito, que Atari ha comercializado sabiamente, es el más conocido de todos los juegos por ordenador. Se convirtió en un fenómeno social y dio origen a todo un universo de juegos de persecución intergaláctica. A finales de los años setenta Atari estaba en el centro del *boom* de los juegos recreativos. La empresa seguía produciendo éxitos uno tras otro: *Asteroides, Battlezone, Ciempiés, Lunar lander, Comando de misiles* y *The tempest*.

Pero el *boom* recreativo se extinguió con la misma vertiginosa rapidez con que había comenzado. Los clientes se volcaron a los ordenadores personales porque éstos ofrecían dos principales ventajas. Con ellos uno podía jugar a los juegos recreativos sin acudir a los salones de juegos y sin tener que apilar monedas, al mismo tiempo, uno disponía de una máquina electrónica muy flexible.

En un primer momento Atari respondió a este cambio en la demanda del mercado convirtiendo su mejor software recreativo en juegos por ordenador. Éstos utilizaban cartuchos en estado sólido que se enchufaban en la parte posterior de una unidad de ordenador personal y o bien se sumaban a la propia ROM del ordenador o la sustituían. Aunque ésta resultó ser una forma efectiva de adquirir un juego por ordenador, puesto que no requería que el jugador cargara el programa del juego en la memoria desde cassette o disco, los componentes en estado sólido hacían que los cartuchos resultaran muy costosos. Y como estos cartuchos no eran reprogramables (los programas estaban grabados físicamente en los circuitos), a menudo la empresa se quedaba con montañas de material electrónico de desecho proveniente de aquellos juegos que no conseguían cobrar la popularidad suficiente.

Los éxitos disminuyen

La línea comercial seguida por Atari pronto evidenció signos de debilitamiento. La empresa basó la cifra de futuras ventas para algunos de sus cartuchos de juegos en las de PacMan, el juego que obtuvo un éxito tan fenomenal, y al fin hubo de pagar el precio por este error de cálculo. Se hizo un inventario de los cartuchos invendibles, y 14 camiones cargaron con ellos acabando por consignarlos en un gran pozo del desierto de Nevada.

Atari tampoco se resarció con una configuración exclusiva de los productos para juegos por ordenador: convertir el código de ordenador en una entidad física no lo es todo para distribuirlo eficazmente. Se puede transmitir por teléfono o por cable, o emitir por radio o por televisión. Cada vez más están saliendo al mercado nuevas técnicas y productos que permiten estas formas de transmisión. En 1983, por ejemplo, la Romox Corporation de Estados Unidos presentó una máquina a la que llamaron *Romox Programming Terminal*. Se trataba de una máquina de discos rígidos de 15 Megabytes que se podía comunicar, a través de las líneas telefónicas, con una base de datos de software para

Cortesía de Atari

Nolan Bushnell
Los éxitos de Atari se basaron en los esfuerzos y la dedicación de un hombre: Nolan Bushnell. Cuando Bushnell creó el Pong (el primer juego para ordenadores), en 1971, sin duda ignoraba el contenido de esa caja de Pandora que estaba abriendo

Sistema exclusivo para juegos
Por muy poco dinero, el Video Computer System de Atari viene completo con dos palancas de mando, adaptador de corriente y un cartucho del juego PacMan. Al ser "exclusivo" para juegos, el VCS no se puede utilizar como ordenador para aplicaciones generales, y todos los juegos se venden en cartucho

juegos. Además, también era un dispositivo EPROM con ranuras para aceptar los conectores de los principales cartuchos. La importancia de esta máquina reside en el hecho de que ahora se puede acudir a un comerciante de la zona, disponer de una lectura instantánea de los "Veinte mejores", los veinte juegos más vendidos, y seleccionar uno de ellos para llevarlo de inmediato en un cartucho Romox en blanco.

Una alternativa a este método de distribución fue Gameline, el sistema de "paga lo que juegues" que creó Bill van Meister en Estados Unidos. Gameline comercializa un modem enchufable para máquinas Atari VCS que acopla el ordenador personal al sistema telefónico. Originalmente estos juegos cuestan un dólar por 45 minutos de juego.

Dos de las redes más grandes de informática personal de Estados Unidos, Compuserve y The Source, ofrecen software para juegos como parte de su servicio regular para proveedores que, de lo contrario, enchufan ordenadores personales en una base de datos remota a través de un modem y de la red telefónica. Coleco, una empresa fabricante de videojuegos domésticos, se ha unido con la AT&T (American Telephone and Telegraph) para proporcionar un servicio de entretenimiento interactivo. Atari entiende que éste es el camino a seguir y se ha unido a Activision para enviar programas de juegos a través de la red telefónica en su plan secreto *Ataritel*. Siempre que Atari permanezca en sociedad con Warner Communications, la cosa funcionará; pero si Atari se vendiera entonces perdería acceso a Warner Amex Cable Communications, de la cual depende el plan.

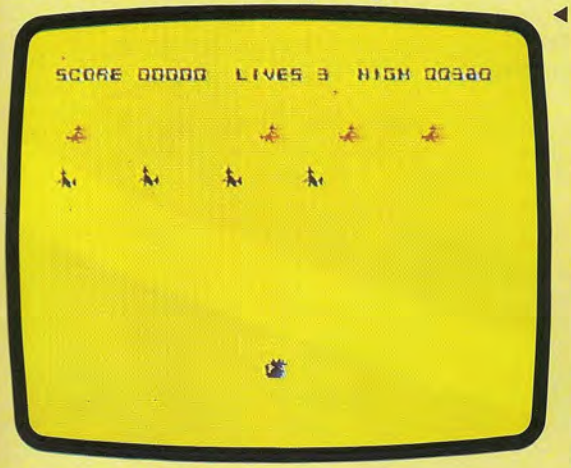
"Major Havoc"
Durante la mayor parte de la década de los setenta, Atari funcionó principalmente gracias a los beneficios que generaban las máquinas recreativas como esta que vemos en la fotografía, uno de los muchos juegos de "captura". El advenimiento del ordenador personal exigió una estrategia comercial completamente nueva

Atari ha tenido sus problemas, originados parcialmente en el pasado por la rivalidad entre la decadente sección videojuegos y el creciente departamento ordenadores personales de la empresa, y los ha resuelto ahora mediante fusión. Pero aún posee una buena gama de máquinas personales que durante mucho tiempo han sido la vanguardia de los gráficos y el software fácil de utilizar para ordenadores personales. Las máquinas personales más nuevas son similares, en cuanto a diseño, a las antiguas, que fueron notablemente avanzadas para su época. Estas máquinas configuran tres chips a la medida: Pokey, Antic y GTIA, que controlan puertas de entrada-salida, gráficos y color.

Todas las máquinas se basan en el procesador 6502 y existe en la actualidad una útil variedad de programas de utilidades. Entre éstos se incluyen: VisiCalc, Atariwriter (un paquete para tratamiento de textos) y un programa para administración doméstica. La softcard Z80 es ahora una realidad, y esto hace que los ordenadores Atari sean aptos para equiparlos con el Personal CP/M de Digital Research. La empresa también está prodigando sus atenciones al software. Su sección de Gran Bretaña ha nombrado a un mediador de software para solventar sus problemas relacionados con el software, para comercializar su mejor software convenientemente adaptado a otros micros (en particular las máquinas Commodore) y, lo más significativo, para observar de cerca el escenario británico y detectar a los jóvenes programadores de juegos. Al cabo de un par de años en los cuales los hados no le fueron muy propicios, Atari posee el potencial suficiente para recobrar sus días de esplendor.

Aventura y fantasía

El diseñador de juegos



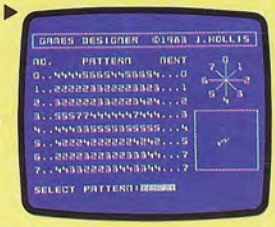
Formato estándar
Los sprites aparecen sobre un trasfondo desnudo y luego descienden lenta y ordenadamente

Efecto final
Esta vista muestra las modificaciones a los sprites originales y el color del fondo. El único elemento que no se ha alterado es el gato

Configuración de sprites
Seleccione en el diagrama la coordenada apropiada que desea rellenar (o borrar). En la parte inferior de la pantalla puede ver la forma y el color reales del sprite

Menú de configuración
Este menú le permite seleccionar la dirección de movimiento de su nave o láser base, los colores de fondo y primer plano, la apariencia de los extraterrestres y los efectos de sonido para todos los elementos

Menú de movimiento
Éste le permite determinar el enfoque de ataque. En el extremo superior derecho de la pantalla verá un diagrama de direcciones numeradas y abajo a la derecha hay una visualización de patrones donde puede supervisar los efectos



Ian McKinnell

Existen paquetes para juegos que permiten al usuario formular las reglas del juego que ha diseñado y dar vuelo a su imaginación

La mayoría de los juegos para ordenadores personales entran en dos categorías: juegos de aventuras con participación, con o sin una representación gráfica del escenario en la pantalla, como *El Hobbit*, y los "simuladores de fantasía" totalmente basados en pantalla, como *Space invaders* o *Asteroids*.

Incluso un análisis superficial de los dos tipos genéricos revela la causa de su similitud. Tomando en primer lugar los simuladores de fantasía, el juego estilo *Space invaders* exige dos requisitos previos: una base de disparo y un blanco. De modo que si pudiéramos construir versiones abstractas de estas dos cosas y permitir que el diseñador de juegos decida de forma independiente acerca de la posición del defensor, la potencia de fuego y la frecuencia e intensidad de las ráfagas de ataque, entonces sería factible producir diversos juegos, cada uno de los cuales se diferenciaría en alguna sutileza respecto al siguiente, simplemente variando los parámetros. Un análisis de los numerosos juegos, cuyos nombres suenan parecidos, producidos en una "generación", revela que los productores profesionales de software han aplicado este criterio.

Un ejemplo excelente de un paquete de juegos que le permite al usuario hacer exactamente eso es *Games designer* (Diseñador de juegos), de John Hollis (de Software Studios/Quicksilva, para el Spectrum de 48 K). El *Games designer* ofrece ocho juegos esbozados. Después de seleccionar uno de ellos, usted puede alterar todos los parámetros básicos citados, pero no construir un juego nuevo. El paquete es activado por menú.

Después de escoger un juego, que será el bosquejo sobre el cual se efectuarán las modificaciones, se le ofrecen unas opciones del tipo y la envergadura de dicha modificación, empezando por la

forma y el color de los sprites utilizados para representar a los protagonistas. Aun cuando se presenta un menú de los sprites existentes, esto no es una limitación sino más bien una comodidad, porque el usuario puede comenzar por un diseño predefinido y modificar cualquier bit dentro de la matriz de 12 × 12, u optar por empezar con una "página en blanco" y originar una figura completamente nueva. El formato de 12 × 12 (dos caracteres por dos caracteres) da como resultado un sprite perfectamente utilizable y característico.

La definición de sprites es activada por menú, visualizándose las opciones de las órdenes disponibles a la izquierda de la pantalla, y una imagen ampliada del sprite en cuestión a la derecha. Un agregado muy útil es una segunda imagen del sprite, que se visualiza al pie del texto, en el color y las dimensiones reales.

El siguiente paso consiste en definir de qué modo se desplazarán estos sprites alrededor del campo para juegos. Esta sección, denominada *configuración*, le permite al jugador mayores posibilidades que simplemente dirigir el movimiento de distintivos individuales. Desde este momento, por el orden en el cual se visualizan las opciones del menú principal, se lo invita a seleccionar un patrón de movimiento para los distintivos de pantalla. Para hacer más complicado el movimiento, en esta etapa se pueden unir entre sí dos o más patrones.

Ahora el usuario debe decidir la frecuencia de las ondas de ataque e insertar efectos especiales, tanto visuales como sonoros.

Habiendo creado un juego, sólo falta guardarlo en cassette para que se pueda volver a jugar a él en el futuro. Y aquí el paquete fracasa estrepitosamente, desde el punto de vista del usuario, ya que no almacena un juego en forma representable, sino sólo como parámetros de entrada para el paquete *Games designer* propiamente dicho, haciendo que el ejercicio resulte completamente inútil para quienes desearían desarrollar un juego para después venderlo.

Una pluma creativa

En contraposición directa con el *Games designer*, *The quill* (La pluma de ave) está dirigido a quienes desean crear juegos de aventuras e intentar comercializarlos; en el manual aparece incluso una sección titulada *Para vender su aventura*, que proporciona valiosas indicaciones acerca de los pasos que habría que dar para la verificación y depuración. Una de las principales sugerencias consiste en hacer que la mayor cantidad posible de personas pruebe el juego con sentido crítico. Hay que destacar que los autores sólo exigen que "se mencione en algún lugar [del juego] que se lo escribió con *The quill*", una rara muestra de altruismo.

Aunque se lo presenta como un generador de juegos del tipo *Dungeons and dragons* (Calabozos y dragones), *The quill* utiliza técnicas más parecidas a las que emplean los paquetes comerciales para administración de bases de datos. Se divide en tres partes principales: la base de datos propiamente dicha; un editor de base de datos, que permite establecer los parámetros, y un intérprete de base de datos, que ejecuta el juego de forma interactiva.

Así como uno podría catalogar *Space invaders* y todos los demás juegos similares como juegos protagonistas-posicionamiento, del mismo modo la base de los juegos de aventuras es el "laberinto conceptual": un laberinto que existe en otras dimensiones aparte de las puramente espaciales. De este modo, además de buscar el camino que conduce a la salida, al jugador se le presenta una lista de objetos, cada uno de los cuales sólo es útil en una circunstancia específica; si usted nunca entra en una habitación donde hay un portalámparas vacío, por ejemplo, no tiene ningún sentido que lleve consigo una bombilla.

Para dotar de mayor interés al juego, se suele limitar el número de objetos que el jugador puede llevar consigo de un lugar a otro, pero no hay impedimentos para que el diseñador utilice este límite artificial para hacer ir y venir al jugador acumulando los objetos que necesitará para realizar las tareas que se le asignan a medida que avanza el juego.

Definiendo el objeto
The quill ofrece a los usuarios la posibilidad de crear juegos de aventuras. De estilo similar a algunos paquetes para administración de bases de datos, exige que se definan las posiciones y los objetos que se incluyen en la aventura. Después, las técnicas de programación orientadas hacia un objeto permiten llamar a estos atributos asignados y visualizarlos, o utilizarlos como parámetros cada vez que el objeto o posición aparezca en el juego

Ian McKinnell

La primera etapa en el diseño de un juego de aventuras consiste en crear el guión y el escenario en que se desarrollará. El manual de *The quill* comienza esbozando un juego sencillo que enfrenta al jugador con una opción de diez objetos en un entorno simple que abarca seis compartimientos. Este juego no está incorporado en el programa, lo que obliga al usuario a dar entrada a todos los parámetros bosquejados en el manual antes de poder jugar a él, tal como tendría que hacer si usted hubiera creado su propio juego.

The quill pone un límite al número de parámetros que se pueden utilizar en el juego, pero éste es tan elevado (252 posiciones y 210 objetos) que es muy poco probable que se quede sin opciones.

El paquete puede producir juegos de aventuras que compitan con la mayoría de los juegos disponibles a nivel comercial (de hecho, cierta cantidad de ellos se han escrito a partir de él), pero tiene sus limitaciones. En primer lugar, no permite la creación de gráficos en pantalla como los que tan bien se emplean en *El hobbit*, por ejemplo; tampoco permite ninguna clase de interacción con personajes, de modo que no tiene ningún sentido intentar crearlos. Además de ofrecerle al usuario de juegos de aventuras un método simplificado para componer sus propios juegos, su mayor mérito reside en la disciplina que requiere su utilización.

Nacido para triunfar

Desde su aparición, en 1982, se ha apoderado del mercado de microprocesadores de 16 y 32 bits

El 68000 guarda un fuerte parecido con el MC6800, un microprocesador Motorola algo más viejo que todavía se sigue utilizando mucho, en especial para periféricos tales como controladores inteligentes. Esto significa que el 68000, mucho más capaz, es fácil de conectar en interface y tiene el respaldo de una amplia variedad de hardware prefabricado. Éste incluye tableros de E/S con chips PIA (*Peripheral Interface Adaptor*: adaptador para interface de periféricos) 6821, unidades de representación visual (VDU) con CRTC (*Cathode Ray Tube Controllers*: controladores de tubo de rayos catódicos) 6845, relojes que utilizan los sincronizadores programables 6840 y controladores de disco.

Otra configuración del 68000 hace que les resulte atractivo a los diseñadores de ordenadores: la anchura de sus buses de datos y direcciones. Éstos están completamente separados el uno del otro, y cada bit posee su propia patilla, a diferencia del 8086, 8088 y Z8000, en los cuales las patillas están multiplexadas entre sí: los dos buses comparten un juego de patillas y las señales se intercalan y se decodifican en su punto de destino.

El procesador, por consiguiente, puede operar tan rápidamente como se lo permita el resto del sistema, y con los chips de RAM más recientes de 50 o 90 nanosegundos (10^{-9} segundos), esto significa una reducción, o incluso la desaparición, de los estados de espera. El procesador más veloz de la serie 68000 es el 68000L12, muchos de cuyos componentes se pueden hacer funcionar a 14 Megahertzios.

Sinclair Research ha utilizado el sucesor del 68000, el 68008, para el QL. Internamente es muy similar a los otros de su misma serie, pero, para hacerlo más compatible con los sistemas de ocho bits existentes, posee un bus de datos de ocho bits en lugar del bus de 16 bits de anchura total. Dado que necesita menos patillas, viene en un paquete normal en número de 40.

Pronto la Motorola va a producir un microprocesador aún más potente. El 68020 es un microprocesador de 32 bits que requiere un paquete de 96 patillas, cuya forma y estilo aún están por definir. El 68881, un procesador especializado en matemática de coma flotante con ocho registros (cada uno de 80 bits de ancho) y que aumentará notablemente la cantidad de datos "reales" que se puedan manipular, está también en fase de planificación.

Otros chips de la serie 68000 proporcionan funciones de E/S similares a las que ofrecían chips más antiguos, aunque muy mejoradas. Pero desde el punto de vista del programador, el 68000 posee muchas ventajas en relación a la mayoría de los otros procesadores de 16 bits, debido a la simetría de sus registros de direcciones y de datos y al rico juego de instrucciones.

Aunque tampoco es perfecto. En primer lugar, se hace una distinción entre los registros de direcciones y los de datos, a pesar de que son del mismo tamaño (32 bits) y, en la mayoría de los casos, se opera sobre ellos de la misma manera mediante las mismas instrucciones. A consecuencia de ello, a menudo es necesario desplazar datos desde un registro de direcciones a un registro de datos, manipularlos y luego devolverlos al registro de direcciones. Habría sido más sencillo si Motorola hubiera hecho posible utilizar cualquier registro tanto para datos como para direcciones.

En segundo lugar, hay redundancias en el juego de instrucciones, pero como éstas son producto de lo que podría denominarse "cruce de modalidad de direccionamiento", no tienen mayores consecuencias. En realidad, este fenómeno se produce porque las diversas modalidades de direccionamiento son tan distintas que en algunas ocasiones una puede significar exactamente lo mismo que otra, a pesar de haber llegado a ella mediante instrucciones diferentes.

No obstante, en general la serie Motorola 68000 proporciona unas CPU amplias, veloces y eficaces, que se están utilizando cada vez más. En 1983 fueron empleadas en el Lisa y el Mackintosh de Apple, en el QL de Sinclair y en muchas máquinas de gestión para usuarios múltiples de menor proyección en el mercado. Al proporcionar configuraciones que hace sólo un par de años hubieran costado un buen número de billetes, y al estar disponibles a un precio razonable, parecen estar llamadas a hacerse populares entre la nueva generación de máquinas, tal como lo son en la actualidad el Z80 y el 6502.

Las series ganadoras

6502
Desarrollado por MOS Technology, el microprocesador 6502 habría de convertirse, con el Z80 de Zilog, en el soporte de la industria de microordenadores. Utiliza un bus de direcciones de 16 bits y un bus de datos de 8 bits. De todas sus peculiaridades, la que más destaca es la organización de sus registros. Hay un solo acumulador, pero se puede utilizar toda la página de memoria 0 como registros para usos generales

68000
Motorola volvió a diseñar y a desarrollar el 6800 para crear el 6809, pero no llegó a tiempo para asegurar para el mismo un amplio sector del mercado de 8 bits. Esto habría de resultar ventajoso, dado que indujo a la empresa a desarrollar el procesador 68000, de 16/32 bits. El 68000 puede utilizar muchos de los chips de apoyo de las series 6502/6800 y está construido alrededor de ocho registros de datos de 32 bits y siete registros de direcciones de 32 bits

Z80
Teóricamente más potente que el MOS 6502, el Zilog Z80 utiliza estructuras de buses de datos y direcciones similares, pero posee un juego de registros considerablemente más fuertes (12 registros de 8 bits para fines generales y dos registros índice de 16 bits) y un juego de instrucciones mucho más amplio. Quizá su mayor ventaja respecto al 6502 sea su capacidad para apoyar el sistema operativo CP/M

Kevin Jones

Reparaciones

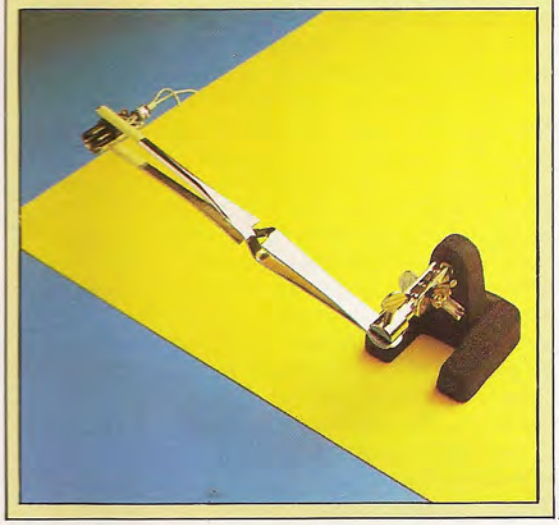
Comenzamos una serie de artículos dedicados a las reparaciones y añadidos mediante soldadura. El principiante podrá así efectuar él mismo esos trabajos que tan caros suelen cobrar los profesionales

Tanto la soldadura corriente como la fuerte son métodos para unir entre sí dos objetos de metal mediante una aleación metálica blanda (que, por tanto, se funde con facilidad); para los trabajos de electrónica se emplean el plomo y el estaño. Los objetos a unir se calientan a una temperatura superior al punto de fusión de la soldadura (280 °C), aplicamos el soldador a los componentes, los unimos y retiramos la fuente de calor. A medida que se enfrían, la soldadura se solidifica.

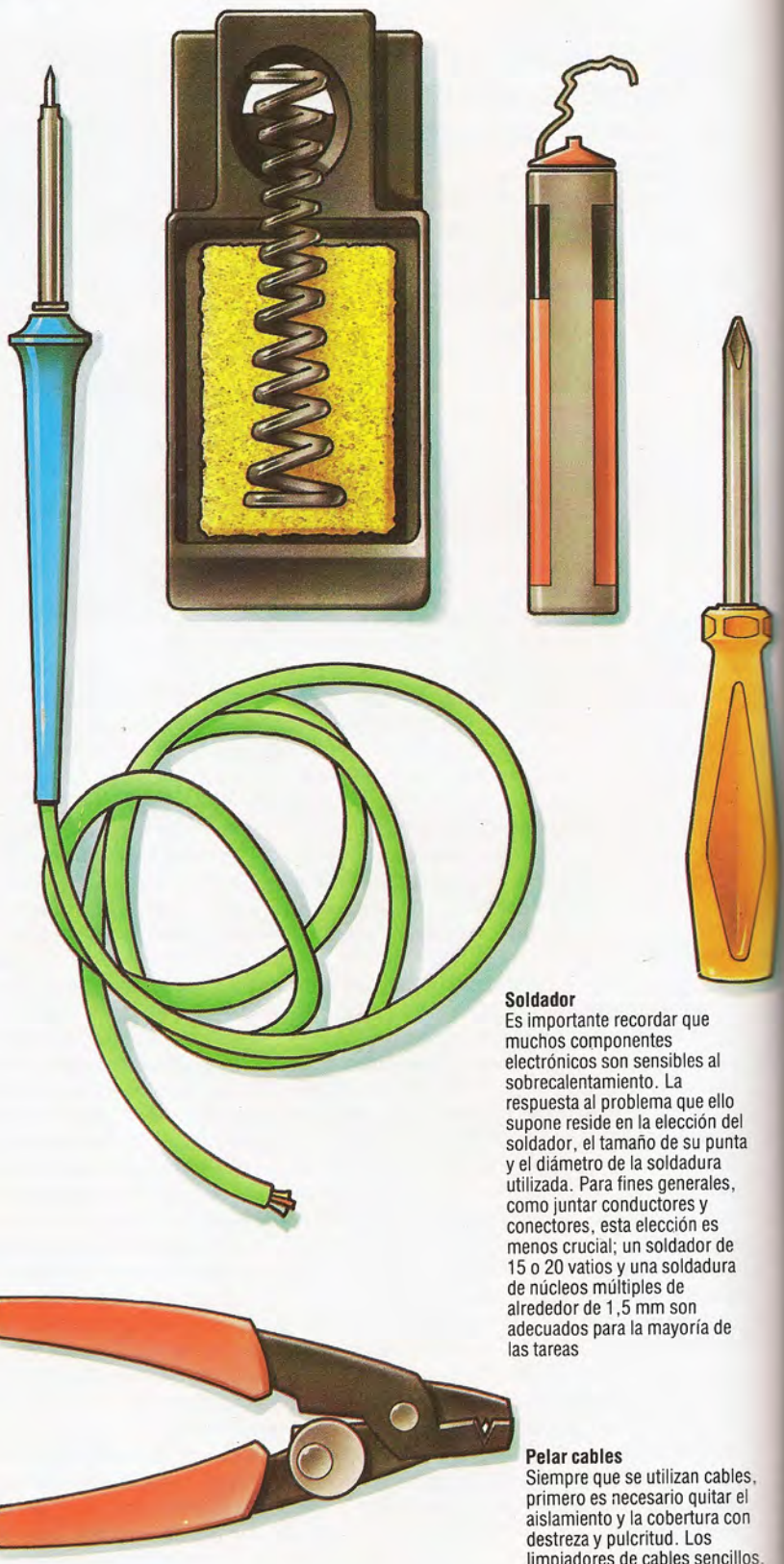
Un hierro para soldar aplicado directamente a una barra de soldadura la fundirá casi de inmediato. La soldadura caliente se enfriará casi en el acto al entrar en contacto con el componente frío. El resultado se conoce como juntura "seca". En el mejor de los casos, no permanecerá adherido. En el peor, establecerá una conexión muy pobre, que tal vez permita incluso un flujo intermitente de electricidad. Sólo existe una forma de evitar esta juntura imperfecta: calentar el componente hasta que el soldador se funda al contacto.

Eficiencia artesana

Una de las razones por las cuales los ordenadores forman parte de nuestra vida cotidiana es su reducido tamaño. Los elementos que los componen, por tanto, son tan diminutos que trabajar con ellos puede ser problemático. Existen varias prensas de tornillo y pinzas ligeras a un precio razonable. Si una pinza no agarra lo suficiente, hágale unos manguitos de cinta adhesiva con la goma hacia afuera, que recubran los brazos en forma de tenacillas

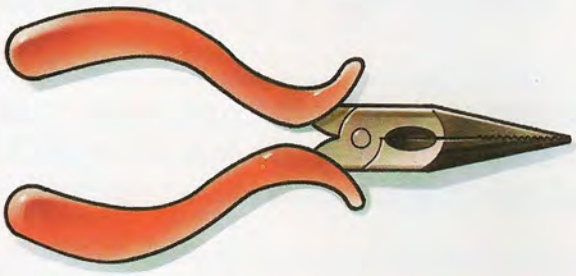


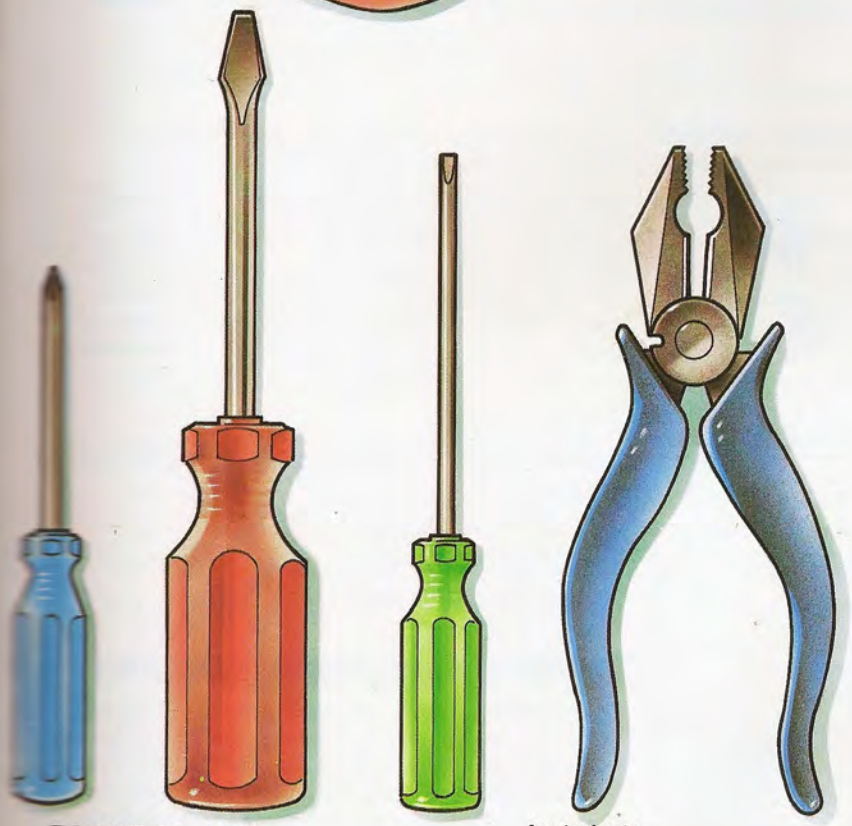
Ian McKinnell



Soldador
Es importante recordar que muchos componentes electrónicos son sensibles al sobrecalentamiento. La respuesta al problema que ello supone reside en la elección del soldador, el tamaño de su punta y el diámetro de la soldadura utilizada. Para fines generales, como juntar conductores y conectores, esta elección es menos crucial; un soldador de 15 o 20 vatios y una soldadura de núcleos múltiples de alrededor de 1,5 mm son adecuados para la mayoría de las tareas

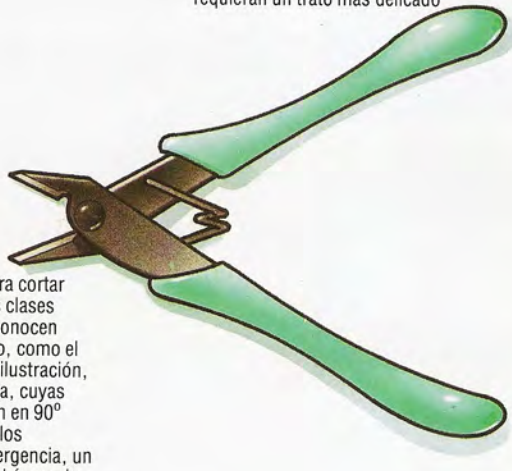
Pelar cables
Siempre que se utilizan cables, primero es necesario quitar el aislamiento y la cobertura con destreza y pulcritud. Los limpiadores de cables sencillos, como el que vemos aquí, son muy baratos y se pueden prefijar para una determinada profundidad de corte, quitando limpiamente el aislamiento pero dejando intacto el cable





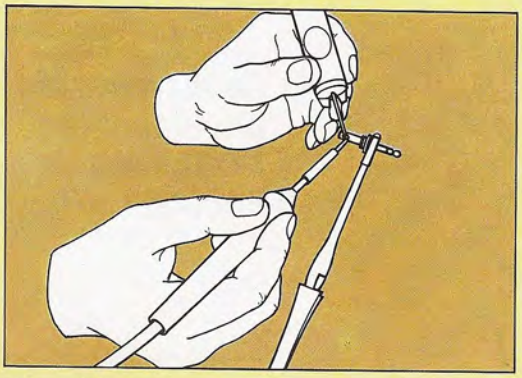
El destornillador adecuado
Existen dos variedades de destornillador: el corriente y el cruciforme, aunque en este último caso es necesario diferenciar entre el Philips y el Pozidriv. Por los tamaños en que se opera con la mayoría de los ordenadores personales, los dos modelos son intercambiables. Es probable que usted llegue a necesitar tanto el corriente como el cruciforme, y quizá en varios tamaños

Apretar fuerte
Para el principiante bastan dos clases de alicates: los alicates combinados, que pueden también utilizarse para cortar alambre en los casos más duros, y los redondos, mucho más ligeros, para las tareas que requieran un trato más delicado

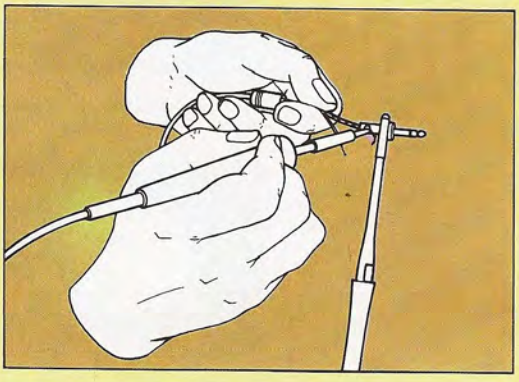


Punto de corte
Las herramientas para cortar alambres son de dos clases principales, que se conocen como fresas de disco, como el par que vemos en la ilustración, y cortadores en punta, cuyas puntas de corte están en 90° respecto al plano de los mangos. En una emergencia, un simple cortauñas podría servir para la tarea; ¡pero no espere que después de usarlo pueda volver a cortarse las uñas!

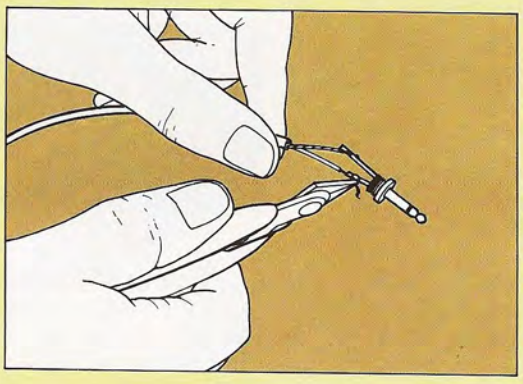
Soldar un enchufe



Pele y estañe
La primera etapa del proceso para formar una conexión consiste en pelar los alambres del cable retirando el aislamiento. No se quede corto. Pele más alambre del que necesite y recórtelo después. Sostenga el cable en la prensa del tornillo y caliéntelo con el soldador. Aplique la soldadura al cable y cuando lo comience a pasar utilice el soldador para guiarlo limpiamente a través de toda la superficie expuesta del cable. Este proceso, que se conoce como estañado, simplifica el posterior proceso de soldar; para entonces la soldadura estará sobre el componente. Repita esto mismo con el enchufe. Cuando el terminal todavía esté lo suficientemente caliente como para hacer correr la soldadura, aplique el cable estañado, quite la fuente de calor cuando la soldadura tanto del componente como del cable se haya fundido y esté unida, sóplelas para enfriarlas por debajo del punto de fusión, y la juntura ya está hecha



Manteniendo todo en orden
La siguiente etapa de la tarea consiste en recortar el material de desecho. Recorte los cables y también las patillas terminales de los componentes; pero sólo en el último momento, es decir, después de haber verificado la tarea. La razón del recorte es sencilla: un extremo largo de cable inservible, o una patilla terminal que sobresalga podrían hacer contacto con alguna otra cosa, lo que produciría un cortocircuito o, peor todavía, uno de esos enervantes fallos intermitentes



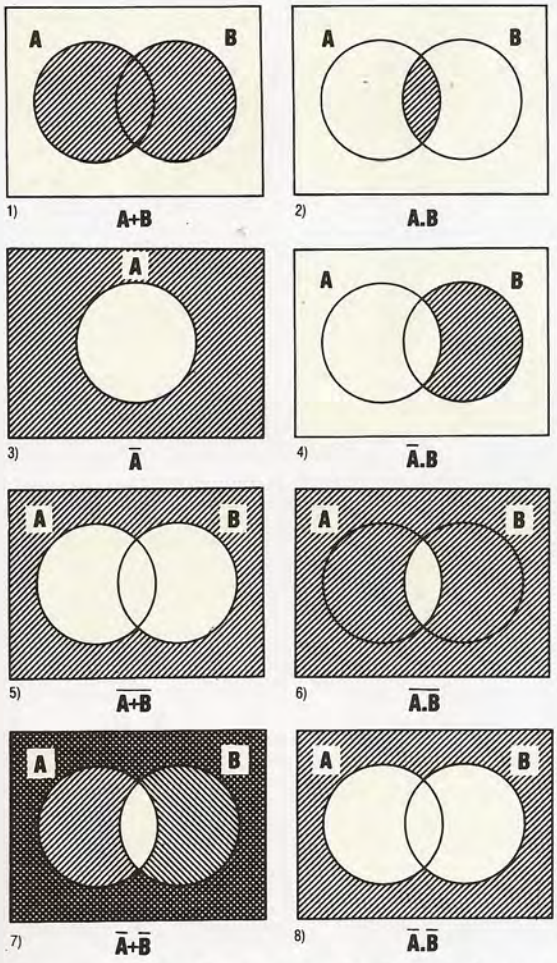
Michael Brownlow

Kevin Jones

Refinando el proceso

Saber cómo se simplifican expresiones complejas de álgebra booleana, reduciéndolas a un mínimo de operadores (AND, OR, NOT), será de especial importancia cuando se apliquen a los circuitos lógicos

Los diagramas de Venn son un complemento gráfico valioso para la simplificación de expresiones de álgebra booleana, al permitir dibujar el resultado de una expresión como zonas sombreadas. La superficie dentro de un rectángulo (con símbolo 1 = identidad o conjunto universal) representa todas las combinaciones posibles de valores verdaderos de las entradas, y los círculos dentro del rectángulo corresponden a combinaciones determinadas. A continuación ofrecemos algunas representadas en diagramas de Venn:



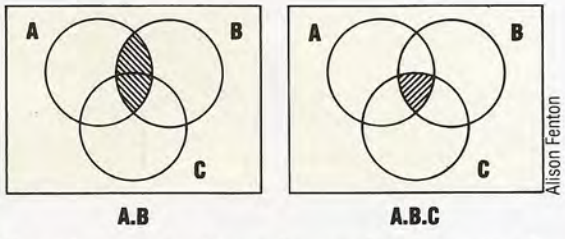
Comparando los diagramas 5 y 7 salta a la vista que NOT(A OR B) no es lo mismo que NOT(A) OR NOT(B). Del mismo modo, los diagramas 6 y 8 demuestran que NOT(A AND B) no equivale a NOT(A) AND NOT(B).

Tal vez la forma más sencilla de imaginar AND y OR en términos de diagramas de Venn sea pensando que A.B es la superficie común a A y a B; y que A + B es la unión de superficies de A y B. Varias relaciones son evidentes en el álgebra booleana. Para cada una de ellas usted podría intentar construir un diagrama de Venn como muestra de que son verdaderas. (0 representa el *conjunto vacío*, es decir, una combinación imposible.) He aquí estas seis relaciones evidentes:

1) A.A = A
2) A.$\overline{A}$ = 0
3) A.0 = 0
4) A.1 = A
5) A.(A + B) = A
6) A.($\overline{A}$ + B) = A.B

Leyes del álgebra booleana

El concepto de *dualidad* es otra valiosa, y esta vez intrigante, ayuda para la simplificación, que se basa en la simetría de los operadores AND y OR. Para formar la dual verdadera de cualquier relación verdadera, cambie todos los AND por OR y viceversa y, del mismo modo, todos los ceros por unos y viceversa. Por ejemplo, tomemos la quinta relación de las incluidas en la lista precedente. La dual de esta relación sería A + A.B = A. Esta expresión también es verdadera. Demuestra el importante principio de la *absorción*. Mirando un diagrama de Venn, es fácil ver que el término A.B cae enteramente dentro de A y, por consiguiente, se puede decir que ha sido absorbido por A. Esta idea se puede ampliar para un caso de tres variables, como A.B + A.B.C. = A.B. El par de diagramas de Venn que ofrecemos ilustra la última expresión:



Alison Fenton

Por otra parte, vuelva a observar los diagramas de Venn del principio. Comparando los diagramas 5 y 8, vemos que la siguiente relación importante siempre es verdadera: $\overline{A + B} = \overline{A}.\overline{B}$. Comparando los diagramas 6 y 7 vemos que: $\overline{A.B} = \overline{A} + B$. Estas

dos relaciones son las célebres *leyes de Morgan* y se pueden aplicar a casos más complicados como éste de tres variables: $\overline{A + B + C} = \overline{A}.\overline{B}.\overline{C}$ y $\overline{A}.\overline{B}.\overline{C} = \overline{A + B + C}$). Las leyes de Morgan se pueden aplicar por etapas:

$\overline{(A + B)}.\overline{C}$

$= \overline{A}.\overline{B}.\overline{C}$ (según Morgan, diagramas 5 y 7, aplicado al paréntesis)

$= \overline{A + B + C}$ (según Morgan, diagramas 6 y 8, tomando como un solo componente el paréntesis)

Existen además tres propiedades del álgebra normal que se pueden aplicar al álgebra booleana. La *propiedad asociativa* permite trastrocar los paréntesis:

$(A.B).C = A.(B.C) = A.B.C.$

$(A + B) + C = A + (B + C) = A + B + C$

El orden en que se escriben las letras se puede alterar por la *propiedad conmutativa*:

$A.B. = B.A.$

$A + B = B + A$

La *propiedad distributiva* permite multiplicar los paréntesis:

$A.(B + C) = A.B + A.C$

Ejemplos de simplificación

1) Simplificar $(\overline{A + B} + \overline{A}.B).B$

$= (\overline{A}.\overline{B} + \overline{A}.B).B$ (Morgan)

$= \overline{A}.\overline{B}.B + \overline{A}.B.B.$ (propiedad distributiva)

$= 0 + \overline{A}.B$ ($\overline{B}.B = 0$, $B.B = B$)

$= \overline{A}.B$

2) Simplificar $\overline{A}.\overline{B} + \overline{A}.B + A.B$

$= \overline{A}.(\overline{B} + B) + A.B$ (propiedad distributiva)

$= \overline{A} + A.B$ ($\overline{B} + B = 1$)

$= \overline{A} + B$ (dual de relación 6)

3) Simplificar $\overline{\overline{A} + B} + \overline{\overline{A} + \overline{B}} + \overline{A}.B$

$= \overline{\overline{A}}.\overline{B} + \overline{\overline{A}}.\overline{\overline{B}} + \overline{A}.B$ (Morgan)

$= A.\overline{B} + A.B + \overline{A}.B$ ($\overline{\overline{A}} = A$)

$= A.(\overline{B} + B) + \overline{A}.B$ (propiedad distributiva)

$= A + \overline{A}.B$ ($\overline{B} + B = 1$)

$= A + B$ (dual de relación 6)

Una puerta XOR simplificada

En el capítulo anterior analizamos un circuito no simplificado para una puerta del OR-exclusivo (o sea, XOR). Volvamos ahora a examinar el mismo problema, pero esta vez sabemos cómo simplificar la expresión booleana resultante y, por tanto, el circuito. La tabla de verdad para la puerta XOR era:

ENTRADA		SALIDA
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Dijimos que esta tabla de verdad previamente se resumía así: $C = \overline{A}.B + A.\overline{B}$. Aquí poca simplificación se puede hacer, y debemos resignarnos a usar un circuito de cinco puertas. Pero hay un enfoque alternativo para abordar este problema. A partir de la tabla de verdad se puede decir que C es 1 si A y B no son ambos 1 o ambos 0. En términos booleanos podemos escribir:

$C = \overline{A.B + \overline{A}.\overline{B}}$

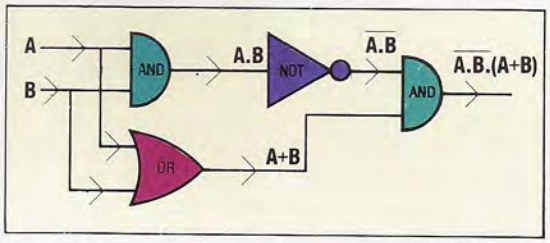
Utilizando repetidamente las leyes de Morgan, podemos simplificar el circuito para obtener:

$C = (\overline{A.B}).(\overline{\overline{A}.\overline{B}})$

y por último:

$C = \overline{A.B}.(A + B)$

que sólo necesita un circuito de cuatro puertas:



Liz Dixon

Circuito sumador completo

Anteriormente analizamos el proceso de la suma binaria y diseñamos un circuito simple para sumar dos bits que tuviera dos salidas, una para el dígito de la suma parcial y otra para llevo. A este circuito lo denominamos sumador medio. Si a la primera entrada la llamamos X y a la segunda entrada Y, podemos observar que, a partir de la tabla de verdad de un sumador medio (véase p. 513), la salida (S) suma (el resultado) corresponde a la expresión booleana: $S = \overline{X}.Y + X.\overline{Y}$. Utilizando la ley de Morgan, obtenemos esta simplificación: $S = \overline{X.Y}.(X + Y)$. La salida llevo (C) es sencillamente: $C = X.Y$.

En aritmética binaria hay, de hecho, tres dígitos a sumar en una columna cualquiera de la suma de adición. Además de los dos dígitos a sumar, hay también un llevo procedente de la columna previa que incluir. Para poder reproducir el proceso de la suma normal debemos diseñar un circuito con tres entradas y dos salidas. Si al llevo de la columna anterior lo llamamos P, entonces la tabla de verdad para un sumador completo sería:

ENTRADAS			SALIDAS	
P	X	Y	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tomando los casos en los que S = 1, la expresión booleana para S a partir de la tabla de verdad sería la siguiente:

$$S = \overline{P}.\overline{X}.Y + \overline{P}.X.\overline{Y} + P.\overline{X}.\overline{Y} + P.X.Y$$

Utilizando las reglas que hemos aprendido, podemos simplificar esta expresión:

$$S = \overline{P}.(\overline{X}.Y + X.\overline{Y}) + P.(\overline{X}.\overline{Y} + X.Y)$$

(propiedad distributiva)

$$S = \overline{P}.(\overline{X}.Y + X.\overline{Y}) + P.\overline{(\overline{X}.Y + X.\overline{Y})}$$

(ley de Morgan)

Del mismo modo, podemos formar una expresión para C. A partir de la tabla de verdad:

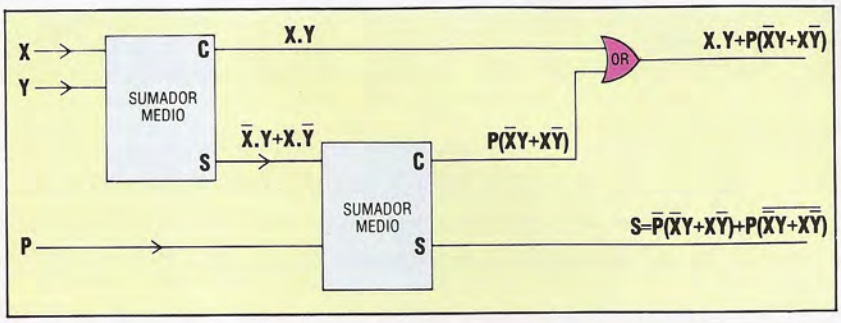
$$C = \overline{P}.X.Y + P.\overline{X}.Y + P.X.\overline{Y} + P.X.Y$$
$$C = X.Y.(\overline{P} + P) + P.(\overline{X}.Y + X.\overline{Y})$$

(propiedad distributiva)

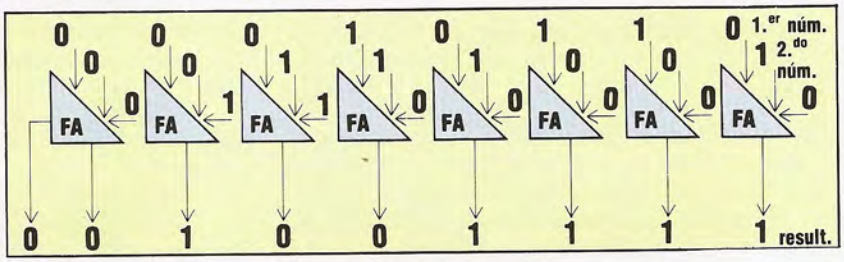
$$C = X.Y + P.(\overline{X}.Y + X.\overline{Y})$$

$(\overline{P} + P = 1)$

Observe que $\overline{X}.Y + X.\overline{Y}$ es la salida suma de un circuito sumador medio. Por consiguiente, se puede diseñar un circuito sumador completo a partir de dos sumadores medios.



Veamos cómo una serie de ocho sumadores completos se combinan dentro de la ALU para efectuar la adición binaria de dos números de ocho bits.



EJERCICIO 3

1) Simplificar estas expresiones:
   a) $A.(\overline{A} + \overline{B})$
   b) $X + Y.(X + Y) + X.(\overline{X} + Y)$
   c) $P.Q + \overline{P}.Q + \overline{P}.\overline{Q}$
   d) $\overline{\overline{X} + Y.\overline{Z} + \overline{Z}.Y}$

2) Una alarma para automóvil posee un interruptor on/off e interruptores en las puertas delanteras. La alarma sonará si una de éstas o ambas se abrieran cuando el interruptor on/off estuviera colocado en *on*. Dibuje una tabla de verdad que muestre las tres entradas y la salida alarma. Utilice esa tabla de verdad para escribir una expresión booleana para el sonido de la alarma y dibuje un circuito lógico para el sistema de alarma.

3) La luz de un recibidor funciona desde un interruptor situado en la puerta, un interruptor situado al pie de la escalera o uno situado arriba de las escaleras. Diseñe un circuito lógico apropiado.

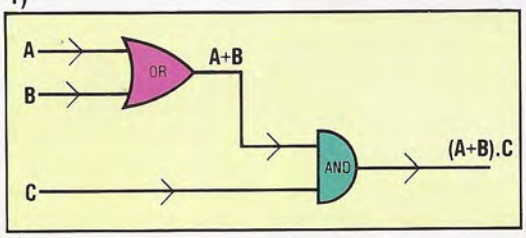
4) Acaban de abandonarle en una isla desierta con otras dos personas. Una de ellas siempre dice la verdad, la otra siempre miente. Si en algo aprecia su vida, ha de descubrir cuál de las dos es amiga de la verdad. Para ello hay varias preguntas que podría formularles a ambas. Confeccione tablas de verdad para investigar las posibles respuestas. He aquí un ejemplo:

"¿Dice usted siempre la verdad?"

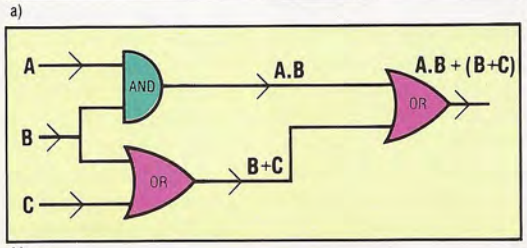
		POSIBLES RESPUESTAS	
		SI	NO
POSIBLE IDENTIDAD DE QUIEN CONTESTA	MIENTE	1	0
	DICE LA VERDAD	1	0

Respuestas al ejercicio 2 de página 513

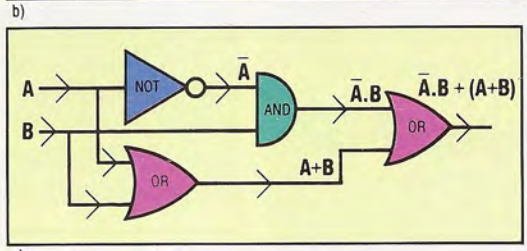
1)



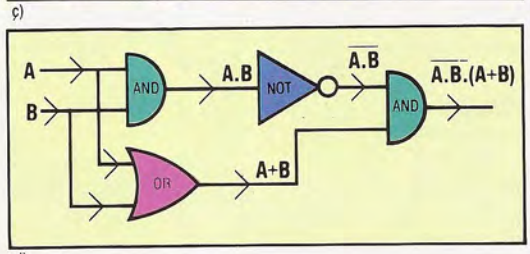
a)



b)



ç)



d)

2a) $C = A.B$
b) $S = \overline{A.B}\ (A + B)$
3a) $X = A + B).(B.C + C)$
b) $X = \overline{A.B} + \overline{B}$

Diagrama de bloques

No siempre ha de utilizarse este diagrama, pero en los casos más complejos evidencia la lógica del programa con toda claridad

El diagrama de bloques, llamado también *flowchart de programa*, es aquel que representa detalladamente las funciones de un programa y sus relaciones. Los procedimientos que llevan a la solución de un problema se componen de una sucesión de fases elementales en correcta secuencia. Este desarrollo se hace menos necesario a medida que el problema pierde complejidad, ya que en muchos casos se puede resolver simplemente con la aplicación de una fórmula, prescindiéndose, dada su síntesis, del empleo de un proceso. Pero en problemas complicados, resulta casi indispensable acudir a él para alcanzar el resultado.

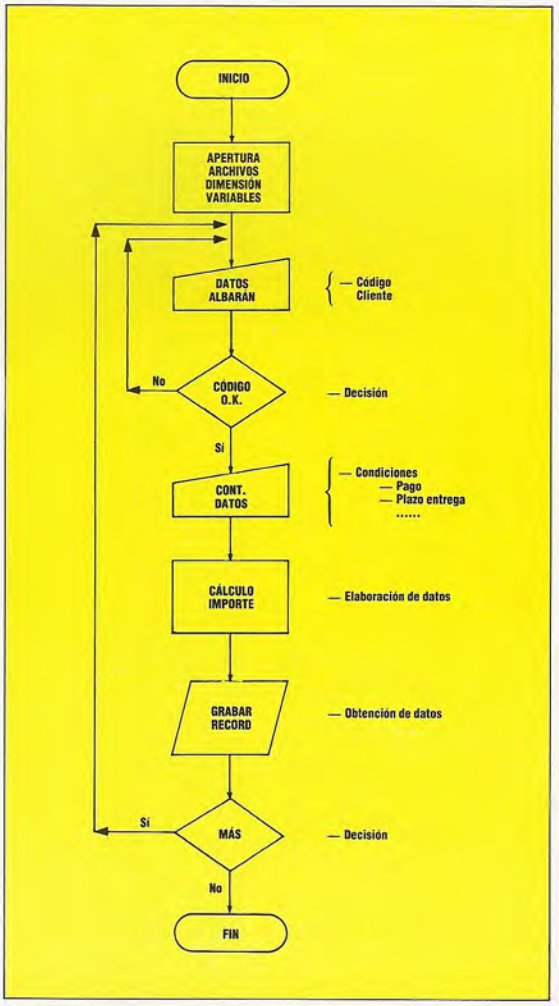
Así, definiendo las fases elementales de las diferentes alternativas de un problema y describiendo su secuencia correcta, puede llegarse a programar su solución. De igual manera, un diagrama de bloques proporciona una serie de informaciones diferentes. En primer lugar, sirve como documentación del programa, ya que se puede recurrir a él en caso de duda de la secuencia utilizada durante la programación o después de ella, y al mismo tiempo muestra de forma gráfica la solución del problema, a la que se llega tras seguir la secuencia de símbolos.

Un diagrama de bloques debe, asimismo, representar la lógica del programa, y en último lugar ha de verificar que se han tenido presentes todas aquellas variantes posibles que puedan influir en la resolución, sin dejar ningún punto al azar.

Tomando el ejemplo del capítulo anterior sobre la automatización de la gestión de una empresa figurada, que sirvió para ver la utilización del diagrama de flujo, representemos el que nos ocupa actualmente y que será aquel que se entregará al programador para que pueda llevar a cabo su labor.

1. En primer lugar, se hará la apertura de los archivos con los que se vaya a trabajar (clientes, artículos, etc.) y se dimensionarán una serie de variables destinadas a contener los datos que se utilizarán a lo largo del desarrollo del programa (tablas, contadores, acumuladores, etc.). Habrá que desarrollar después lo que se va a convertir en la secuencia repetitiva del programa.
2. Comienza con la entrada por teclado del número de código del cliente a cuyo cargo corre el albarán.
3. Debe hacerse en este punto una comprobación (de forma automática) consistente en comparar el mencionado código con el registro correspondiente, dando opción así a dos posibles salidas: *a*) en caso de que el código sea erróneo, la salida supondrá volver de nuevo al punto *b*, donde se introducirá nuevamente el código del cliente; y *b*) si se comprueba que el código es correcto, el proceso pasará secuencialmente a las demás fases.
4. Se da aquí entrada, también por teclado, al resto de datos que se precisen para poder actualizar el registro correspondiente y para usarlos en las pertinentes operaciones.
5. Supone este paso la elaboración y obtención de resultados. En él se realizarán las operaciones necesarias, partiendo de los datos disponibles.
6. Al llegar aquí, los datos modificados se incluyen en el registro y éste se graba nuevamente en el fichero, conservándolo así debidamente actualizado.
7. Por último, se ofrece la posibilidad de repetir el proceso tantas veces como se quiera. Si se continúa, la secuencia retornará al punto *b*; en caso contrario, finaliza el programa.

Con este ejemplo, cuya solución gráfica se adjunta, queda patente la utilidad del diagrama de bloques, del que un buen programa no debería prescindir en la mayoría de los casos.



Hasta el límite

La tercera versión del "best-seller" Spectrum (50 000 unidades vendidas en España en tan sólo 11 meses) presenta particularidades dignas de ser analizadas más de cerca

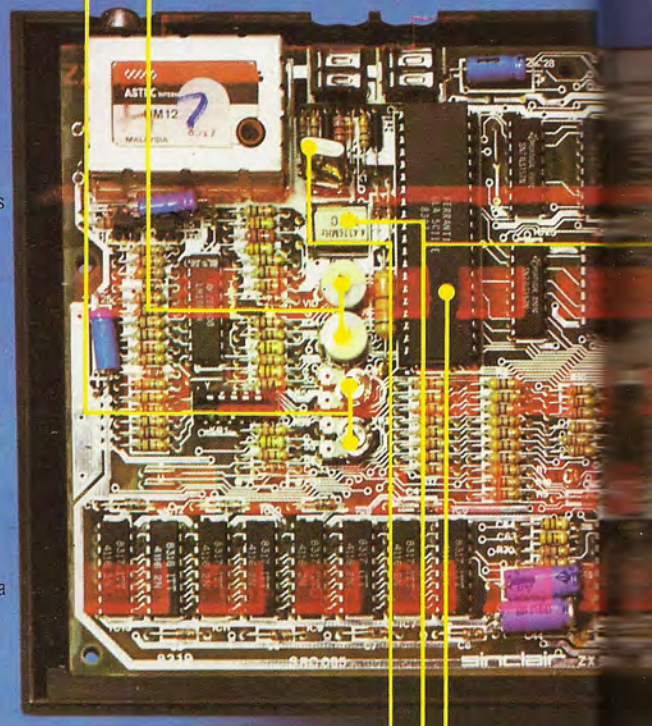
Como elemento de hardware, la peculiaridad más notable del Spectrum es el teclado. Si bien podemos afirmar que es del tipo QWERTY, hasta aquí llega todo su parecido con una máquina de escribir. Cada una de las 40 teclas es parte de una membrana que confiere a las teclas cierto recorrido: al "tacto", estas teclas parecen de esponja.

El Spectrum posee un conector de buses del sistema, que permite al usuario conectar simultáneamente la impresora ZX (diseñada originalmente para el ZX81), la interface 1 ZX y la interface 2 ZX. También hay conectores MIC y EAR, que permiten el almacenamiento en cassette de programas. El procedimiento para cargar programas no es muy satisfactorio. Si bien al cargar y guardar correctamente aparecen sobre el borde de la pantalla unas franjas azules y amarillas que así lo indican, para poder guardar un programa uno debe primero desconectar el conductor EAR. Igualmente incómoda es la carencia de un botón Reset en el ordenador, lo que significa que cada vez que se produce una rotura del sistema se ha de quitar el conector de alimentación eléctrica, lo que con el tiempo podría debilitar las conexiones. Por suerte, algunas pequeñas firmas independientes producen dispositivos accesorios que incorporan tanto un interruptor para guardar o cargar *(save or load)* como un interruptor para restauración *(reset)*.

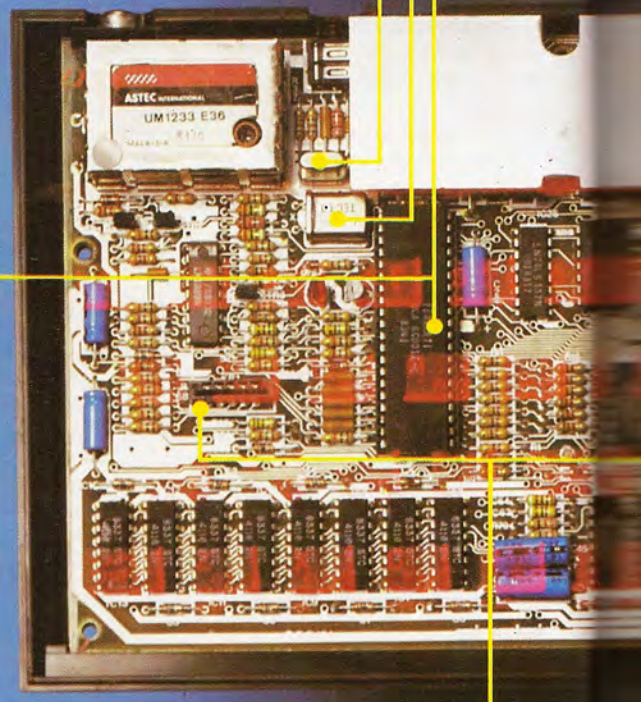
Esto es sólo un ejemplo de la forma en que Sinclair Research parecería ceder en su interés por la máquina. Puede que sea, no obstante, una política deliberada de la empresa, porque mientras disfruta de los beneficios de las ventas de su ordenador, Sinclair no ahorra esfuerzos en un proyecto como el QL. Aun así, la empresa ha desarrollado los microdrives ZX y la unidad relacionada interface 1 ZX, que proporcionaron el potencial de almacenamiento de apoyo de 680 Kbytes, una puerta RS232 y la idea de conectar en redes de área local hasta 64 unidades Spectrum. Asimismo, se introdujo la unidad interface 2 ZX, que permite acceso al software basado en RAM y conectar dos palancas de mando.

Sinclair Research también ha dejado la producción de software en manos de otros, al haber dado acogida a ciertos paquetes producidos por firmas de software independientes con su autorización.

Evolución del Spectrum
Desde que se introdujo en el mercado británico, en 1982, hasta su eclipse parcial por el QL, a comienzos de 1984, del ZX Spectrum de Sinclair se vendió más de un millón de unidades y pasó por tres versiones en su tablero original (o sea, tablero de circuito impreso que sostiene todos los componentes principales). La primera versión se utilizó sólo para las primeras 60 000 unidades que se vendieron, de modo que no es común. Las dos últimas versiones difieren en dos aspectos primordiales. Primero, en la segunda versión se podía "afinar" el sistema de circuitos de salida de video mediante los dos condensadores de equilibrio y las dos resistencias variables que vemos en la fotografía. En segundo lugar, la "modificación temporal" del microprocesador perteneciente a la segunda versión se logró formalizar cuando se introdujo la tercera versión. El disipador de calor está en un lugar diferente porque el chip regulador de voltaje se ha colocado más cerca del conector para entrada de corriente

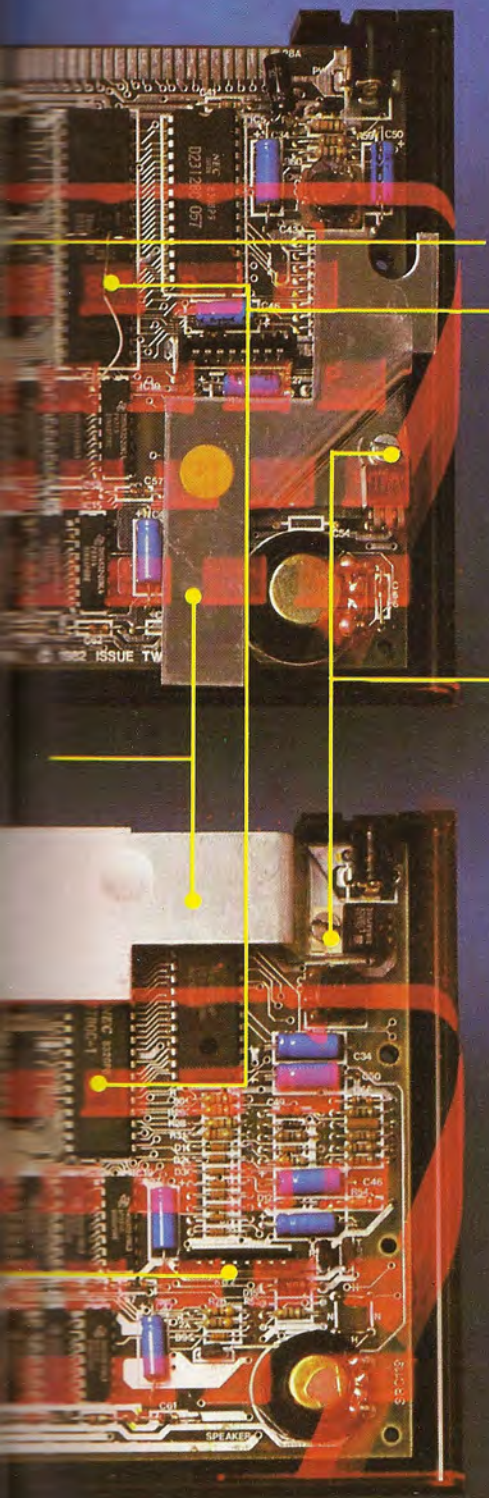


Cristal del reloj principal
Funciona a 14 MHz



Disposición lógica no comprometida
Este chip de 40 patillas sustituye a una gran variedad de chips lógicos, controla las operaciones de entrada-salida, incluyendo la generación de una señal de video compuesta, que después se modula con una frecuencia de radio, y controla los interruptores de la CPU

Conectores



Cristal reloj video en color
Funciona a 4,4336 MHz

Unidad Central de Proceso Z80A
Observe la "modificación temporal" para los tableros originales de la segunda versión, haciendo un puente entre las patillas 11 y 30 mediante un transistor. Esta modificación tiene el efecto de permitir que se seleccione la ULA sólo cuando la línea de dirección cero y la IORQ (*Input/Output Request:* solicitud de entrada-salida) están las dos bajas

Regulador de voltaje

SINCLAIR SPECTRUM

DIMENSIONES
233 x 144 x 30 mm

CPU
Z80A

CAPACIDAD DE MEMORIA Y VELOCIDAD
16 K y 48 K
3,5 MHz

CARACTERISTICAS DE LA PANTALLA
La pantalla está dividida en 24 líneas de 32 caracteres. Gráficos con mapas de bits a una resolución de 256 x 192. Dieciséis caracteres ya programados para gráficos de bloques más 21 caracteres para gráficos definibles por el usuario. Ocho colores, flash y dos niveles de brillo. Color independiente para el borde

INTERFACES Y PUERTAS
Conector para bus del sistema. Conectores para almacenamiento en cassette y televisor

LENGUAJES DISPONIBLES
BASIC y lenguaje ensamblador Z80. La ampliación mediante software permite utilizar FORTH, LOGO, Micro-PROLOG y otros

TECLADO
Teclado de 40 teclas móviles ASCII tipo membrana. El diseño permite reclamar hasta ocho funciones desde una sola tecla a través de una serie de teclas Shift (cambio)

DOCUMENTACION
El ordenador viene completo con un pequeño manual de introducción y un manual más amplio con una guía para operar el micro y orientación acerca del BASIC de Sinclair

PUNTOS FUERTES
En su versión de 48 K, el Spectrum ofrece una configuración ideal para experimentación del recién iniciado. Goza de gran apoyo de software y hardware producido por fabricantes independientes

PUNTOS DEBILES
El teclado, aunque diseñado con gran inteligencia, no ofrece las facilidades del tipo máquina de escribir. La visualización en pantalla está configurada de una forma no estandarizada, y puede plantear problemas a las personas sin experiencia

Ian McKinnell

El archivo del Spectrum
Además del elevado número de paquetes de juegos existentes para el Spectrum, también existe en el mercado una estimable cantidad de software para gestión y administración doméstica, gran parte del cual se vende a un precio muy asequible. Con la introducción del microdrive ZX, no existe razón alguna por la cual el Spectrum no pueda aplicarse al mantenimiento de grandes bases de datos, hojas electrónicas y tratamiento de texto.
Psion, uno de los colaboradores más próximos de Sinclair en la producción de software, ha sido el responsable del nacimiento de gran parte de este material. Entre éste destaca la producción de la exitosa serie de paquetes VU, que incluye el VU-CALC, VU-File y VU-3D. Y, a pesar de las deficiencias del teclado del Spectrum, Microl ha producido *The word processor* (procesador de textos), que ofrece espacio de almacenamiento equivalente a 10 páginas A4 de texto, y la mayoría de las facilidades que normalmente ofrecen los paquetes más completos para edición de textos (incluyendo la posibilidad de mezclar archivos)

Unidad emancipada

A diferencia de otros sistemas de disco, las unidades Commodore son "inteligentes", ya que cuentan con su propio microprocesador y su RAM

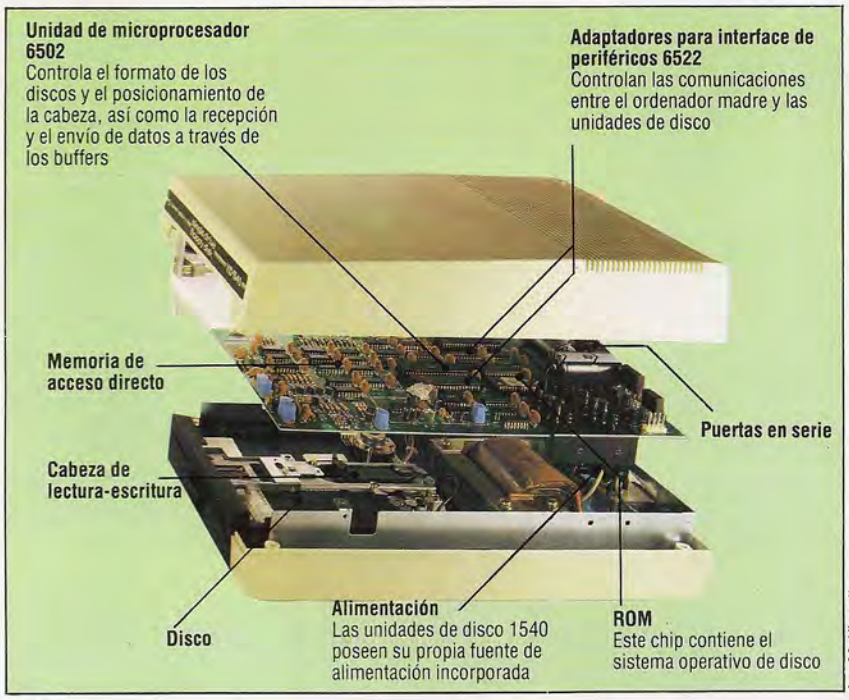
El principal problema de las unidades de disco inteligentes es que su fabricación es cara. Después de la introducción del ordenador personal Vic-20, Commodore lanzó una versión económica de la unidad de disco PET, denominada Vic-1540. El Commodore 64 incorpora facilidades similares a las del Vic-20 para acceder a la 1540, pero diferencias menores hicieron que fuera necesario efectuar una POKE antes de utilizar la unidad y otra POKE al terminar. Pero tan engorroso procedimiento ya no es necesario, porque Commodore realizó algunas modificaciones en el DOS para rectificar el defecto y volvió a lanzar la 1540 como la 1541. Esta versión más reciente es totalmente compatible tanto con el Vic-20 como con el 64. Por razones de simplicidad nos referiremos a ambas unidades como 1541, porque no existe ninguna diferencia en cuanto a la forma en que se utilizan.

La 1541 se controla a través de un microprocesador 6502, dos VIA (*Versatile Interface Adaptors*: adaptadores versátiles para interface) 6522, dos Kbytes de RAM y ocho Kbytes de ROM, que contiene el DOS. El DOS que se proporciona es muy potente y permite programar complicadas rutinas para crear y manipular archivos de programas (PRG), archivos de datos secuenciales (SEQ) y archivos de acceso directo, todos con refinados procedimientos para verificación de errores. El control del ordenador se ejerce a través de una versión en serie de la interface IEEE488. Esta interface admite las mismas órdenes que su equivalente en paralelo más poderoso (a través del cual se controlan los otros periféricos Commodore) y permite "conectar en cadena margarita" periféricos dotados de IEEE488 en serie, de modo que, por ejemplo, una unidad de disco puede dar salida a archivos hacia una impresora mientras el ordenador lleva a cabo otra tarea. Esto se consigue mediante la utilización de órdenes con los números de archivos lógicos y números de los dispositivos correspondientes.

El formato de los diskettes es de 35 pistas por cara; las pistas están divididas en sectores, desde 21 en la pista exterior hasta 17 en la más interior. Cada sector contiene un bloque de 256 bytes de datos de archivo, más datos de sincronización, identificación y suma de control. Cada diskette almacena 683 bloques, de los cuales 664 están disponibles para el usuario. Esto otorga una capacidad máxima de aproximadamente 170 Kbytes, según el tipo de archivos que se almacene. El DOS administra la distribución de datos en un diskette manteniendo un mapa de disponibilidad de bloques (*Block Availability Map*: BAM) y un directorio. El BAM se almacena en la pista 18, sector 0, y consta de 144 bytes que indican qué bloques están en uso y cuáles están libres para el almacenamiento. El directorio comienza en el sector 1, pista 18, y es una lista de un máximo de 144 archivos, con el nombre de cada uno, que contiene información específica relativa al tipo de archivo y de cuántos bloques consta. Tanto el BAM como el directorio se van actualizando a medida que se escriben datos en el diskette o que se eliminan del mismo.

A pesar de su elevado precio de venta, la flexibilidad del sistema de unidad de disco Commodore ofrece una magnífica relación calidad-precio. Las posibilidades de su fiable almacenamiento masivo (que se puede incorporar a un sistema de administración de periféricos y no se inmiscuye ni en la memoria del ordenador ni en el tiempo del procesador) justifican el gasto. No obstante, es lamentable que, debido a la interface en serie, la 1541 sea de operación comparativamente lenta. En el manual para el usuario, Commodore ni siquiera menciona la velocidad de transferencia de datos ni el tiempo de acceso promedio. Aunque es con creces 50 veces más rápido que el almacenamiento de cassette, es probable que el 1541 sea el de más lento acceso de todos los sistemas de disco más populares. Si se necesita una gran capacidad de almacenamiento, se puede adquirir una interface que se conecta en la puerta en serie y que imita a la puerta en paralelo de que disponen los ordenadores PET. Esto no acelera especialmente las operaciones, pero sí permite que usted conecte toda la gama completa de periféricos PET a un Vic-20 o un Commodore 64.

La unidad de disco Commodore
Disponible tanto para el Vic-20 como para el Commodore 64, al ser una configuración estándar del más reciente ordenador portátil SX-64, la unidad de disco 1540 de Commodore es una unidad inteligente que no exige nada ni a la CPU ni a la memoria del ordenador al que se acopla. De hecho, posee su propio procesador, el MOS Technology 6502, que alimenta a los mismos ordenadores



Ian McKinnell

El manejo del disco...

El DOS de Commodore admite una amplia gama de órdenes diseñadas para que el usuario construya complicados programas para manipular archivos de acceso directo, así como el programa normal y la manipulación de archivos de datos. Estas órdenes se utilizan tal como vemos más abajo. En todos los casos, 8 es el número identificador de dispositivo de la unidad de disco.

SAVE
Crea archivos de programas (PRG) con nombre (hasta 16 caracteres) que pueden ser programas o datos secuenciales. El formato es el siguiente:

SAVE"NOMBRE ARCHIVO",8

LOAD
Se construye así:

LOAD"NOMBRE ARCHIVO",8

Esta orden copia el archivo PRG especificado en RAM desde la parte inferior de la memoria para usuario hacia arriba. La orden

LOAD"NOMBRE ARCHIVO",8,1

vuelve a copiar el archivo especificado en las posiciones de memoria de las cuales se tomó para guardar (SAVE) originalmente.

LOAD"$",8

copia el directorio del disco en la memoria para el usuario. Luego se puede LISTar como un programa en BASIC y contiene lo siguiente:

Nombre del disco
Identificador de disco de 2 caracteres
Hasta 144 nombres de archivos
Tipo (PRG o SEQ) de cada archivo
Longitud en bloques de cada archivo
Número de bloques libres disponibles

VERIFY
Se construye así:

VERIFY"NOMBRE ARCHIVO",8

y compara el archivo especificado con el archivo que contiene la memoria para el usuario y genera un mensaje de error en el caso de que sean diferentes. Se utiliza para verificar que los archivos se hayan guardado (SAVE) correctamente.

OPEN
Establece un canal de comunicación exclusivo que se identifica mediante un "número de archivo lógico" (*Logic File Number: LFN*) dentro del intervalo [1, 255]. Se pueden abrir (OPEN) simultáneamente hasta 10 LFN. OPEN también establece una "dirección secundaria" (*Secondary Adress: SA*), que determina la forma en que se comportará el dispositivo al cual se accede. La única dirección secundaria de la unidad de disco es 15, que da acceso al canal de prioridad *command* (orden). A OPEN se le da entrada del siguiente modo:

OPEN LFN,8,SA

CLOSE
Asume el formato:

CLOSE LFN

Termina el archivo lógico especificado. Los archivos lógicos siempre se deben cerrar (CLOSE) cuando ya no se los necesita.

PRINT#, INPUT# y GET#

PRINT# funciona de modo similar a PRINT con la excepción de que a los datos se les da salida, como un archivo SEQ, al archivo lógico OPEN especificado y no a la pantalla. Se construye así:

PRINT#LFN,"DATOS", o bien
PRINT#LFN,A$,B$,...

Del mismo modo, INPUT# y GET# leen archivos SEQ. INPUT# recupera datos en cadena pero sólo es eficaz si las cadenas almacenadas están separadas mediante punto y coma o sólo comas, de lo contrario INPUT# tratará los datos como una cadena larga. GET# recupera los datos a un byte por vez, incluyendo las comas, o los punto y coma. Ésta es más útil cuando no se conoce el contenido de un archivo y éste no está separado. Los siguientes ejemplos ilustran los formatos:

INPUT#LFN, A$, B$...
GET#LFN, A$,B$...,

Cuando PRINT# se utiliza junto con un archivo lógico abierto (OPEN) al canal de órdenes (p. ej., OPENLFN,8,15) de esta manera:

PRINT#LFN,8,15,"serie de órdenes"

se transforma en la orden para manipulación de disco más poderosa de las disponibles. Las series de órdenes se utilizan para ejecutar órdenes para mantenimiento de disco y avanzadas órdenes para archivos de acceso directo (*relative*: REL).

...y su mantenimiento

Utilizadas conjuntamente con PRINT# u OPEN en el canal de órdenes, en el formato indicado, estas series de órdenes realizan las siguientes funciones:

NEW
Da formato y nombre al diskette
Construye el BAM y el directorio
Asigna el identificador de disco de 2 caracteres (DI)
Orden: "N:NOMBRE DISCO,DI"

INITIALISE
Verifica el BAM en RAM de disco con el BAM del disco
Orden: "I"

VALIDATE
Elimina los bloques destinados mediante avanzadas órdenes REL no retenidas en el directorio y archivos que no se han cerrado (CLOSE)
Escribe un BAM nuevo
Orden: "V"

RENAME
Modifica el listado del directorio de un archivo especificado
Orden: "R:NOMACTUAL = NOMANTERIOR"

SCRATCH
Borra archivos especificados del disco o directorio
"S:NOMARCH 1, NOMARCH 2,..."

COPY
Escribe una copia de un archivo en el mismo disco
Orden: "C:NOMDUP + NOMORIS"
Junta archivos SEQ y los escribe como un único archivo SEQ en el mismo disco. Conocida como orden "concatenante":
"C:CONNOMB = NOMB1,NOMB2,..."

Verificación de errores
En el panel frontal de la unidad de disco 1541 hay un LED (*Light Emitting Diode*; diodo emisor de luz) verde "encendido" y un LED rojo que indica el estado del disco, donde:

Encendido = Leyendo un disco o escribiendo en él
Apagado = Esperando instrucciones
Intermitente = El DOS ha detectado un error

Para descubrir la naturaleza del error se debe leer el canal de error del DOS. El siguiente programa imprime los códigos de error generados por el DOS. En el manual para el usuario de la unidad de disco se proporciona una lista de los códigos de error y sus significados.

```
10 REM**VERIFICACION
   ERROR DE DISCO**
20 OPEN15,8,15
30 INPUT#15,EN,EM$,ET,
   ES
40 PRINTCHR$(147)
50 PRINT"N.º ERROR"EN
60 PRINTEM$
70 PRINT"PISTA"ET
80 PRINT"SECTOR"ES
90 CLOSE15:END
```

El ABC del BBC

Continuando con nuestro estudio del BASIC de los ordenadores personales más populares, ahora examinaremos las particularidades de la versión que incorpora el BBC Micro, una de las más elogiadas

Entre líneas

No olvide numerar las líneas de su programa en múltiplos de diez. Hasta los mejores programadores tienen que hacer inserciones en sus programas de cuando en cuando...

La crítica que suele suscitar con más frecuencia el BASIC es que se trata de un lenguaje sin estructurar que favorece (o, al menos, que no hace nada por controlar) malos hábitos de programación en el principiante, en particular la solución "rápida y sucia" de los problemas, lo que lleva, por ejemplo, a la utilización indisciplinada de GOTO.

El empleo de ELSE con la sentencia IF...THEN puede eliminar la utilización más común de GOTO, al permitir que en la misma sentencia se traten tanto los casos verdaderos de una condición como los falsos. Por ejemplo, estas líneas:

```
1500 IF TEST > 0 THEN GOTO 1800
1600 PRINT "VALOR FUERA DE ESCALA"
1700 GOTO 1900
1800 PRINT "NO HAY PROBLEMA"
1900 NEXT L
```

se pueden sustituir por:

```
1500 IF TEST > 0 THEN PRINT "NO HAY
     PROBLEMA" ELSE PRINT "VALOR FUERA DE
     ESCALA"
1900 NEXT L
```

GOSUB suele tomar como argumento un número de línea, lo que ofrece dos inconvenientes; en primer lugar, GOSUB 1000, por ejemplo, no proporciona ninguna pista acerca del objetivo de la subrutina de la línea 1000; en segundo lugar, la especificación de números de línea hace que sea muy difícil volver a numerar el programa o mezclarlo.

GOSUB, como GOTO, es de ejecución relativamente lenta, porque cada vez que se obedezca la instrucción se ha de buscar en el programa la línea especificada.

Las funciones y los procedimientos del BASIC del BBC responden a estas objeciones. Ambas son bloques de códigos a modo de subrutinas, pero se las llama por un nombre en vez de por el número de línea, de manera que pueden estar autodocumentadas o al menos tener un significado en el listado, y no necesitan verse afectadas por subsiguientes renumeraciones o mezclas. Además, las llamadas a funciones y procedimientos por lo general se ejecutan con más rapidez que las órdenes GOSUB y GOTO.

Los procedimientos y las funciones empiezan con DEF PROC o DEF FN, seguido de un nombre, y casi siempre (pero no necesariamente) una lista de parámetros. Por ejemplo:

```
1200 DEF FNcalc(a,b,c) = (a−b)* c/100 y
2500 DEF PROCoperac (w,x$,y$,z)
```

La definición utilizará estos parámetros como si fueran variables del programa. Sin embargo, cuando el programa llama a la función o el procedimiento, los parámetros, o variables ficticias, se pueden sustituir por cualquier número de variable o expresión literal del mismo tipo de datos que el parámetro original. Por ejemplo:

```
250 resultado = FNcalc (precio, costo, 12) o
545 PROCoperar (6, nombres$, "pérez",
    matriz (12))
```

Los valores de los parámetros se emplean entonces en la definición en lugar de las variables ficticias. Observe que una función se puede utilizar en una expresión como si fuera una variable o una cantidad aritmética, mientras que una llamada a procedimiento se emplea como si fuera una orden en BASIC. La orden LOCAL impide que se produzca el error de una subrutina común:

```
100 FOR K = 1 TO 10:GOSUB 500:NEXT K:END
500 FOR K = 1 TO 5:PRINT"****";NEXT:RETURN
```

Aquí la variable K se utiliza como el contador del bucle de la línea 100 del programa principal y otra vez en la subrutina de la línea 500, un descuido que afectará a la ejecución pero que puede resultar difícil de evitar (o de rastrear). Pero en un procedimiento BBC, este peligro se puede evitar:

```
100 FOR K = 1 to 10:PROCestrellas:NEXT:END
500 DEF PROCestrellas
520 LOCAL K
540 FOR K = 1 TO 5:PRINT"****":NEXT
560 ENDPROC
```

La orden LOCAL significa que entre las líneas 500 y 560 la variable K es una nueva variable, independiente de la variable K de cualquier otro lugar del programa, y que no tiene incidencia sobre cualquier otro valor de K. (Observe que PROCestrellas es un procedimiento sin parámetros.)

REPEAT...UNTIL es una estructura de bucle en la cual la iteración continúa hasta que la expresión condicionada que sigue a la palabra UNTIL sea verdadera; el control pasa entonces a la sentencia que le sigue. Por ejemplo:

```
200 DATA 12,234,31,45,65,0,76,81
250 REPEAT
300 READ número:suma = suma + número
350 UNTIL número = 0
400 PRINT "La suma es";suma
```

Esto resulta mucho más legible y es mucho menos susceptible de error que un bucle GOTO o un bucle FOR...NEXT ficticio.

El BASIC del BBC posee las valiosas ayudas para depuración que son TRACE, ON ERROR..., y ERL. TRACE hace que se visualicen en la pantalla los números de línea del programa a medida que se van ejecutando; ON ERROR GOTO (o GOSUB) significa que cualquier error que normalmente sería fatal (incluyendo el pulsar la tecla ESCAPE) durante la ejecución del programa, haría que el control pasara a una rutina para manipulación de errores definida por el usuario (como un volcado de todos los valores de las variables). ERL es una variable del sistema que contiene el número de la línea en la cual se ha producido el error.

El BBC posee una riqueza de ampliaciones para el BASIC exclusivas, como las llamadas al sistema operativo, la orden VDU, las diversas variables del sistema y el ensamblador, que son en sí mismas suficientemente complejas como para ser objeto de comentarios sueltos que ofreceremos más adelante.

Los gráficos BBC

Las órdenes del BASIC del BBC que están directamente relacionadas con los gráficos son:

MODE
Selecciona la modalidad visualización del ordenador con MODE N, donde N = de 0 a 7:

Modalid.	Gráficos	Colores	Texto
0	640×256	2	80×32
1	320×256	4	40×32
2	160×256	16	20×32
3	sólo texto	2	80×25
4	320×256	2	40×32
5	160×256	4	20×32
6	sólo texto	2	40×25
7	Teletexto		40×25

Los ordenadores BBC Modelo A sólo pueden acceder a las modalidades 4, 5, 6 y 7; el Modelo B tiene acceso a todas las modalidades. De la modalidad 0 hasta la 6, el usuario puede modificar el juego de caracteres mediante la orden VDU. Los caracteres de teletexto de la modalidad 7 son fijos y no corresponden al código ASCII estándar.

COLOUR
Establece uno de los 16 colores para texto y fondo según la modalidad seleccionada con:

COLOUR N
donde N va de 0 a 15 para los colores de texto y de 128 a 143 para los colores del fondo. Los colores establecidos por cada color de N no son constantes de una modalidad a otra. En la guía para el usuario se ofrecen listas de los valores de N en relación a los colores para cada una de las modalidades.

VDU
Ésta es una orden sumamente útil. VDU A equivale a PRINT CHR$(A). De la misma manera, VDU A,B,C produce los mismos efectos que PRINT CHR$(A);CHR$(B);CHR$(C). Esto significa que las muchas y complicadas rutinas para texto y gráficos bajo el control de los 32 códigos CHR$, que duplican los efectos de la mayoría de las órdenes de BASIC relacionadas con los gráficos, se pueden construir con un pequeño número de órdenes VDU.

CLG
Limpia la superficie para gráficos de la pantalla corriente y desplaza el cursor a su posición "base", en el extremo inferior izquierdo de la pantalla.

CLS
Limpia la superficie para texto de la pantalla y desplaza el cursor para texto hasta su posición "base", arriba a la izquierda. También se borrará todo gráfico que hubiera en la pantalla.

DRAW
Dibuja líneas en la pantalla en las modalidades 0, 1, 2, 4 y 5. Se construye así:

```
DRAW X,Y
```

El punto definido por las coordenadas X e Y es el final de la línea. El punto de partida puede ser o bien el punto final de la última línea trazada o un punto definido mediante una orden MOVE.

GCOL
Establece los colores corrientes del fondo y primer plano de los gráficos mediante:

```
GCOL N,M
```

donde N establece cómo se ha de emplear el color (de 0 a 4) y M define el color lógico utilizando los mismos principios que COLOUR. N posee los efectos siguientes:

0 — Trazar el color especificado por M
1 — OR color M con color actual
2 — AND color M con color actual
3 — OR-exclusivo color M con color actual
4 — Invertir color actual

MOVE
Posiciona el cursor para gráficos en un punto especificado mediante:

```
MOVE X,Y
```

Tiene el mismo efecto que DRAW pero sin trazar una línea.

PLOT
Se puede utilizar para muchas funciones de gráficos, incluyendo el trazado de puntos, líneas y triángulos. Se construye así:

```
PLOT K,X,Y
```

donde K define el tipo de gráfico a trazar (PLOT). K asume valores comprendidos en la escala 0-225 para especificar el tipo de líneas a dibujar y los colores que toman de acuerdo a las listas que se proporcionan en la guía para el usuario.

POINT
Proporciona el número relacionado con el color lógico de pantalla de la coordenada de pantalla especificada, por medio de:

```
VARNUM = POINT(X,Y)
```

donde VARNUM es una variable numérica.

Monitor de memoria

Al principio puede que la numeración hexadecimal le parezca un artilugio superfluo y rebuscado, pero poco a poco se irá percatando de que es una herramienta muy útil para tratar con la memoria

Llegados a este punto del estudio en torno al lenguaje máquina vale la pena insistir en el tema de la representación de los números. Nosotros estamos familiarizados con el sistema decimal de base 10 (denominado también *sistema denario*), que es el empleado en la vida diaria, y ya sabemos algo del sistema binario. Interesa recalcar que tanto el sistema decimal como el binario no son más que expresiones alternativas del mismo concepto: el número. El ser humano, salvo casos accidentales, tiene el mismo número de dedos en cada mano. Usted dice que el número es cinco, y alguna otra persona podría llamarlo *fünf*, o *cinq*, o *pente*; pero todas esas personas están aludiendo a la misma cantidad o al mismo número: lo único diferente es el sistema oral o escrito de representación. Son representaciones diferentes pero equivalentes. Hay una correspondencia de uno a uno entre todos los números expresados en castellano y todos los números expresados en cualquier otro idioma, pues todos estos sistemas tienen la misma coherencia interna. La aritmética produce los mismos resultados independientemente del idioma que se utilice para describir los componentes individuales de una expresión aritmética.

Los distintos sistemas numéricos son exactamente iguales que los diferentes idiomas. El número de los dedos de una mano no se modifica porque se diga *five* o *cinco*, ni tampoco se altera porque se escriba 5 o 101b (recuerde que la *b* significa número escrito en sistema binario). Las dos únicas razones por las que se escoge un sistema u otro son la costumbre o la conveniencia.

Si, de niños, nos convenía aprender la representación decimal era porque se trata del sistema numérico más comúnmente utilizado a nuestro alrededor. Pero no es el único sistema. Los relojes digitales, por ejemplo, emplean un sistema de aritmética caprichoso; parte decimal, parte módulo 60 (hay 60 minutos en una hora y 60 segundos en un minuto) y parte módulo 24 (24 horas en un día). Con anterioridad a 1971, la moneda británica se contaba por unidades de 12 (peniques por chelín) y 20 (chelines por libra). Aprender a utilizar estos sistemas costaba años de angustiantes ejercicios escolares.

Cuando hablamos de ordenadores, nos resulta instructivo empezar por hablar acerca de los números binarios porque ejemplifican muy estrechamente las operaciones eléctricas del ordenador, al ser simplemente secuencias de estados encendido-apagado. Si sólo deseáramos hablar de números de un solo byte, entonces el sistema binario nos podría servir de alternativa completa al decimal: traducir de binario de ocho bits a decimal se hace sorprendentemente sencillo después de un poco de práctica. Por desgracia, las direcciones de memoria en particular y los números utilizados en general son normalmente demasiado largos como para que quepan en un byte, de modo que con el transcurso de los años los programadores y los ingenieros de ordenadores han visto la necesidad de un sistema numérico que reúna la conveniencia lógica del binario y la amplitud del decimal. Esto sólo lo ofrecen dos sistemas: el hexadecimal y el octal. El primero, ahora estándar en microinformática, familiarmente conocido como *hexa*, tiene por base el número 16. El octal, de base 8, se usó en la informática de ordenadores de unidad principal, pero se está sustituyendo por el hexadecimal.

El sistema hexadecimal

Al analizar la representación decimal y la binaria, hemos visto que la elección de la base numérica tiene dos consecuencias: la base es el número de símbolos diferentes que necesita el sistema, y es el factor multiplicativo teniendo en cuenta la posición. Por ejemplo, en decimal hay diez dígitos exclusivos (0-9) y el valor de un dígito decimal se multiplica por diez cada vez que se desplaza hacia la izquierda en un número decimal (19 no es "uno-nueve", sino "diez y nueve").

El hexadecimal, por lo mismo, exige 16 dígitos exclusivos, que, por acuerdo, son los dígitos de 0 a 9 y las letras desde la A a la F. Contar en hexa es una simple cuestión de ir nombrando los dígitos únicos y una vez agotados volver a utilizarlos en notación posicional. El número hexa que sigue a 9, por tanto, es A (decimal 10); el siguiente es B; el siguiente C; y así sucesivamente hasta F (decimal 15). Esto agota los dígitos individuales, de modo que el número que sigue a F es 10 (que se lee: "uno-cero hexa"), que corresponde al decimal 16. A partir de esto podemos ver cómo se utilizan dos dígitos individuales y que el valor de los lugares en un número hexa de varios dígitos aumenta según un factor de 16 con desplazamiento hacia la izquierda. En un número decimal, a los lugares los llamaremos: unidades, decenas, centenas y millares. De modo semejante, en un número hexa los lugares son: unidades, "dieciseisenas", "doscientas-cincuenta-y-seisenas" y "cuatro-mil-noventa-y-seisenas". Comparando los cambios en la columna binaria con los cambios en la columna hexa, usted debería ser capaz de ver la principal ventaja de los números hexas: cualquier número binario de *cuatro* bits puede expresarse con un número hexa de *un único* dígito (o sea, de 0 a 15 en decimal). Esto quedará más claro con algunos ejemplos:

Decimal	Binario	Hexa
0	00000000	0
1	00000001	1
2	00000010	2
3	00000011	3
.....		
7	00000111	7
8	00001000	8
9	00001001	9
10	00001010	A
11	00001011	B
12	00001100	C
13	00001101	D
14	00001110	E
15	00001111	F
16	00010000	10
17	00010001	11
.....		
24	00011000	18
25	00011001	19
26	00011010	1A
27	00011011	1B
.....		
31	00011111	1F
32	00100000	20
33	00100001	21

Cualquier número binario de hasta ocho bits (un byte) puede quedar reducido a dos cifras hexas. Compárense las respectivas extensiones:

de 0 a 255 en decimal
de 00000000 a 11111111 en binario
de 0 a FF en hexa

Por consiguiente, para convertir un número hexa a binario basta con transformar cada dígito hexa en un número binario de cuatro bits. Si un número de un solo byte se expresa como un número hexa de dos dígitos, entonces el dígito hexa más a la izquierda corresponde a los cuatro bits binarios situados más a la izquierda, mientras que el dígito hexa situado más a la derecha corresponde a los cuatro dígitos binarios situados más a la derecha. Si dividimos un byte de esta manera obtenemos dos *nybbles* (un nybble corresponde a medio byte). El nybble más a la izquierda, que se corresponde con el dígito hexa más a la izquierda, se denomina nybble superior o más significativo; y el nybble más a la derecha se denomina nybble inferior o menos significativo. He aquí un ejemplo:

```
                Nybble    Nybble
                Superior  Inferior
                   ↓         ↓
206     =        1100      1110    =  C E
↑                  ↑                   ↑
decimal  equivalente     binario     equiv. hexa
```

Es importante que nos familiaricemos con el sistema numérico hexadecimal, por la sencilla razón de que la manipulación de bytes de ocho bits resulta con él mucho más sencilla que utilizando el binario. Para que se convenza de ello bastará hacer un poco de práctica, no sólo con ejemplos numéricos, sino particularmente con direcciones y contenidos de bytes de memoria. Una vez comprendida su importancia (y se comprenderá muy pronto) usted se preguntará cómo le concedía tanta utilidad al decimal.

En este capítulo sobre lenguaje máquina le ofrecemos programas para el BBC Micro, el Commodore 64 y el Spectrum, que nos permiten adivinar el contenido de un byte elegido de la memoria. Estos programas "Mempeek", como los hemos llamado, primero le piden que enuncie la "dirección de comienzo" (es decir, que especifique el número del primer byte) y luego que dé el número de bytes a analizar. Si, por ejemplo, usted desea especificar el byte 1953 como su punto de partida y solicita que se visualicen los contenidos de los cuatro bytes siguientes, entonces la pantalla mostrará el número decimal 1953 en la columna de la izquierda, y luego listará los contenidos del byte 1953, byte 1954, byte 1955 y byte 1956 en las cuatro columnas siguientes.

Tenga presente que si la máquina muestra que el byte 1956 contiene el número decimal 175, lo que queremos decir es que en uno de los chips de memoria, una zona que la máquina denomina byte 1956 lleva un patrón de ocho niveles de voltaje. Si 0 voltios se representa con 0, y 5 voltios con 1, luego el byte 1956 lleva el patrón de voltaje 10101111. Éste es el que hemos escogido para interpretar como binario, y su equivalente decimal es 175.

Es de vital importancia recordar que la mayoría de las veces que hablamos de ordenadores utilizamos una especie de taquigrafía muy imprecisa, y ampliarla a una descripción física siempre es saludable y debería contribuir a evitar las confusiones.

El contenido de un byte visualizado en la pantalla no es el "real". Lo que vemos son datos de caracteres que se le han asignado a los patrones de voltaje de los bytes. Esto significa que habiendo interpretado los patrones de voltaje como números binarios, y habiendo convertido los números binarios en números decimales, estamos dando un paso más hacia adelante y convirtiendo números decimales en caracteres según el código ASCII (*American Standard Code for Information Interchange*: código norteamericano estándar para intercambio de información). Estos datos de caracteres se visualizan en la última columna de la visualización. Se trata de un código reconocido internacionalmente, que está incorporado en la mayoría de los ordenadores y que sustituye a los números decimales entre 0 y 127 para todos los caracteres de un teclado (históricamente, un teclado de teleimpresora). En este código, el número decimal 65 significa el carácter en mayúscula "A", el 66 significa "B", el 67 significa "C", y así sucesivamente. Entre los caracteres no alfabéticos, 32 significa un espacio, 42 significa un asterisco, 13 significa la tecla Return (retorno), y así sucesivamente.

Los caracteres ASCII imprimibles empiezan en el número 32 y terminan en el número 127. Los códigos fuera de esta escala no están definidos, o no son imprimibles, o son específicos de determinadas máquinas. Debido a ello, cuando ejecute los programas Mempeek, el monitor le imprimirá un punto para representar cualquier byte que contenga un número fuera de escala. En el próximo capítulo de este curso ofreceremos un amplio juego de caracteres ASCII para los valores comprendidos entre 0 y 127.

Una investigación del juego de caracteres ASCII es especialmente útil como trasfondo para una comprensión cabal del código de lenguaje máquina por dos importantes razones. En primer lugar, sirve para descubrir que la forma en que se interpreten los contenidos de la memoria es pura arbitrariedad. Se puede decir que un byte contiene un número, o

una dirección, o un carácter codificado, o una instrucción, o cualquier otra cosa que nos venga en gana. En cualquier caso, serán datos que esperan ser interpretados. En segundo lugar, proporciona una visión bastante más comprensible de la memoria, en especial de aquellas partes de la misma que realmente contienen datos de caracteres, algunos de ellos utilizados por el sistema operativo de la máquina y algunos otros empleados por el usuario.

Los datos del sistema operativo incluyen todos los mensajes de error y aviso: por ejemplo, READY, o NONSENSE IN BASIC (no significa nada en BASIC), o START TAPE THEN PRESS RETURN (accione cinta, luego pulse Return); todo cuanto sea capaz de decirle al usuario debe codificarse en ASCII y almacenarse en la memoria. Puede que no haya pensado nunca en esto, pero es una idea reveladora de las limitaciones de un ordenador como máquina "inteligente". Nuestra inteligencia es diferente: no memorizamos mensajes, sino que elaboramos un pensamiento y luego generamos una combinación adecuada de palabras para expresarlo.

Mapas de memoria

Un mapa de memoria es una representación esquemática de la distribución dada a la memoria junto con los parámetros de zonas específicas. Algunas zonas de la memoria siempre se utilizan con el mismo fin. En el Commodore 64, por ejemplo, desde el byte 0 hasta el byte 1024 los utiliza el sistema operativo de BASIC como área de trabajo. Otras zonas de la memoria poseen diversos usos según el tamaño y el estado del programa. Las fronteras entre estas zonas pueden ser fijas (en los diagramas que ofrecemos abajo aparecen como líneas continuas) o bien *flotantes* (las líneas a trazos). Las fronteras fijas no se alteran nunca, mientras que las flotantes son para aquellas áreas de la memoria que fluctúan a tenor de las necesidades. En el mapa de memoria del Commodore, las fronteras de la RAM de pantalla son fijas (en los bytes 1024 al 2048), y las de la zona de memoria donde se conservan las variables de BASIC fluctúan según la cantidad que se use en un momento dado.

Los programas Mempeek de la página siguiente se pueden utilizar para localizar las posiciones corrientes de las fronteras flotantes de la memoria de su máquina. El Commodore posee seis indicadores de fronteras flotantes (también denominados variables del sistema). En el cuadro inferior ponemos un ejemplo de cómo se emplean los contenidos de un par de bytes para calcular la dirección de memoria requerida. El BASIC del BBC posee cuatro variables del sistema para determinar y el Spectrum cinco.

Es necesario recordar que un mapa de memoria es una representación estática de algo que cambia de forma continuamente mientras se está utilizando la máquina. Cada una de las fronteras flotantes está sujeta a modificaciones en cualquier momento. Vea en la página siguiente algunas ideas para ampliar los programas Mempeek para observar las variaciones en los valores de los indicadores.

Las variables de sistema Commodore

43,44 Comienza el TEXTO DEL PROGRAMA EN BASIC
45,46 Comienzan las VARIABLES EN BASIC
47,48 Comienzan las MATRICES EN BASIC
49,50 Terminan las MATRICES EN BASIC
51,52 Parte inferior del ALMACENAMIENTO DE SERIES EN BASIC
55,56 Parte superior del ALMACENAMIENTO DE SERIES EN BASIC

Ejemplo

Utilice el programa Mempeek de p. 539 para inspeccionar los contenidos de estos bytes. Su visualización en pantalla podría tener el siguiente aspecto:

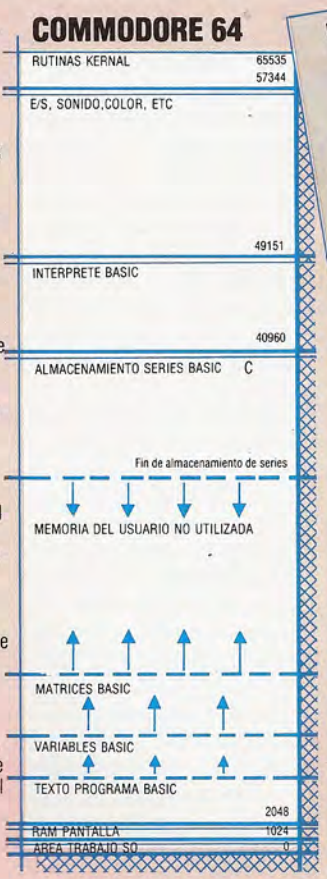
43 0 8 11 9

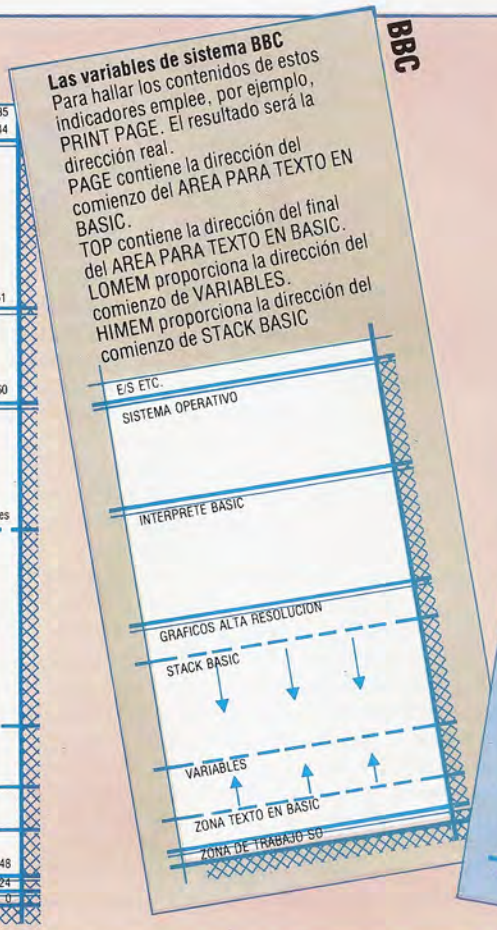
La primera columna es la dirección del primer byte al cual se ha accedido. La segunda y tercera columnas visualizan los contenidos del byte 43 y byte 44. Éstos son los bytes de desplazamiento y de página (véase p. 516) de la dirección de comienzo del área de texto en BASIC. Ésta se calcula así:

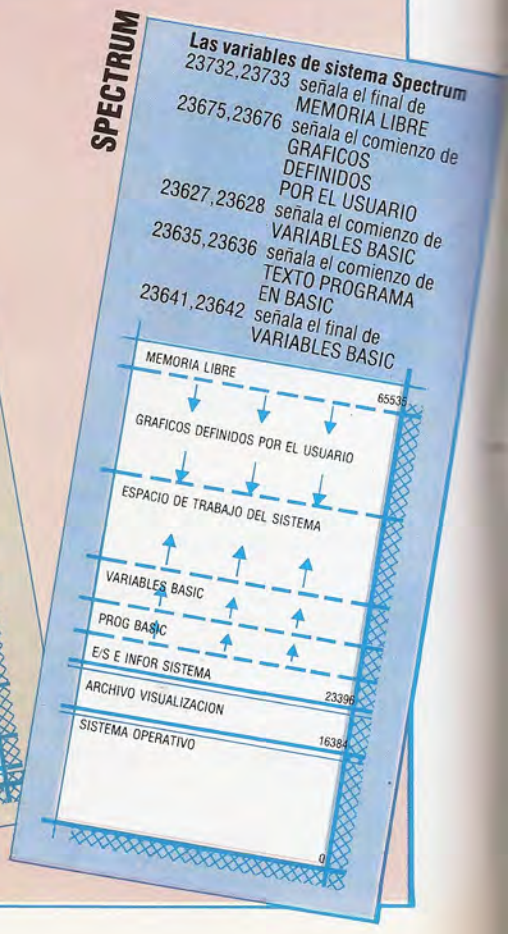
8*256 + 0 = 2048

La cuarta y quinta columnas de la visualización son los bytes de desplazamiento y página para el final del área de texto en BASIC. La dirección se calcula así:

9*256+11 = 2315







BBC MICRO

```
7 REM*************************************
8 REM*          MEMPEEK 1          BBC     *
9 REM*************************************
20 MODE 7
30 *TV 255
40 CLS
50 REPEAT
100 INPUT"DIRECCION DE COMIENZO ",DC
200 INPUT"NUMERO DE BYTES (0 PARA
    SALIR) ",NB
250 PRINT "*******************************
    **********"
300 FOR B%=DC TO (DC+NB-1) STEP 4
350 H$="":C%=6
400 PRINT TAB(0);B%;TAB(8);
450 C% =4
500 FOR C=0 TO 3
550 PK%=? (B%+C):PK$="."
600 PRINT PK%;
650 IF PK%=13 THEN PK$=CHR$(124)
700 IF (PK%>31) AND (PK%<128) THEN
    PK$=CHR$(PK%)
750 H$=H$+PK$
800 NEXT C
850 PRINT TAB(32);H$
900 NEXT B%
950 UNTIL NB=0
1000 REM*************************
```

Spectrum

```
7 REM*************************************
8 REM*          MEMPEEK 1 SPECTRUM      *
9 REM*************************************
30 DIM H$(4)
50 FOR L=0 TO 1 STEP 0
100 INPUT"DIRECCION DE COMIENZO ";DC
200 INPUT"NUMERO DE BYTES (0 PARA
    SALIR) ";NB
250 PRINT "*************************
    *****"
300 FOR B=DC TO (DC+NB-1) STEP 4
350 LET H$="...."
400 PRINT B;TAB 7;
500 FOR C=0 TO 3
550 LET PK=PEEK(B+C)
600 PRINT PK;" ";
650 IF (PK> 31) AND (PK< 128) THEN
    LET H$(C+1)=CHR$ PK
700 IF PK=13 THEN LET H$(C+1)="■"
800 NEXT C
850 PRINT TAB 26;H$
900 NEXT B
950 IF NB=0 THEN LET L=2
1000 NEXT L
1050 REM*************************
```

Use los Mempeek

Cuando dé entrada al programa Mempeek en su máquina, asegúrese de guardarlo (SAVE) y verifíquelo concienzudamente antes de ejecutarlo (RUN), porque en esta clase de programas los errores de digitación pueden producir roturas irreparables.

Primero el programa le solicitará una dirección de comienzo y después el número de bytes que usted desea examinar. Ambos deben ser números enteros positivos comprendidos entre 0 y 65535. Dar entrada a 0 como el número de bytes hará que el programa termine (salida). Supongamos que introduce como dirección de comienzo el byte 230. La visualización en pantalla podría asumir el siguiente aspecto:

```
DIRECCION COMIENZO? 230
NUMERO DE BYTES (0 PARA SALIR)? 8
--------------------------------

230  193  32  65  49   . A1
234  129  64  93  98   .@]b

DIRECCION DE COMIENZO?
```

La columna situada más a la izquierda proporciona la dirección decimal del primer byte, las cuatro columnas siguientes proporcionan los contenidos decimales de los cuatro bytes a partir de dicha dirección en adelante, y la última columna proporciona la representación en caracteres de los contenidos de los bytes, cuando esto sea posible; de lo contrario, dará '.'

Puede que opte por "vagabundear" con este programa a lo largo y ancho de la memoria, tomando nota de cualquier dirección interesante, y trate luego de hallar en qué lugar de la memoria almacena el sistema operativo sus mensajes de error y las palabras clave en BASIC. Su manual para el usuario podrá serle de ayuda en esto.

Una vez hallados los indicadores que definen las fronteras de las diversas zonas de la memoria, puede tratar de agregarle al programa algunas líneas REM y ver el efecto que ello tiene en los valores de los indicadores. Después agregue algunas líneas al principio del programa para hacer alguna manipulación de series y, nuevamente, vea el efecto que produce en los indicadores y en los contenidos del área de almacenamiento de variables.

Por ejemplo:

```
3 DIM Z$(254)
4 LET X$ = ""
5 FOR M = 1 TO 255:LET X$ = X$ + "*":NEXT M
```

Commodore 64

```
7 REM*************************************
8 REM*     MEMPEEK 1 COMMODORE          *
9 REM*************************************
30 PRINT CHR$(147)     :REM LIMPIAR
9 FPANTALLA
40 PRINT CHR$(142)     :REM MAYUSCULAS
50 FOR LP=0 TO 1 STEP 0
100 INPUT"DIRECCION DE COMIENZO ";DC
200 INPUT"NUMERO DE BYTES (0 PARA
    SALIR) ";NB
250 PRINT "*************************
    **************"
300 FOR B=DC TO (DC+NB-1) STEP 4
350 H$=""
400 PRINT B;TAB(8);
500 FOR C=0 TO 3
550 PK=PEEK(B+C):PK$="."
600 PRINT TAB(8+5*C);PK;
650 IF PK=0 THEN PK$=CHR$(122)
700 IF (PK> 31) AND (PK< 128) THEN
    PK$=CHR$(PK)
750 H$=H$+PK$
800 NEXT C
850 PRINT TAB(32);H$
900 NEXT B
950 IF NB=0 THEN LP=1
1000 NEXT LP
1050 REM*************************
```

De la bellota a la encina

Acorn Computers produce dos excelentes ordenadores personales: el BBC Micro y el Electron

Chris Curry

Herman Hauser

Cortesía de Acorn

El fundador de Acorn, Chris Curry, era un empleado y amigo de sir Clive Sinclair. Curry se había incorporado a la Sinclair Radionics en 1965, cuando Sinclair le ofreció un puesto como ingeniero de proyectos nuevos con un estipendio semanal de 11 libras.

En la Sinclair Radionics, Curry se hizo cargo del proyecto de investigación que en 1971 produjo la calculadora Executive. Durante los cinco años siguientes, se dedicó a desarrollar calculadoras, de factura tal que hoy son consideradas como las precursoras del moderno ordenador personal. En 1975, la Sinclair Radionics dejó de producir y Curry se unió a Sinclair en una operación independiente llamada Science of Cambridge. Esta nueva empresa se proponía reunir los componentes electrónicos en conjuntos de piezas para montar y venderlos como *kits*.

Una idea de mucha aceptación fue el reloj de pulsera con calculadora. Pero Curry también se sintió atraído por los ordenadores de un solo tablero que comenzaban a surgir en Estados Unidos, y se propuso crear su propio modelo desmontable. Se llamó MK14 (microprocesador en forma de kit de 14 chips) y constaba de un microprocesador National Semiconductor, con 256 bytes de RAM, de una pequeña memoria fija que contenía el monitor, y de los componentes necesarios para la alimentación eléctrica de una pantalla LED (*Light Emitting Diode*: diodo emisor de luz) de ocho dígitos.

Cortesía de Acorn

Curry comprobó que la empresa constantemente estaba proporcionando consejos e ideas por teléfono a aficionados a la electrónica, y decidió proponer a Herman Hauser, que estudiaba el doctorado en filosofía en la Universidad de Cambridge, para que atendiera estas consultas. Pronto, sin embargo, los planteamientos de Curry empezaron a disentir de los de Sinclair, y pensó que bien podía ser hora de crear una empresa propia. Con Hauser como nuevo socio, Curry formó una empresa llamada Cambridge Processor Unit (un nombre escogido con picardía, ¡ya que sus siglas eran CPU!). Desde un pequeño despacho situado en Bridge Street (Cambridge), ambos ofrecían sus servicios como consultores en electrónica y ordenadores.

El éxito del MK14 y las novedades llegadas de Estados Unidos demostraban claramente que lo que deseaban los clientes era un ordenador dentro de una caja con BASIC en el tablero. Ya que habían escrito una versión rápida de BASIC para control de la máquina en uno de sus trabajos de consulta, CPU decidió incorporarla a una máquina y sacarla al mercado. La máquina recibió el nombre de Atom (átomo) y CPU adoptó Acorn (bellota) como nombre comercial para la empresa encargada de su lanzamiento al mercado. Se pretendía que la máquina captara sobre todo el mercado educativo, pero la mayoría de las escuelas creían que su BASIC se apartaba demasiado de la versión Microsoft como para que fuera aceptable. No obstante, la máquina fue recibida con mucho entusiasmo por parte de los aficionados. Acorn siguió adelante con una nueva versión del Atom, que recibió el nombre de Proton (protón), destinado a su utilización en laboratorios y colegios.

Pero en 1981, con el Proton en fase de prefabricado, Curry se enteró de que la BBC buscaba una máquina para su programa de alfabetización informática. Curry demostró a la BBC las capacidades del procesador 6502, pero no en el Proton sino en un sistema diseñado especialmente.

Las indicaciones de la BBC aludían a una máquina fácil de manejar por los principiantes pero que se pudiera ampliar para conformar un estándar muy elevado. La máquina debía ofrecer además una buena relación calidad-precio (la compañía de radiotelevisión británica inicialmente hablaba de un precio orientativo de 200 libras). Con la oposición de Sinclair Research, el trabajo se encomendó a Acorn, que creó el BBC Micro, del que hoy se produce al mes un promedio de 12 000 unidades.

Programas para estudiantes

El software educativo suele basarse en el método de preguntas y respuestas, siendo las primeras muy directas y estas últimas precisas y nada ambiguas

Se espera que dentro de poco la mayor parte de las escuelas españolas puedan contar con un ordenador. No obstante, tal vez sea Gran Bretaña el país que se encuentre más avanzado en el campo de la educación asistida por ordenador (EAO) y, en consecuencia, donde la creación de software de carácter pedagógico está a la vanguardia tanto en calidad como en cantidad. En la actualidad es posible hallar en el mercado británico numerosos paquetes de repaso, que suelen exigir un mínimo esfuerzo por parte del programador. La tarea es sencilla, sólo es cuestión de presentar el texto y calificar la respuesta dada por el usuario. Una vez que se establece este procedimiento, se puede utilizar la misma estructura para dar cabida a una amplia variedad de textos relativos a diversas materias y rutinas pregunta-respuesta, con el objeto de complementar o reemplazar el libro de texto. Aunque esta clase de programas son eficaces, se debe admitir que son más bien monótonos. No obstante, no cabe duda de que las futuras generaciones de software de repaso de exámenes harán uso de las facilidades para gráficos y color de que ahora disponen la mayoría de los ordenadores personales para representar visualmente el material que ilustra el texto.

Idiomas

Para los idiomas que se suelen incluir en los planes de enseñanza existe bastante software de repaso destinado a diversas máquinas, incluyendo las ya utilizadas en las escuelas, como el BBC Micro y el ZX Spectrum, y aquellas que por lo general se utilizan más para software de juegos, como sucede en el caso de la gama Atari.

En Gran Bretaña, Kosmos Software ofrece un grupo de este tipo tanto para el Spectrum como para el BBC Modelo B en cassette; *The French mistress* (La maestra de francés), *The German master* (El maestro de alemán) y *The Spanish tutor* (El preceptor de español) son similares en cuanto a su método operativo, y los tres vienen en dos niveles. El nivel A se compone de listas de palabras o frases cortas (de hasta 59 caracteres de longitud) en inglés y en el idioma de que se trate, por categorías como "familia", "alimentos", "seres vivos" y otras similares; el cassette del nivel B abarca adjetivos y adverbios, conjugaciones y tiempos verbales.

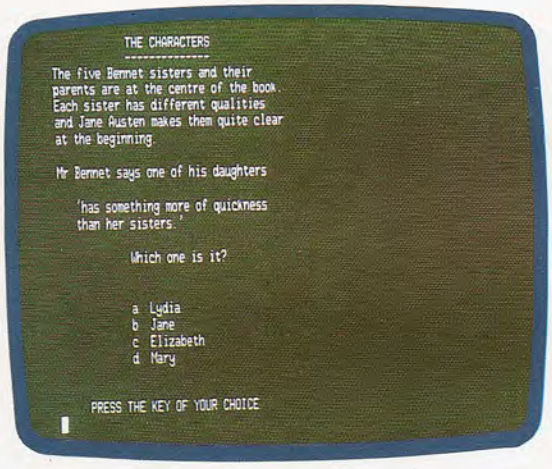
El usuario tiene la opción de trabajar partiendo del idioma extranjero y traducirlo a su lengua, o viceversa, así como de crear listas de palabras propias, que se pueden guardar en cassette para posterior consulta. Cada uno de los cassettes está dividido en 16 lecciones, siendo la longitud máxima de cada una de ellas de 250 términos. Este límite también rige para las listas creadas por el usuario. Éste puede elegir entre la modalidad aprendizaje, en la cual se visualiza una palabra extranjera y luego su equivalente en lengua propia; un autotest, en el que sólo se visualiza uno de los idiomas, y una prueba de exactitud con tiempo medido.

Cortesía de Pilot Software

Inglés

Los interesados en aprender inglés tienen a su disposición un variado software. *English language*

(Lengua inglesa) de Commodore, por ejemplo, está a la venta en España en versión de disco y se basa en material proporcionado por International Correspondence Schools. Forma parte de una serie más amplia para ordenadores Vic-20 ampliados. Se centra de manera primordial en la gramática y en el empleo correcto de las palabras. El menú principal ofrece elegir entre composición (conjunto de ejercicios sobre definición y utilización de las palabras, en que se formulan preguntas tipo test, cuya respuesta correcta está incluida entre otras opciones), ortografía, gramática y comprensión. El paquete se complementa con un manual que explica el método operativo mediante ejemplos adecuados y que también proporciona textos para la realización de los ejercicios de síntesis y comprensión.



Literatura inglesa

Lamentablemente para los creadores de software, los programas de estudio de literatura inglesa tanto en el nivel medio como en el avanzado se basan en obras concretas (y la lista de títulos es larga).

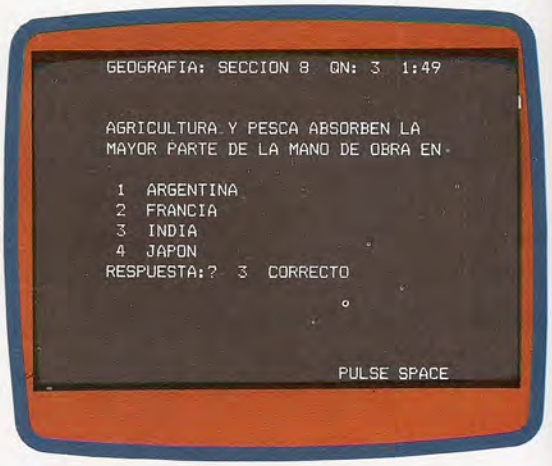
Pride and prejudice (Orgullo y prejuicio), la clásica novela de Jane Austen, producida por Sussex Software, constituye un excelente ejemplo del ingente esfuerzo que supone la elaboración de un paquete de este tipo. Escrito para el Research Machines 380Z (y, por consiguiente, disponible, con algunas modificaciones, para cualquier otro ordenador basado en CP/M) y la serie Commodore 3000 y 4000, el paquete se entrega en tres discos y se centra en los personajes, los temas y el argumento de la novela, así como en su contexto social.

Aunque Sussex Software se ha visto obligada a utilizar el cuestionario tipo test para evaluar el conocimiento adquirido por el alumno, cada pregunta viene reforzada por una explicación exhaustiva, y se ofrecen con la mayor profundidad posible las razones que sustentan cada respuesta.

Historia y geografía

La historia, la geografía y la economía requieren la interpretación de hechos concretos y, por lo tanto, son materias que se prestan muy bien al repaso asistido por ordenador.

En España, Commodore ha editado varios programas para facilitar el estudio de la Historia universal. En ellos se analizan la Edad Antigua y la



Edad Media. En cuanto a Geografía, esta firma cuenta con programas tipo test mediante los cuales se pueden estudiar las capitales de los países asiáticos y africanos, pudiéndose modificar por el propio educador para adaptarse a otros continentes. Otro programa permite cargar mapas de diferentes zonas del mundo y enterarse dónde están localizadas las diferentes capitales. Todos estos programas están presentados en disco y en versión castellana. Atari, por su parte, dispone de programas para estudiar geografía general (dibujos de mapas del mundo y banderas de Europa).

Matemáticas y ciencias

Algunas casas distribuidoras de las grandes firmas de ordenadores personales ya han confeccionado catálogos en castellano del software disponible para el aprendizaje y la práctica de las ciencias. Así, para el Vic-20 de Commodore existe, en cinta y en disco, un programa específico de matemáticas para BUP con siete cuestionarios que cubren temas de geometría, aritmética, álgebra, etc.

Atari, por su parte, ha puesto a disposición de los alumnos de Educación General Básica y de Bachillerato dos cintas o discos para el estudio de la física y de la química. Sería interesante disponer de la biblioteca en inglés Rose Software para el Spectrum o la de SciCAL para el BBC Micro.

Son también dignos de nota los libros que comienzan a venderse en librerías con programas escritos especialmente en BASIC que dan solución a ciertos problemas y ejercicios de ciencias.

Liz Heaney

Un sistema clásico

A pesar de tener una capacidad de almacenamiento y una velocidad de operación reducidas, la tradicional unidad de disco Atari 810 cuenta con una interesante gama de órdenes

Cuando se lanzaron los ordenadores Atari 400 y 800, sin duda fueron las primeras máquinas diseñadas específicamente para su utilización personal. Casi de inmediato Atari produjo el sistema de unidad de disco simple 810 para aumentar las capacidades de los ordenadores, pero debido al elevado precio de cada unidad el sistema no consiguió hacerse popular. Aun así, con la llegada del Atari 600 compatible, es conveniente volver a analizar la unidad de disco 810.

Esta unidad utiliza discos flexibles de 5 ¼ pulgadas, de densidad simple y de una sola cara. Se conecta al ordenador a través de la puerta especial de E/S en paralelo. Se pueden conectar en "cadena margarita" hasta cuatro unidades 810, numeradas de 1 a 4 conforme se establezca un interruptor de código para unidades situado en la parte posterior de cada unidad de disco. Aunque la 810 contiene su propio microprocesador, no se puede decir que sea una auténtica unidad inteligente, porque parte del DOS II de Atari (el sistema operativo en disco de que va provista normalmente) se debe cargar en RAM para poder acceder a la unidad de disco. El DOS ocupa aproximadamente cinco Kbytes de la RAM para el usuario, dejando apenas ocho Kbytes de RAM libres en aquellas máquinas de 16 Kbytes como la 600. Sin embargo, en un futuro cercano habrá tableros de RAM de 48 Kbytes para el 600, que incrementarán la RAM a 64 Kbytes.

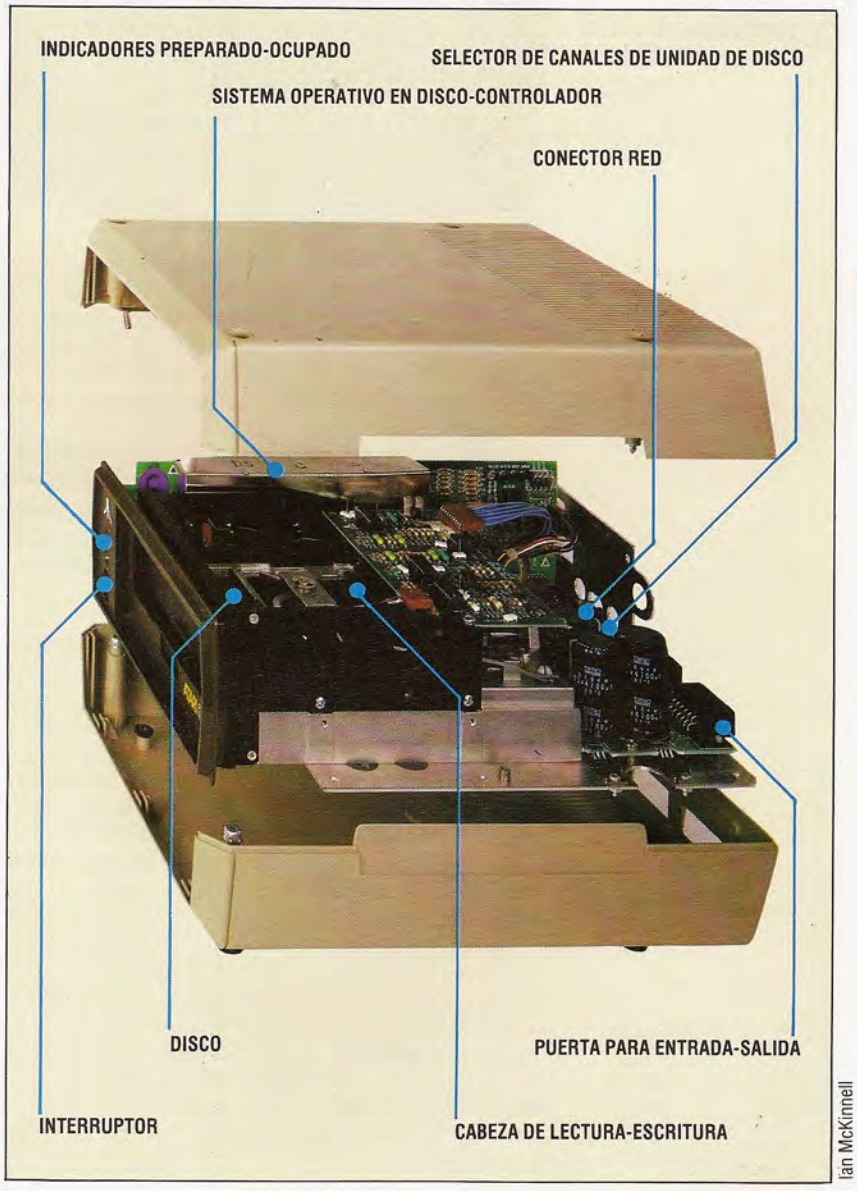
El DOS de Atari se proporciona con la unidad de disco en un disco maestro y consta de tres archivos separados pero relacionados entre sí. Éstos son: DOS.SYS, que contiene el sistema de administración de archivos y las instrucciones de órdenes residentes en la RAM mientras se utilicen; DUP.SYS, un archivo de utilidades en disco que retiene el menú del DOS y algunas instrucciones de órdenes para el DOS; y AUTORUN.SYS, que aloja un archivo que, mediante una orden, se carga automáticamente en la RAM y se ejecuta para llamar al menú del DOS y a las porciones del DOS residentes en la RAM.

Para acceder al DOS con el cartucho de BASIC insertado, se debe encender la unidad de disco antes de darle potencia al ordenador y antes de introducir el disco maestro. Luego, al encender el ordenador (puesto en "on"), éste "calza" (o, más específicamente, carga) parte del DOS.SYX en la RAM. Para llamar al menú DOS de 15 opciones, se debe digitar DOS (RETURN).

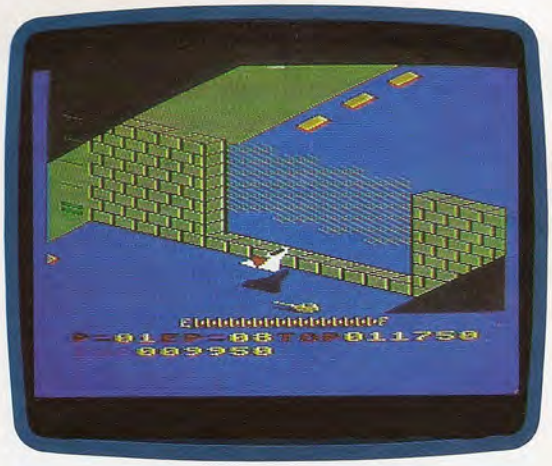
El menú del DOS proporciona 15 opciones de órdenes para administración del disco que se seleccionan mediante las letras de la A a la O. El sistema de menú ofrece un sencillo método para administrar el disco, pero tiene una importante desventaja y es que escribe sobre una porción de la memoria para el usuario, destruyendo cualquier programa o datos que estuvieran almacenados. Esto significa que un programa que se esté editando o escribiendo se debe guardar en disco antes de llamar al menú y volver a cargarlo luego cuando se termina con el menú de opciones. El DOS posee una facilidad en virtud de la cual esto se puede realizar de forma automática si así se solicita, pero aun así esto continúa siendo incómodo.

El DOS incluye una verificación automática de los datos almacenados a cada orden de escribir, lo que limita la velocidad de transferencia de datos a

La unidad Atari 810
Los principales inconvenientes de esta unidad de disco son su interface en serie, lo que hace que la unidad funcione bastante lentamente, y su limitada capacidad de almacenamiento, 88 Kbytes por disco. No obstante, un sistema operativo en disco de relativa complejidad contribuye a atenuar estas desventajas



Ian McKinnell



En el interior de la cabina
Zaxxon es sólo uno de los juegos pertenecientes a la gama que con tanto éxito ha producido Atari para sus ordenadores personales. El jugador es el piloto de un caza a reacción que, en pugna con otros aviones, efectúa un ataque a tierra. La pantalla simula el parabrisas del avión

Blue Max
Un segundo juego de combate entre aviones de caza producido por Atari sitúa al jugador en otros tiempos. *Blue Max*, de venta en disco, cassette o cartucho, lo sitúa a uno al mando de un avión de caza biplano de la primera guerra mundial. Aunque no pretende ser un simulador de vuelo, el punto fuerte del juego es la calidad de sus gráficos

Cortesía de Soft

2,4 Kbytes por segundo. Si se necesitara una mayor velocidad, cabe suprimir la verificación mediante la orden POKE 1913,80, con lo que se logra una velocidad de transferencia de datos de 4,8 Kbytes por segundo. POKE 1913,87 restaura la verificación.

La orden FORMAT DISK da formato a un disco vacío colocado en una unidad seleccionada, dividiéndolo en 40 pistas, cada una de ellas dividida a su vez en 18 sectores capaces de almacenar 128 Kbytes de datos cada uno, reservando tres bytes para el uso del sistema de administración de archivos. El DOS destina ocho sectores para el directorio del disco y un sector para la tabla del contenido de volumen (*Volume Table of Contents*: VTOC), equivalente del BAM, y reserva cuatro sectores para utilización del DOS, dando un total de 707 sectores × 125 bytes, o sea, 88 375 bytes (alrededor de 86 Kbytes) disponibles para almacenamiento.

Las órdenes estándar LOAD, SAVE y las órdenes en BASIC relacionadas se pueden utilizar para almacenar tanto archivos de programas como de datos especificando la unidad de disco de almacenamiento. También se pueden almacenar archivos cuya lectura sea secuencial o directa, a un byte por vez.

El sistema de unidad de disco 810 se diseñó hace al menos cinco años y, si bien el DOS II de Atari posee muchas configuraciones eficaces y exclusivas, actualmente el sistema ha quedado anticuado. La bajísima capacidad de 86 Kbytes, la pérdida de unos preciosos ocho Kbytes de memoria para el usuario y el elevado precio de venta se combinan para hacer que la relación calidad-precio de la 810 sea discutible.

Órdenes del disco Atari

Para la manipulación de archivos por lo general se utiliza una especificación de archivos estandarizada. Ésta es:

COMMAND "DN:NOMARCHI.EXT"

donde COMMAND es la orden DOS; N es el número de la unidad de disco (opcional 1-4 en unidad única); el NOMARCHI (nombre del archivo) contiene hasta ocho caracteres para identificar al archivo (el primero debe ser alfabético); y .EXT es el ampliador opcional de archivo (se puede emplear para indicar el tipo de datos almacenados).

Cuando se necesita esta especificación de archivo, se alude a ella como FSP. También se pueden emplear *wildcards* (tarjetas no especialmente preparadas) con algunas órdenes en las que se puede utilizar "?" para sustituir un único carácter y '*' puede reemplazar cualquier número de caracteres válidos después del '*'.

Para seleccionar una opción del menú DOS digite la letra adecuada y pulse (RETURN). En todos los casos FNM significa D1:NOMARCHI.EXT.

A. DISK DIRECTORY (directorio del disco)
(RETURN) visualiza una lista de todos los nombres de los archivos que contiene el disco de la unidad 1, el ampliador y el número de sectores de cada archivo.

B. RUN CARTRIDGE (ejecutar cartucho)
Esta opción devuelve el control del ordenador al cartucho insertado, por lo general BASIC.

C. COPY FILE (copiar archivo)
Copia un archivo de un disco a otro disco o bien al mismo bajo un nombre distinto. Por ejemplo esta línea,

D1:NOMARCHI.EXT,D2:NOMARCHI.EXT

copia NOMARCHI.EXT de la unidad de disco 1 a la 2. Del mismo modo:

D1:NOMARCHI.EXT,D1:NOMARCHI.BAK

hace una copia de seguridad de un archivo en el mismo disco. El nombre del archivo debe diferir en algo y en este caso se cambia el ampliador.

D. DELETE FILE(S) (borrar archivo-s)
FNM (RETURN) elimina uno o varios archivos especificados. Si se han de eliminar todos los archivos use '*' en lugar de nombre de archivo y ampliador.

E. RENAME FILE (cambiar nombre de archivo)
Cambia el nombre de un archivo especificado y actualiza el directorio. Por ejemplo:

D1:NOMANTI NOMACTU

Se pueden utilizar otras tarjetas *(wildcards)* para cambiar los ampliadores de un grupo de archivos.

F. LOCK FILE (proteger archivo)
FNM (RETURN) protege mediante software un archivo contra toda modificación o borrado mientras no se vuelva a dar formato al disco. Los archivos protegidos de esta manera van precedidos por '*' en el directorio visualizado.

G. UNLOCK FILE (liberar archivo)
FNM (RETURN) libera el archivo especificado protegido anteriormente o todos los nombres de archivos en cuestión si se los utiliza con otras tarjetas *wildcards*.

H. WRITE DOS FILES (escribir archivos DOS)
Siga las instrucciones visualizadas para guardar el DOS en un disco formateado.
I. FORMAT DISK (dar formato a un disco)
Siga las instrucciones para dar formato a un disco.
J. DUPLICATE DISK (duplicar disco)
Siga las instrucciones para copiar un disco entero de una unidad a otra o, copiando el contenido del disco en la memoria, en la misma unidad.
K. BINARY SAVE (guardar en binario)
FNM, CCCC, FFFF (RETURN) guarda el contenido de una zona especificada de la memoria, por lo general un programa en código de lenguaje máquina. CCCC es la dirección de comienzo y FFFF la dirección de final de un hexadecimal de cuatro dígitos.
L. BINARY LOAD (cargar en binario)
FNM (RETURN) vuelve a cargar un archivo guardado mediante BINARY SAVE en las posiciones en que estaba almacenado.
M. RUN AT ADDRESS (ejecutar en dirección)
Cuando se le solicite en la visualización, dé entrada a la dirección en la cual ejecutar un archivo cargado mediante BINARY LOAD en un hexadecimal de cuatro dígitos. (RETURN) ejecuta el programa.
N. CREATE MEM.SAV (crear MEM.SAV)
Siga las instrucciones para crear un archivo denominado MEM.SAV. Cuando se llama al menú DOS, el DOS guarda automáticamente el contenido de la memoria sobre el cual escribirá el menú y lo vuelve a cargar cuando se seleccione RUN CARTRIDGE.
O. DUPLICATE FILE (duplicar archivo)
FNM (RETURN), y a continuación siga las instrucciones necesarias para copiar archivos de un disco a otro en una única unidad. Se permiten otras tarjetas *(wildcards)*.

Las otras órdenes que controlan los archivos de programas y los archivos de datos son:

SAVE LOAD LIST ENTER RUN OPEN# CLOSE# PRINT# INPUT# NOTE# POINT# PUT# GET# STATUS# X10

Archivos de programas:
SAVE FSP guardar (FSP)
Escribe el programa especificado en disco en forma 'distintivada'.
LOAD FSP (cargar FSP)
Lee el programa distintivado especificado en la memoria del usuario desde la parte inferior de la memoria hacia arriba.
LIST ESP,LN1,LN2 (listar FSP,LN1,LN2)
Almacena un programa en BASIC en ATASCII (versión Atari del ASCII estándar). Especificando sólo FSP se almacena el programa entero. LN1 y LN2 representan números de línea y se pueden utilizar para especificar los números de línea inicial y final de un programa a almacenar. Se emplea conjuntamente con ENTER para mezclar programas.
ENTER FSP (dar entrada a FSP)
Lee el archivo especificado, previamente almacenado mediante LIST, en la memoria del usuario y lo mezcla con un programa ya retenido en la memoria. De existir números de línea repetidos, las líneas contenidas en el archivo recién leído ocupan el lugar de las de la memoria.
RUN FSP (ejecutar FSP)
Lee en la memoria el archivo distintivado especificado y lo ejecuta automáticamente.

Archivos de datos:
OPEN#
Esta orden controla el acceso a canales de comunicación especiales denominados *bloques de control de entrada-salida (Input/Output Control Blocks:* IOCB) y los conecta con un dispositivo apropiado, en este caso una unidad de disco y FSP:

OPEN#IOCB,AC1,AC2,FSP

donde IOCB es el canal de E/S (1-5); AC1 es el código auxiliar 1 (especifica el tipo de operación de E/S de acuerdo a una tabla que figura en el manual del DOS); y AC2 siempre es 0 para unidad de disco.

Las siguientes órdenes están relacionadas con bloques de control de E/S abiertos (OPEN) de la forma que hemos explicado arriba.

CLOSE#IOCB (cerrar IOCB)
Libera al IOCB especificado de las condiciones de E/S establecidas arriba. A los IOCB cerrados (CLOSE) no se puede acceder.
PRINT# (imprimir)
Escribe datos numéricos (X,Y) o en serie (A$) en el IOCB; y así:

PRINT#,X,Y o PRINT#,IOCB,A$

INPUT#
Lee datos numéricos o en serie del IOCB especificado; por ejemplo:

INPUT#IOCB,X,Y o INPUT#IOCB,A$

NOTE#
Se establece antes de almacenar datos mediante PRINT#. Proporciona un registro del número de sector y número de byte donde se va a almacenar el siguiente byte en disco. La lista resultante se puede almacenar como una tabla en otro archivo para el acceso a los datos al azar, de a un byte por vez si fuera necesario, mediante POINT#. Por ejemplo:

NOTE#IOCB,S,B

donde S es el número de sector (1-719) y B es el número del byte (0-124).

POINT#
Lee el byte especificado de los datos, almacenado mediante NOTE#, en la memoria para el usuario:

POINT#IOCB,S,B

PUT#
Escribe un único byte en el IOCB. Por ejemplo:

PUT#IOCB,N

donde N = de 1 a 255.

GET
Lee un único byte almacenado por PUT# con:

GET#IOCB,N

STATUS#IOCB,ERROR
Proporciona una variable especificada, en este caso ERROR, el número corriente de errores para la última operación de E/S en el IOCB. El número resultante se puede entonces verificar mediante la tabla de errores del manual del DOS de Atari.
X10 CN,#IOCB,AC1,AC2,FSB
Se utiliza para duplicar algunas de las funciones del menú del DOS mediante el empleo del número de orden relativo a la función requerida.

Precios en picado

Tomando como base 1973, el costo de un ordenador se ha reducido a la quingentésima parte por cada bit de memoria. Si un coche Rolls Royce hubiera experimentado el mismo ritmo de fabricación abaratada que el ordenador (alcanzando un nivel similar en cuanto a reducción de precio), hoy costaría 500 pesetas y su rendimiento rondaría los 300 000 km por litro de gasolina

Lógica del programador

Los operadores lógicos AND y OR son valiosas herramientas que se utilizan tanto en las instrucciones de lenguaje máquina como en la mayor parte de las versiones de BASIC

Estos dos operadores lógicos se emplean de varias formas tanto en código de lenguaje máquina como en BASIC. Un ejemplo familiar del uso de AND y OR es la relación de dos instrucciones dentro de una sentencia condicionada. Por ejemplo, diga cuál será el resultado de este programa en BASIC:

```
10 FOR I = 1 TO 5
20 FOR J = 1 TO 5
30 IF I = 3 AND J = 2 THEN PRINT I,J
40 NEXT J
50 NEXT I
60 END
```

El programa ejecutará el par de bucles anidados pero sólo imprimirá los valores de I y J si I = 3 y J = 2. Por lo tanto, este programa imprimirá en pantalla el siguiente resultado:

```
3 2
```

OR se puede utilizar de forma similar. Si corregimos la línea 30 para que se lea:

```
30 IF I = 3 AND J = 2 OR J = 4 THEN PRINT I,J
```

Se producirá la siguiente salida:

```
1 4
2 4
3 2
3 4
4 4
5 4
```

El ordenador lleva a cabo la operación AND con prioridad sobre la operación OR. I y J se imprimirán si I = 3 y J = 2, o bien siempre que J = 4. El orden de prioridad se puede modificar mediante la utilización de paréntesis. Adivine cuál será la salida del programa si se modifica así la línea 30:

```
30 IF I = 3 AND (J = 2 OR J = 4) THEN PRINT I,J
```

Teclado operado con la nariz

El sistema MATE (*Memory Assisted Terminal Equipment*: equipo de terminal asistido por memoria) se desarrolló en 1978 en la Universidad de Essex en un ordenador Vector 111. El programador era un espástico agudo, que manipulaba el teclado con la nariz.
MATE posee un teclado especial para producir algunos caracteres que no requiere mantener oprimida una tecla de cambio mientras se pulsa otra tecla, y almacena una base de datos de palabras a las que se puede dar entrada con la pulsación de una sola tecla

Aislar bits en un registro

Muchos ordenadores personales utilizan registros especiales para controlar diversas funciones de la máquina. Cada bit de un registro de este tipo puede controlar un aspecto diferente de dicha operación. Por ejemplo, en el Commodore 64 hay un registro de ocho bits que controla el encendido y apagado de los sprites. Cada uno de los bits del registro está relacionado con uno de los ocho sprites disponibles. Si alguno de los bits del registro está en 1, se hace visible en la pantalla el sprite controlado por ese bit. Si el bit está en 0, entonces el sprite se apaga y no se puede ver. Utilizando el BASIC, encender cualquier combinación de sprites es una cuestión sencilla que implica calcular el número binario de ocho bits requerido y colocar (POKE) su equivalente decimal en el registro. Sin embargo, este método no tiene en cuenta el estado del registro antes de la orden POKE y podrían apagarse sprites que previamente estuvieran encendidos. La solución del problema consiste en desarrollar una técnica que nos permita aislar los bits que se han de cambiar sin alterar ninguno de los otros.

Con el fin de ilustrar esta técnica, vamos a suponer que originalmente los sprites 0, 1, 5 y 6 están encendidos. El registro que controla la interrupción tendrá el siguiente aspecto:

Número de sprite	7	6	5	4	3	2	1	0
Bit correspondiente	0	1	1	0	0	0	1	1

Vamos ahora a encender el sprite 4, leyendo (PEEK) el registro, pero cuidando de no apagar los sprites 0,1,5 y 6 ya encendidos: enfrentaremos con un OR el byte original (01100011) y este otro (00010000), en decimal 16, que corresponde al sprite 4. Basta después colocar (POKE) el resultado:

Byte original	0	1	1	0	0	0	1	1
OR byte...	0	0	0	1	0	0	0	0
Da nuevo byte	0	1	1	1	0	0	1	1

Utilizando la orden en BASIC POKE reg, PEEK(reg) OR16 podemos ahora encender sin miedo el bit 4 en el registro. Para volver a poner en *off* el bit 4 debemos leer (PEEK) el registro y relacionar el nuevo byte (01110011) y este otro (11101111), decimal 239, con un AND:

Nuevo byte	0	1	1	1	0	0	1	1
AND	1	1	1	0	1	1	1	1
Byte original	0	1	1	0	0	0	1	1

Observe que el número 239 se puede calcular rápidamente restándole 16 a 255. Utilizando la orden en BASIC POKE reg,PEEK(reg)AND239 hemos devuelto al registro su estado original.

Estas técnicas se aplican más ampliamente en la programación en código de lenguaje máquina, donde alterar el estado de los registros de control podría constituir una parte importante del programa.

Respuestas al ejercicio 3

1a) $A.(\overline{A} + \overline{B})$
$= A.\overline{A} + A.\overline{B}$. (propiedad distributiva)
$= A.\overline{B}$ ($A.\overline{A} = 0$)

b) $X + Y.(X + \overline{Y}) + X.(\overline{X} + Y)$
$= X + Y + X.(\overline{X} + Y)$ (relación 5)
$= X + Y + X.Y$ (relación 6)
$= X + Y$ (absorción)

c) $P.Q + \overline{P}.Q + \overline{P}.\overline{Q}$
$= P.Q + \overline{P}.(Q + \overline{Q})$ (propiedad distributiva)
$= P.Q + \overline{P}$ ($Q + \overline{Q} = 1$)
$= \overline{P} + Q$ (dual de la rel. 6)

d) $\overline{X + \overline{Y}.\overline{Z} + \overline{Z}.Y}$
$= \overline{X}.Y.Z.\overline{Z}.\overline{Y}$ (Morgan)
$= X.Y.Z.(Z + \overline{Y})$ ($\overline{X} = X$, y además Morgan)
$= X.Y.Z.Z + X.Y.Z.\overline{Y}$ (propiedad distributiva)
$= X.Y.Z + 0$ ($Z.Z = Z$, $Y.\overline{Y} = 0$)
$= X.Y.Z$

3) Si los tres interruptores son X, Y y Z y la luz del recibidor es P, la tabla de verdad es:

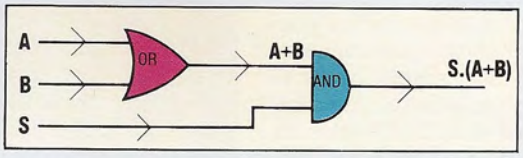
ENTRADAS			SALIDAS
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$P = \overline{X}.\overline{Y}.Z + \overline{X}.Y.\overline{Z} + X.\overline{Y}.\overline{Z} + X.Y.Z$
$= Z.(\overline{X}.\overline{Y} + X.Y) + \overline{Z}.(\overline{X}.Y + X.\overline{Y})$ (pr. dist.)
$= Z.\overline{(\overline{X}.Y + X.\overline{Y})} + \overline{Z}.(\overline{X}.Y + X.\overline{Y})$ (Morgan)

2) La tabla de verdad para el sistema de alarma:

ENTRADAS			SALIDAS
A	B	S	Alarma
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

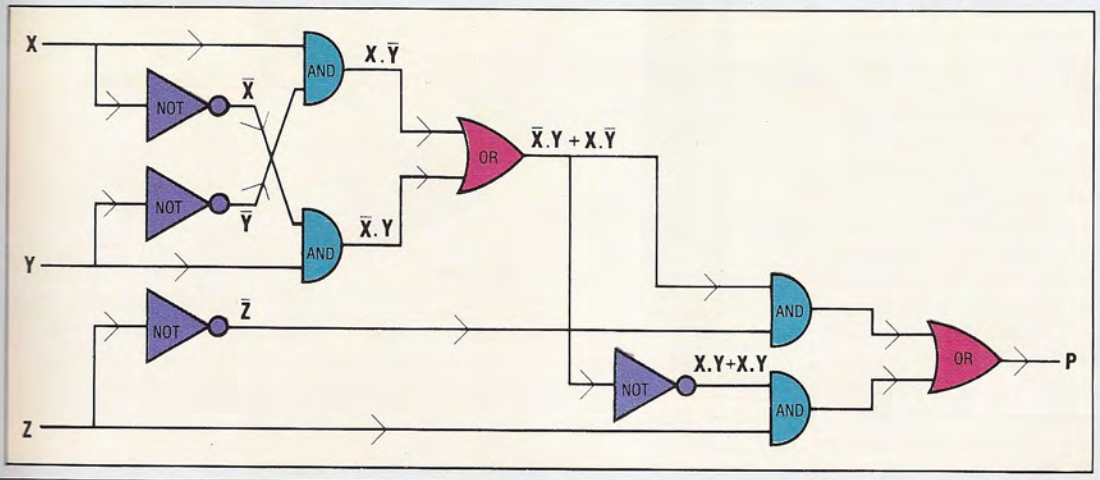
Alarma $= \overline{A}.B.S + A.\overline{B}.S + A.B.S$
$= \overline{A}.B.S + A.S.(\overline{B} + B)$ (prop. distrib.)
$= \overline{A}.B.S + A.S$ ($\overline{B} + B = 1$)
$= S.(A + \overline{A}.B)$ (prop. distrib.)
$= S.(A + B)$ (dual de la rel. 6)



4) Esta tabla demuestra que la pregunta "¿Dice usted siempre la verdad?" sirve de poco, pues tanto quien suela mentir como quien diga la verdad responderán igual. La tabla posee la misma forma que la función $X.Y + \overline{X}.Y$, que se simplifica a Y. Es decir, la respuesta depende de una variable, no de dos, y la pregunta no distingue entre quienes mienten y quienes no. Ahora bien, si preguntamos "¿Tienen alas los cerdos?", la tabla es:

		POSIBLES RESPUESTAS	
		SI	NO
POSIBLE IDENTIDAD DE QUIEN CONTESTA	MIENTE	1	0
	DICE LA VERDAD	0	1

tabla de verdad para la función $X.\overline{Y} + \overline{X}.Y$, que es, asimismo, una tabla para OR exclusivo. La pregunta nos permite identificar a quien contesta.



Retirar y sustituir

Esta vez nuestro ejercicio consiste en quitar un chip de ROM y sustituirlo por un conector que acoja la memoria extraída

Si en el capítulo anterior hemos aprendido a soldar, el siguiente paso consiste en averiguar cómo deshacer conexiones soldadas, con limpieza y de manera expedita. Si sólo estamos tratando con cables y conectores, entonces no necesitamos preocuparnos mucho: probablemente sea más simple cortar el cable, tirar el enchufe y empezar otra vez con componentes nuevos. Pero ¿qué sucede si hemos de sustituir o cambiar de lugar un chip? Hasta los transistores más simples poseen tres patillas o terminales. Para liberar un transistor de su tablero impreso, necesitamos calentar las tres al mismo tiempo hasta el punto de fusión de la soldadura, de modo que se pueda quitar limpiamente el transistor; o podemos retirar las patillas de una en una. Y si la idea de liberar tres patillas a la vez es intimidante, ¿cómo serán las 40 patillas de un microprocesador mediano de ocho bits?

Sustituir un chip

Más que elegir simplemente un chip para reemplazar al azar, o agregar chips de RAM para ampliar la capacidad de una máquina, vamos a afrontar un proyecto algo ambicioso: reemplazar una ROM de BASIC del ZX81 con un conector, tirar un cable plano fuera de la carcasa hasta una placa de circuito impreso, e instalar un conector para acoger la ROM que hayamos extraído o bien cualquier otra. La razón por la que hemos elegido el ZX81 como tema para este ejercicio en particular es que el FORTH se vende como lenguaje incorporado de recambio. Además del nuevo lenguaje, la ROM también incorpora un sistema operativo multitarea que permite la operación simultánea de más de un programa, y cada uno de ellos de forma totalmente independiente respecto a los demás. En una máquina tan pequeña resulta un logro notable, aunque requiere un mínimo de dos Kbytes de RAM, lo que podría suponer la necesidad de agregar un módulo de memoria extra.

Además de ser un ejercicio en el que se pone en práctica todo lo que hemos analizado hasta el momento, un pequeño proyecto como éste también proporciona cierta experiencia en cuanto a la manipulación de componentes y le servirá como demostración de lo necesario que es ser pulcro y preciso. En el próximo capítulo de la serie analizaremos las formas de verificar el artículo ya acabado utilizando un multímetro. Este dispositivo mide corriente, voltaje y resistencia, y es una herramienta de incalculable valor para verificar y probar circuitos y componentes.

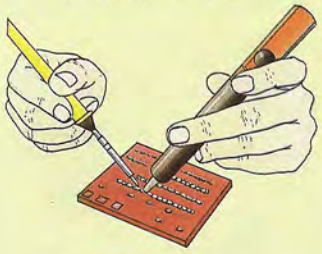
Receta

Si alguna vez ha de sustituir un chip de su ordenador personal o, como en nuestro ejemplo, debe simplemente volver a colocarlo, o si se ha de agregar un chip nuevo, es una excelente práctica instalar un conector de chip para alojarlo. Los chips con conector se pueden reemplazar en un momento; y si las sustituciones han de efectuarse con bastante frecuencia, existen a la venta refinados portachips, que minimizan el riesgo de torcer una patilla. Además de las herramientas que hemos descrito antes (véase p. 524), necesitará o bien un carrete de trenza para desoldar, o una herramienta para desoldar. El ejercicio que hemos escogido implica extraer el ROM de BASIC de un ZX81 y sustituirlo por un conector estándar desde el cual un cable plano conducirá a otro conector fuera de la máquina. El objetivo consiste en reemplazar el BASIC del ZX81 por el FORTH en ROM multitarea de David Husband, proporcionándole al mismo tiempo al usuario la posibilidad de retornar al BASIC en el futuro. En el caso de que desee usted realizar este ejercicio, también necesitará disponer de otros tres elementos: una placa de circuito impreso, un cable plano de 28 vías (ambas cosas las podrá encontrar sin dificultad en cualquier tienda de componentes electrónicos) y un trozo de poliestireno

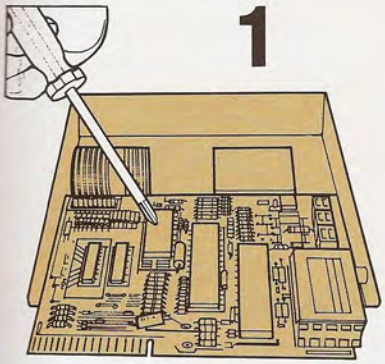
Localizar la ROM
El primer paso consiste en abrir la carcasa y localizar la ROM de BASIC. La carcasa se mantiene unida mediante cinco tornillos, tres de los cuales están situados debajo de los "pies" de goma autoadhesivos de la superficie inferior. El relleno que no oculta ningún tornillo es el que se halla más próximo a los conectores EAR y MIC. Despegue cuidadosamente los otros tres rellenos y extraiga los tres tornillos de cabeza en cruz que hay debajo de ellos, y después los otros dos tornillos que hay a la vista. Levante la superficie inferior de la carcasa para dejar al descubierto la parte inferior del tablero de circuito impreso. Éste se puede extraer entonces de la carcasa mediante los tres tornillos de cabeza en cruz visibles, dos adyacentes al conector marginal y el otro cerca del disipador (la placa de aluminio que hay debajo del teclado). Dé vuelta al tablero. El chip de ROM que estamos buscando está situado arriba y ligeramente a la izquierda de los conectores del teclado

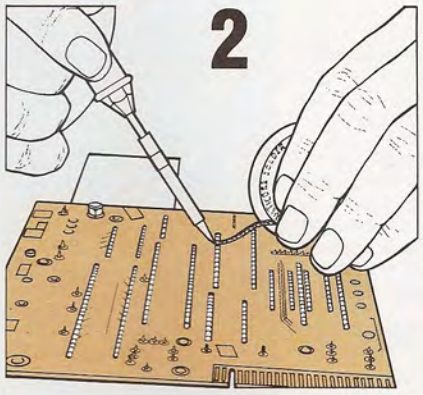
Desoldadores

Existen dos medios patentados para desoldar. El menos caro a corto plazo es la "mecha de soldadura", alambre de cobre muy fino, trenzado en una cinta e impregnado de fundente, que se basa en el fenómeno de la acción capilar (una función de la tensión de superficie que hace que los líquidos asciendan por tubos estrechos) para absorber la soldadura fundida, así como una mecha de tejido llevaría el combustible hasta el quemador de una lámpara o de una estufa. La mecha de soldadura trenzada viene en varios anchos que están directamente relacionados con la cantidad de soldadura a eliminar. Es desechable y no se puede volver a utilizar. El segundo método es mucho más satisfactorio y consiste en un dispositivo parecido a un inflador de bicicleta con resorte, pero que funciona a la inversa: por el extremo aspira en vez de soplar. Este dispositivo tiene larga vida útil y su operación es mucho más rápida que la de las trenzas. Como puede ver, para quitar componentes con éxito es esencial seguir uno de estos dos procedimientos.

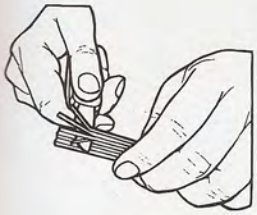


Desoldador
Caliente la juntura a desunir con el soldador hasta que corra la soldadura, aplique la herramienta para desoldar, pulse el botón disparador y la soldadura líquida será absorbida hacia el interior del cuerpo de la herramienta

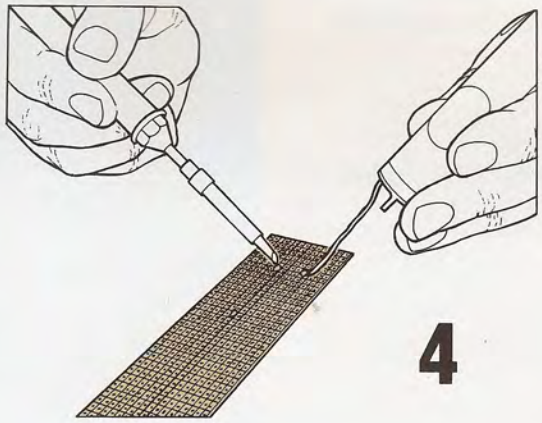




Separar la ROM
Cuando hay que desoldar, como cuando se hacen junturas, el secreto estriba en aplicar suficiente calor. Presione la trenza de soldadura sobre la juntura con la punta del soldador hasta que corra el fundente de la trenza (es probable que vea un poco de humo azul). Verá cómo la trenza absorbe la soldadura. Vaya cortando con regularidad el extremo de la trenza, ya que las secciones pequeñas se saturan rápidamente

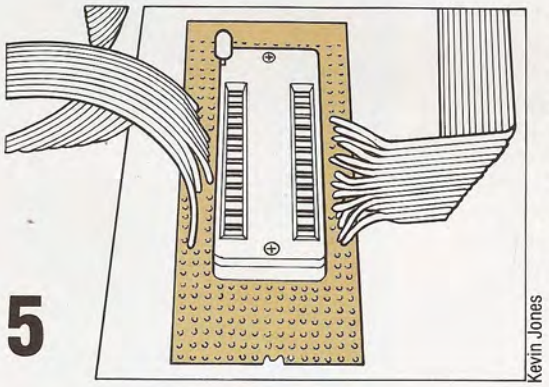


Cable plano
El cable plano viene en diversos anchos: nosotros necesitamos el de 28 vías (es decir, 28 cables separados) o dos tiras de 14 vías. Separe por uno de los extremos cada uno de los 28 núcleos un centímetro y dé una capa de soldadura. Trabajando desde el extremo marcado con un pequeño corte semicircular, suelde el cable plano en su sitio por ambas caras. Todos los chips poseen estos cortes, o a veces puntos, para indicar en qué forma se deben alinear



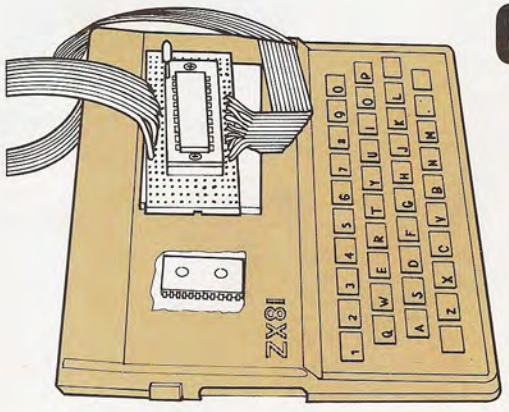
Conectar
Corte los dos trozos de cable plano a la longitud adecuada y pele y estañe los 28 núcleos. Luego, asegurándose de que los núcleos que parten del extremo marcado de la superficie que sostiene el chip se dirijan hacia el mismo extremo del conector, suéldelos en la placa de circuito impreso. Asegúrese de que no se cruce ninguno de los núcleos

Placas de circuito impreso
Existen en el mercado distintos tipos de placas de circuito impreso. Consisten en una lámina de plástico rígida, perforada siguiendo un patrón de matriz estándar, con hebras de cobre que constituyen cada una de las filas de la matriz. El contacto entre cada punto de conexión se puede romper cortando la hebra de cobre y se pueden efectuar conexiones entre las filas. Estas placas también son conocidas con los nombres de Veroboard y Veroblock. Esta última corresponde a una versión más sofisticada



Montar el conector
Para estos fines la placa de circuito impreso es el medio más adecuado. Tiene quince puntos de contacto de ancho por aproximadamente 25 cm de largo y se puede quebrar con los dedos para obtener la longitud deseada. Coloque el conector del chip de modo que quede bien alineado respecto al aislamiento central y suelde la patilla de una esquina para sujetarlo con firmeza; luego vaya trabajando cada lado, asegurándose de que las uniones fluyan limpiamente

ATENCIÓN
La garantía de su ordenador personal (en caso de que aún estuviera en vigor) puede quedar anulada si alguien que no fuera el fabricante o un agente autorizado abriera la carcasa



Acabado
Cuando ya estén hechas todas las conexiones, compruebe, tanto en el propio tablero del ordenador como en el tablero hijo recién creado, que cada una de ellas esté separada de todas las demás. Cuando se trabaja con miniaturas de este tipo, en las que hay muchas conexiones unas junto a otras, éstas pueden entrar en contacto, lo que produciría cortocircuitos. Compruebe las conexiones mediante una lente de aumento y, en caso de duda, asegúrese deslizando a través del agujero una aguja u otro instrumento metálico punzante

Salida impresa

Para fines de gestión, una impresora cara, como la de rueda margarita, es casi esencial, pero para el usuario de un ordenador personal existen alternativas más económicas

La producción de una copia impresa del listado de un programa o trozo de texto está entre las prioridades del usuario de un ordenador. Lamentablemente, el costo de una buena impresora matricial podría representar el doble o el triple del precio del ordenador, mientras que un dispositivo de rueda margarita está de seguro fuera de su alcance. Pero hay buenas ofertas a precio muy económico, como las *impresoras sin impacto*.

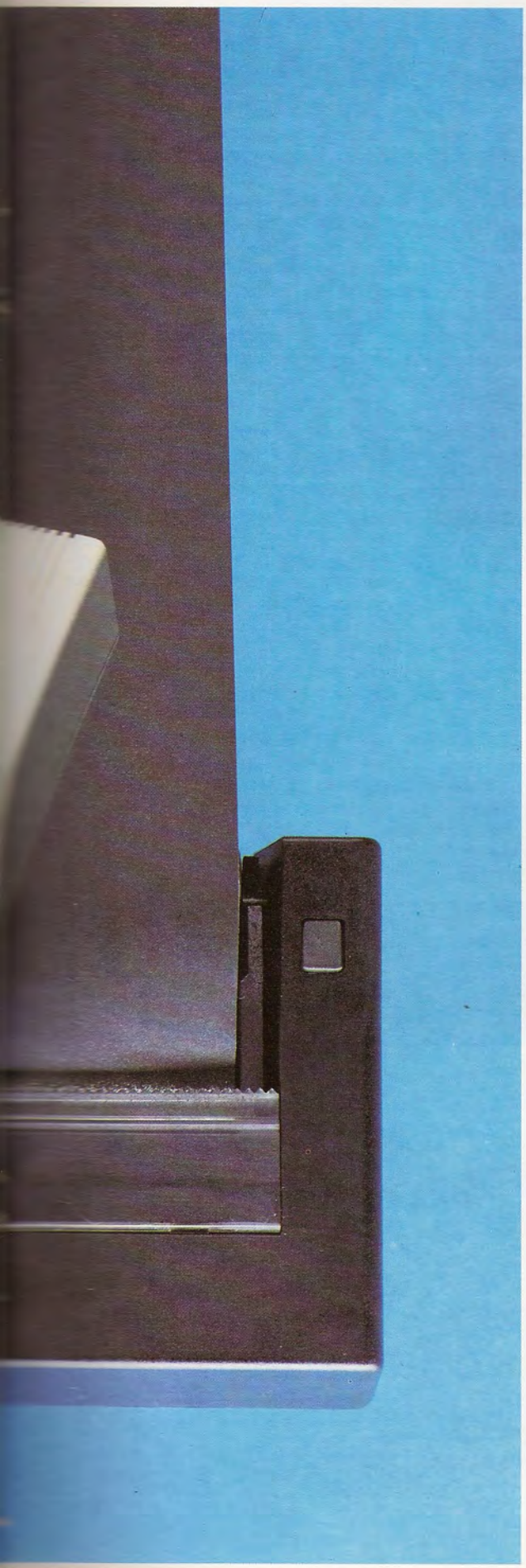
Este nombre proviene de que no martillean agujas ni trozos moldeados de metal a través de una cinta para dejar su marca sobre un trozo de papel. Las primeras impresoras sin impacto se desarrollaron para las cajas registradoras y terminales portátiles, como la famosa serie Texas Silent 700.

Así como una impresora matricial de impacto posee una columna vertical de agujas en una cabeza que se desplaza horizontalmente a través de la página, la impresora térmica posee una columna de elementos de calentamiento. A medida que se van pidiendo puntos, el elemento correspondiente se calienta rápidamente y la diminuta superficie de papel debajo del elemento cambia de color.

Aparte de las terminales de Texas Instruments, las impresoras térmicas también fueron adoptadas por Apple, que produjo una pequeña impresora para su ordenador denominada *Silentype*, y Mattel, que ofrece una unidad similar para utilizar con su ordenador personal Aquarius.

Por su parte, el mecanismo que emplea la impresora electrostática es moderadamente ruidoso y, salvo un ejemplo temprano de Centronics denominado Microprinter P1, no consiguió la aceptación general hasta que sir Clive Sinclair adoptó el sistema para la impresora ZX Printer, de la cual se han vendido enormes cantidades como periférico exclusivo para el ZX81 y el ZX Spectrum.

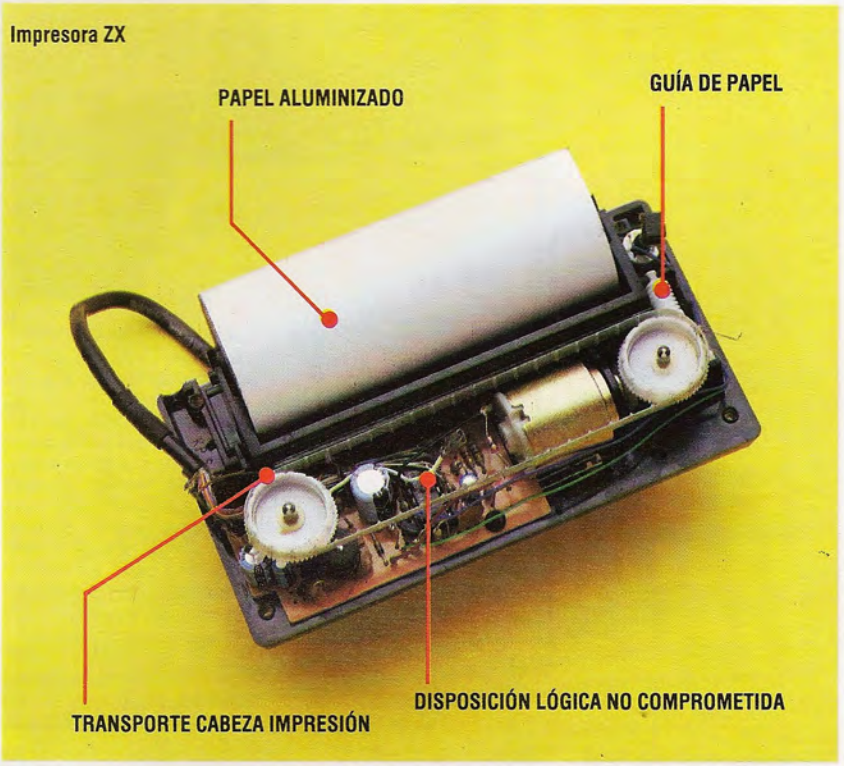
El principio en el que se basan las impresoras electrostáticas es una cabeza de impresión de un único cable que se arrastra a través de un papel con un revestimiento especial. En cada uno de los puntos que se requieren para construir un carácter, la impresora genera una chispa que quema el delgado revestimiento metálico y deja al descubierto el papel negro de fondo. Sinclair mejoró el sistema utilizando dos cabezas en un cinturón continuo,



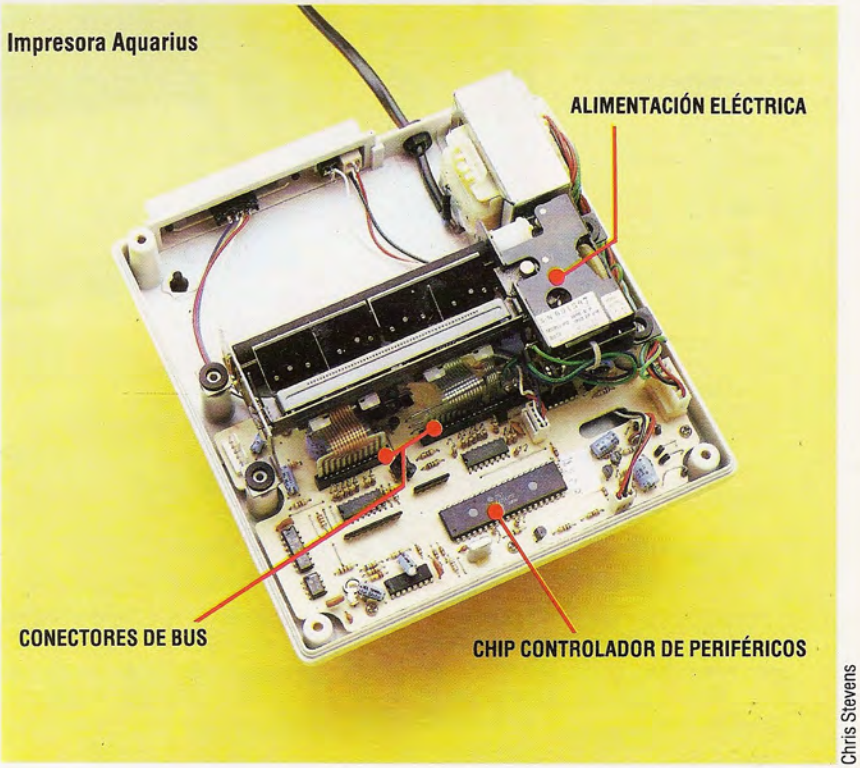
La impresora ZX
Introducida como dispositivo de salida para el ZX81, la impresora ZX también es compatible con el Spectrum. Ofrece un método económico en virtud del cual los usuarios de ordenadores personales pueden obtener resultados y listados de programas impresos. Aunque su reducido costo resulta interesante, su dependencia del papel electrostático la deja en clara desventaja

La impresora Aquarius
Para su microordenador personal, Aquarius optó por el otro procedimiento de bajo precio: la impresión térmica (aunque también hay a la venta una impresora económica a cuatro colores que utiliza lápices esferográficos). Al igual que la impresora ZX, adolece del inconveniente de que depende de un papel especial

Impresora ZX



Impresora Aquarius



Chris Stevens

pero todavía siguen siendo necesarias ocho pasadas de una cabeza para crear cada fila de caracteres. Por suerte, la impresora ZX sólo tiene que imprimir 32 caracteres en cada línea, de modo que la velocidad es aceptable.

Unidad con clase
De todas las representantes de la tecnología de impresión térmicas, pocas son tan elegantes como la Brother EP-22. Por menos de lo que valen muchas de las impresoras más baratas de cualquier tipo, usted no sólo adquiere una impresora térmica de 75 columnas sino, por añadidura, una máquina de escribir portátil. Diseñada para operar con papel térmico y como una impresora normal de impacto con una cinta acoplada, la unidad puede almacenar hasta 2 000 caracteres en su memoria, permitiendo crear cartas o recordatorios sobre la marcha. La memoria electrónica en realidad no admite el tratamiento de textos como tal, pero se puede corregir cualquier carácter de los últimos 16 a los que ha dado entrada, ya que, antes de impresionar el papel, la salida va a una visualización en cristal líquido incorporada. La calidad de la salida no es equiparable con la de una auténtica "impresora" (las letras en minúscula carecen, por ejemplo, de auténticos trazos bajos), pero para acompañar ordenadores tales como el Epson HX-20 y el Tandy Modelo 100 cumple una función que sería difícil de obtener con otra forma de impresora de impacto

El principal inconveniente de ambos tipos de impresora es que utilizan papel especial. Los materiales de este tipo suelen ser caros y sólo se venden en rollos, lo que dificulta el almacenamiento. En el caso de la impresora térmica, es esencial adquirir un papel de la calidad adecuada; de lo contrario, la imagen no se desarrollará correctamente, además de perder intensidad con el tiempo o la exposición al calor. El papel electrostático es aún más delicado y, si es manipulado con las manos húmedas o sudorosas, la imagen se torna confusa y se va diluyendo, ya que el revestimiento se disuelve. En ambos casos la forma de asegurar una imagen buena y duradera consiste en hacer una fotocopia.

A pesar de tales inconvenientes, estos dos tipos de impresora ofrecen a los fabricantes de los ordenadores personales más pequeños la posibilidad de proporcionar alguna forma de sistema de impresión para sus máquinas, a la que, en otras circunstancias, tendrían que renunciar. El método de impresión matricial permite realizar copias directas de la visualización en pantalla, de modo que se pueden reproducir tanto textos como gráficos sin costo adicional, si al usuario no le preocupa demasiado la relativamente baja calidad de la imagen final.

Recientemente, sin embargo, el dominio que ejercen estos sistemas de impresión ha sido desafiado por los plotters de lápices de cuatro colores que suministran Tandy y Sharp. Tales lápices pueden producir juegos de caracteres excelentes, totalmente formados, además de soberbios gráficos de líneas en rollos de papel normal, por aproximadamente el mismo costo que los dispositivos electrostáticos o térmicos. A medida que se va desarrollando la tecnología de impresión, el dispositivo sin impacto, con su sencillo mecanismo y sus limitadas capacidades, sigue siendo el que mejor satisface las necesidades de impresión del mercado de ordenadores personales económicos.

Ian McKinnell

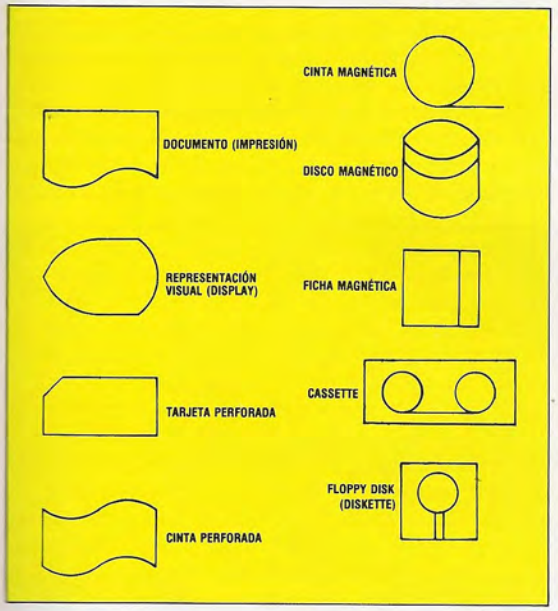
La impresora térmica de Tandy
Más adaptable que las otras dos impresoras de precio económico que hemos descrito, la Tandy TP-10 es compatible con todos los microordenadores de la gama Tandy

Simbología

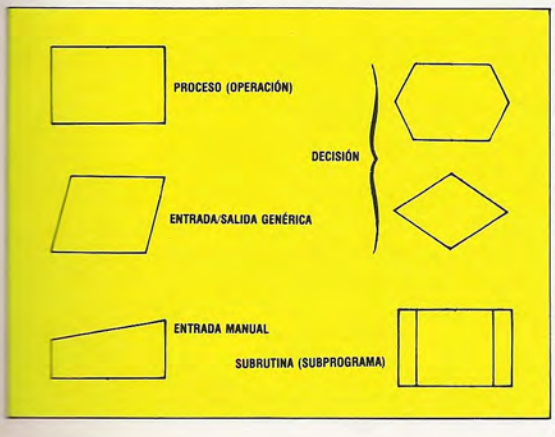
Hemos reunido en esta página los dibujos geométricos que nos servirán de símbolos en posteriores diagramas de flujo, a medida que éstos vayan siendo más complejos

Los diagramas se componen, tal como se ha visto, de una sucesión de símbolos gráficos que representan de forma individual las operaciones que realiza el ordenador. Dichos símbolos pueden clasificarse en cuatro apartados: de sistema, de proceso, auxiliares y de líneas de flujo.

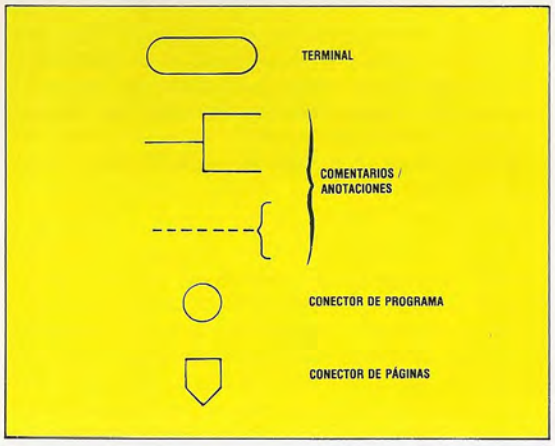
Símbolos de sistema: Son los que representan soportes de datos, ya sean manuales o automáticos. Representan asimismo las unidades y componentes de los sistemas.



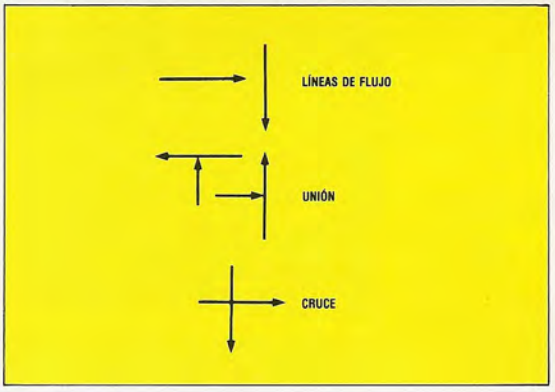
Símbolos de proceso: Representan el desarrollo de las operaciones individuales según la expresión incluida dentro de cada símbolo.



Símbolos auxiliares: Se utilizan para dar mayor comprensión y claridad al diagrama en su conjunto.



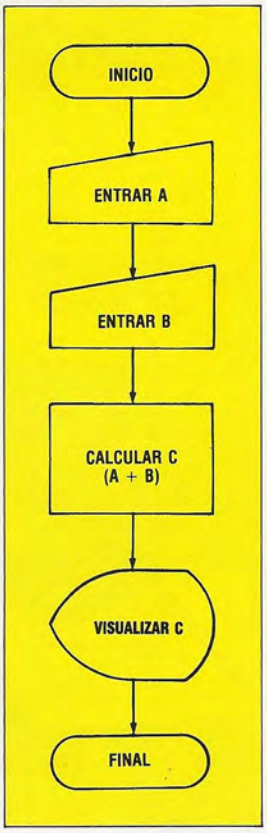
Líneas de flujo: Se las podría considerar como un subapartado del anterior grupo; son las que marcan el orden de la secuencia de operaciones y que envían hacia otros puntos del diagrama, conectándose entre sí.



Paulatinamente, y conforme aparezcan en los problemas que se vayan realizando, se explicarán éstos y otros posibles símbolos o combinaciones, así como la función que cumplen dentro del diagrama.

El ejemplo de la derecha muestra cómo, mediante la introducción de dos valores por teclado, se llega a la visualización del valor resultante tras la operación de la suma de dichas cantidades. Sirve, además, para comentar el símbolo perteneciente al grupo de auxiliares conocido por *terminal*. Dicho símbolo muestra indistintamente tanto el inicio (START) como el final (END) del organigrama.

Su utilización es obligatoria en todo diagrama.



Rapidez y eficiencia

Esta vez analizaremos dos expeditos y recomendables paquetes de tratamiento de textos creados para el BBC Micro

Uno de los paquetes más populares de tratamiento de textos para el BBC Micro es Wordwise, que está almacenado en un chip que se acopla en un conector para ROM del ordenador. Una vez acoplado, digite *W y el BBC se convertirá en un procesador de textos. El paquete Wordwise, tal como se entrega, consta del chip ROM, instrucciones para acoplamiento, un manual de 30 páginas y una cassette que visualiza en la pantalla el texto del manual e ilustra provechosamente las funciones de diversas órdenes.

El Wordwise es de fácil manejo y se puede utilizar de inmediato sin necesidad de adquirir adiestramiento previo y sin tener que remitirse de manera constante al manual. Para iniciarse, tan sólo elija la modalidad edición pulsando la tecla Escape y comience a digitar. La edición la efectúa el cursor y las letras se pueden digitar o borrar. El Wordwise se vale de las teclas de función del BBC para desplazar y copiar texto.

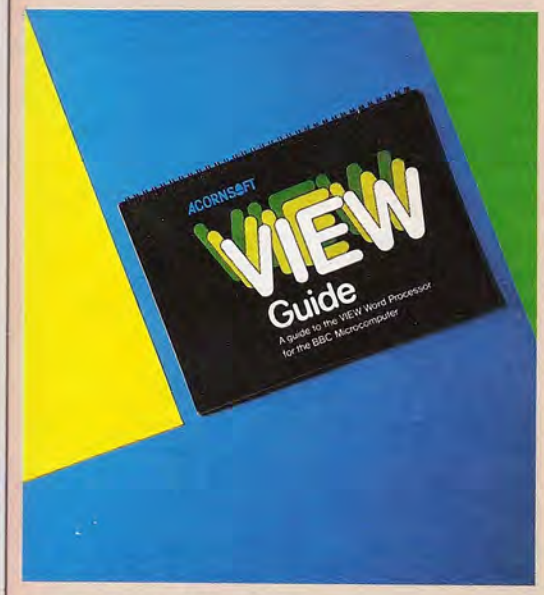
A medida que el usuario se va familiarizando con la mayor flexibilidad que ofrece un procesador de textos en comparación con la máquina de escribir o el lápiz y el papel, puede ir explorando con mayor profundidad las configuraciones. Enseguida entran a formar parte del repertorio del usuario el saltar hasta el principio o el final de una línea o página utilizando las teclas Shift y Cursor, o sobreescribir letras mediante el empleo de una tecla de función.

Los caracteres aparecen en la pantalla en la modalidad de 40 columnas y son totalmente legibles en un televisor doméstico. El aspecto final del texto se puede ver previamente en la pantalla en la modalidad de 80 columnas mediante una tecla de función. Ésta permite apreciar cómo aparecerá el texto una vez impreso. Es probable que en esta etapa de visión previa el recién iniciado en el Wordwise considere primero el trazado general y el aspecto del texto y necesite remitirse a la sección del manual que incluye las órdenes definibles por el usuario. Estas órdenes se ocupan de la faceta de presentación del tratamiento de textos. Como ejemplo, podemos citar la orden LM seguida de un número, que establece la posición del margen izquierdo (*Left*

El equipo completo

Una instalación para tratamiento de textos debe incluir teclado y monitor (o televisor), disco y unidad de disco para almacenamiento, impresora y útiles de escritorio para una salida impresa

Kevin Jones



Una "visión" alternativa

El View ("visión"), paquete de procesamiento de Acorn para el BBC Micro, también se entrega como chip de ROM. Al igual que el Wordwise, se puede instalar de manera que se cargue a sí mismo cada vez que se encienda la máquina, o se le puede dar entrada desde el BASIC.

El View utiliza la potencia del generador de video del BBC Micro para ofrecer visualizaciones de 76, 74, 34 o 16 columnas, con la consiguiente modificación de las dimensiones de los caracteres. Este efecto se produce "aumentando" la imagen completa en pantalla; la pantalla se convierte entonces en una ventana que viaja a través de la pantalla virtual, mucho más grande. La orden Mode, que efectúa estas modificaciones, opera en la modalidad Command (orden) del View, igual que las órdenes de acceso al disco y de "hallar y sustituir" e imprimir.

El recurso de "hallar y sustituir" del View, además de la función normal de buscar a través del documento cada una de las instancias en que se produce una palabra determinada, permite lo que el manual designa como *búsqueda salvaje*. La palabra se especifica de la forma normal, pero cualquier carácter dudoso se sustituye por "?" Por consiguiente, si se da entrada a "tr??" como la palabra clave, la búsqueda hallará "tras", "tren", "tres", "trío", etc., o sea, toda palabra de cuatro letras que empiece con "tr". Hay que ser cuidadoso al definir la clave de búsqueda salvaje, pues se corre el riesgo de que el ejercicio sea contraproducente. Otra configuración del juego de órdenes permite contar las palabras de un texto.

El otro protocolo operativo del View, llamado *modalidad de pantalla (screen mode)*, se divide en dos grupos de órdenes: *immediate* (inmediatas), que controlan la inserción-supresión de caracteres, el movimiento del cursor, los movimientos de bloques y otras necesidades; y *stored* (almacenadas), que se cuidan de los parámetros para la creación o edición de un documento, dar formato a las páginas, encabezamientos y finales de páginas y otros requerimientos continuos. En la modalidad de pantalla, el View hace una utilización excelente de las teclas de función del BBC Micro, cada una de las cuales (10) posee tres usos diferentes según se empleen solas o conjuntamente con las teclas Shift o Control

Liz Heaney

Margin). Del mismo modo, LL para longitud de línea (*Line Lenght*), IN para sangrado (*INdent*), PL para longitud de página (*Page Lenght*) y así sucesivamente. Al ser mnemónicas, es fácil recordar estas órdenes y darles entrada en el texto después de pulsar una tecla de función. No aparecen en el texto visto previamente o impreso.

Si no se utilizan las órdenes definibles por el usuario, el Wordwise pasa por omisión a los valores incorporados: 70 caracteres de longitud de línea, 66 líneas de longitud de página y cinco espacios para el margen izquierdo. De modo, entonces, que se puede disponer de un documento bien trazado antes de dominar las órdenes para dar formato.

Los códigos de control de la impresora para determinar cursivas, destacar y duplicar el tamaño de los caracteres y otras opciones también se pueden digitar como órdenes definibles por el usuario.

Wordwise opera en una modalidad de menú para guardar, cargar e imprimir secciones completas de texto. El menú aparece en la pantalla cuando se selecciona Wordwise y visualiza ocho opciones. Las primeras cuatro afectan a la carga y guardado de texto en, o desde, disco o cassette.

La opción cinco le permite al usuario buscar y reemplazar ítems específicos de texto. Un ejemplo en un artículo como este que nos ocupa sería reemplazar la palabra "digitar" por "teclear". En esta opción, las elecciones son *globales* o *selectivas*: la primera posibilita que se efectúe la sustitución en todas las instancias en las que aparece en el texto; mientras que la segunda permite que el usuario desplace el cursor hasta la primera instancia de aparición, luego a la siguiente y así sucesivamente, escogiendo si sustituir o no la palabra en cada caso. La opción seis del menú permite imprimir el texto y la siete permite verlo previamente. La opción ocho guarda el texto ya con formato sin órdenes definibles por el usuario.

Para pasar del menú a la modalidad de edición, se pulsa la tecla Escape y aparece el texto con el cursor posicionado donde se dejó. Debido a que la pantalla visualiza el texto en la modalidad de 40 columnas, con frecuencia es necesario saltar de la modalidad de edición a la opción siete (visión previa) del menú cuando se da formato al texto. Un ligero problema es que los efectos de los tabuladores no aparecen en la pantalla; por este motivo se suele requerir con frecuencia la visión previa del texto para verificar el trazado final del documento cuando se imprima.

Un aspecto interesante del BBC es que las teclas de función definidas por el usuario que utiliza el Wordwise para desplazar, copiar o eliminar texto y llevar a cabo otras tareas se pueden volver a definir para su empleo con las teclas CTRL o SHIFT con el fin, por ejemplo, de producir un nuevo párrafo, suprimir una línea completa o incluir un código para la impresora como una orden definible por el usuario. Utilizadas aisladamente, las teclas de función retienen las funciones definidas del Wordwise. Las órdenes estrella (*) del BBC se pueden utilizar en la modalidad de menú para seleccionar un tipo de impresora, seleccionar cinta o disco o retornar la máquina al BASIC digitando *B.

El Wordwise es, indudablemente, una útil adición para cualquier BBC Micro y es fácil de instalar y de utilizar. Para el BBC existen en el mercado procesadores de texto más caros y más sofisticados, como el View, pero se puede afirmar que el Wordwise satisface la mayoría de las necesidades de los usuarios.

Análisis de texto

Antes de abordar los programas de lenguaje máquina, analizaremos cómo es el área de textos en una memoria tratada con intérprete de BASIC. Saberlo nos será de gran ayuda más adelante

Cuando usted digita o carga (LOAD) un programa en BASIC en su ordenador, quizá se imagina que el ordenador es un recipiente vacío que no hace nada hasta que llegan sus instrucciones. En realidad, desde el instante mismo en que enchufa su ordenador, éste se pone a ejecutar por su cuenta un complejo programa: el sistema operativo (*Operating System*: OS). Más que un programa es un juego de programas, que está de forma permanente en algunos de los chips de ROM del interior de la máquina. Tiene como objetivo hacer que la máquina funcione: controla la pantalla, comunica con la impresora y las unidades de disco, explora el teclado en busca de pulsaciones de teclas, etc. Para el OS, cuanto entra en la máquina no es sino datos a procesar mediante sus propios programas.

Uno de estos programas se denomina *intérprete de* BASIC y tiene como objetivo inspeccionar el texto de los programas en BASIC y ejecutar las instrucciones que contienen. Por consiguiente, todo cuanto contienen nuestros programas no son más que datos a procesar por el programa intérprete. Cuando se digita un programa, el sistema operativo lo reconoce como tal porque cada nueva línea empieza con un número de línea válido. Con algunas excepciones, cada uno de los caracteres de esa línea del programa se almacena en su propio byte del área de textos de programas en BASIC de la memoria. Cuando usted digita RUN, el sistema operativo le entrega el control al intérprete de BASIC que se pone a trabajar en el procesamiento de sus datos (el contenido del área de textos en BASIC).

El intérprete no modifica su programa en ningún sentido, sino que simplemente lo interpreta y ejecuta. Y como el intérprete obedece órdenes sin más, es bastante factible darle instrucciones para que mire el contenido de cualquier área de la memoria. Si su programa le permite inspeccionar la memoria y usted lo usa para inspeccionar el área de textos, eso para el intérprete no es ninguna paradoja. Él sencillamente sigue instrucciones, si es que puede, y si no puede informa ERROR DE SINTAXIS o ERROR DE DESBORDAMIENTO DE CAPACIDAD o algo similar: no tiene ni el razonamiento ni el vocabulario para producir mensajes de error tales como: PARADOJA TEMPORAL o DISCONTINUIDAD FILOSOFICA.

El sistema operativo almacena el programa en BASIC carácter por carácter, a excepción de las instrucciones. Cada vez que reconoce las letras (o caracteres, o números, o patrones de voltaje) que componen una instrucción en BASIC, el sistema operativo reemplaza esa palabra por un número de código de un único byte, denominado *distintivo* (*token*). Así se ahorra espacio de memoria (RESTORE, p. ej., con sus 7 letras debería ocupar 7 bytes) y hace que la tarea del intérprete de traducir el programa sea mucho más fácil de llevar a cabo.

Las máquinas utilizan distintos tipos convencionales de distintivos, pero, en general, estos códigos son números mayores de 127. Los códigos ASCII para los caracteres imprimibles (reflejados en la tabla de la página siguiente) pertenecen todos a la escala entre 32 y 127. Por consiguiente, cualquier byte del área de textos en BASIC que contenga un número mayor que 127 ha de ser un byte distintivo colocado allí por el sistema operativo. Cuando el intérprete se encuentra con uno de estos bytes, ejecuta la pertinente subrutina incorporada.

Surge el interrogante, sin embargo, de por qué cuando uno LISTa un programa no ve caracteres no imprimibles sino más bien las instrucciones en BASIC. La respuesta es que durante un LISTado el sistema operativo inspecciona cada byte del área de textos y cada vez que encuentra un byte cuyo valor sea mayor que 127 lo trata como si fuera un distintivo. En algún lugar de la memoria hay almacenada una lista completa de las representaciones ASCII de las instrucciones en BASIC y el valor de un distintivo señalará dicha posición. Es lo mismo que si el intérprete estuviera utilizando el valor del distintivo para localizar la subrutina de su ejecución. Y, por consiguiente, el sistema operativo coloca en la pantalla, durante un LISTado, la palabra clave en vez del distintivo. Se lo demostrará un Commodore 64. (En el BBC y el Spectrum es menos directo.) Digite en minúsculas:

```
100 rem*******h**********
```

Ahora LISTe 100 y verá:

```
100 rem*******left$**********
```

En las máquinas Commodore el valor ASCII de la "h" en modalidad minúsculas es 200, de modo que cuando el sistema operativo encontró un valor de 200 en ese byte determinado durante el LISTado, lo interpretó como el distintivo para la palabra clave LEFT$. Si ahora digita:

```
100 rem"*******H**********"
```

y LISTa 100, verá:

```
100 rem"*******H**********"
```

Esto demuestra que es importante recordar que algunos caracteres imprimibles, por lo general caracteres para gráficos, poseen códigos ASCII mayores que 127 y que se los reconocerá como tales siempre y cuando estén entre comillas. De no ser así, se tratarán como distintivos.

Siglas inglesas de los caracteres de control del código ASCII
Aunque los textos en castellano suelen ofrecer al lector la puntual significación de estas siglas, a cuyos caracteres el ASCII otorga valores del 0 al 31, puede que el lector desee saber de dónde derivan. He aquí su procedencia (la traducción se encuentra en el cuadro de la página siguiente)

NUL — Does nothing
SOH — Start heading
STX — Start of text
ETX — End of text
EOT — End of transmission
ENQ — Enquire
ACK — Acknowledge
BEL — Ring bell
BS — Backspace
HT — Horizontal tab
LF — Line feed
VT — Vertical tab
FF — Form feed
CR — Carriage return
SO — Shift out
SI — Shift in
DLE — Data link escape
DC1 — Device control 1
DC2 — Device control 2
DC3 — Device control 3
DC4 — Device control 4
NAK — Negative acknowledge
SYN — Synchronous idle
ETB — End of transmission block
CAN — Cancel
EM — End of medium
SUB — Substitute
ESC — Escape
FS — File separator
GS — Group separator
RS — Record separator
US — Unit separator

Llegados a este punto, podemos comenzar a investigar cómo se almacena en la memoria una línea de programa en BASIC. Los ordenadores se diferencian entre sí sólo en detalles, pero en general los primeros tres o cuatro bytes de una línea de programa en BASIC del área de textos contendrán el número de la línea del programa y alguna información relativa a la longitud de la línea (véase cuadro). El número que usted le otorga a la línea cuando la digita también se almacena (si bien no por su equivalente ASCII, pues, por ejemplo, la línea 61030 necesitaría nada menos que cinco bytes sólo para almacenar su número). Se almacena como un entero de dos bytes. De esta manera, los números entre 0 y 255 (que, recuerde, caben en un byte de ocho bits) se almacenan en un byte de ceros seguido del byte que contiene al número. Los números mayores de 255 se almacenan exactamente como las direcciones paginadas (véase p. 516): el valor del primer byte se multiplica por 256 y se le suma al valor del segundo byte. Por ejemplo, 1000 se almacenaría como 3 232 (3*256 + 232 = 1000). Estos dos bytes siempre están en la misma posición en cualquier línea almacenada en el área de textos (si bien el que sean siempre los dos primeros bytes o no depende de cada máquina).

La información relativa a la longitud de la línea se coloca en un solo byte en el BBC y en dos bytes en el Spectrum. Representa simplemente el número de bytes de la línea (incluyendo los dos bytes para el número de línea y el propio byte que contiene esta longitud). Si conoce la dirección del primer byte de una línea de programa en BASIC en la memoria, y si le suma a él el contenido del byte de longitud de línea, entonces obtendrá la dirección del primer byte de la próxima línea del programa. Dado que el mayor número que se puede expresar con un solo byte es 255, la longitud máxima de una línea de programa en BASIC en el BBC es, por consiguiente, de 255 caracteres. Usted podría utilizar el programa Mempeek de la página 539 para establecer si es ése el límite del número de *caracteres* que puede digitar en una línea de programa, o si se trata del límite de la longitud de la línea tal como ésta se almacena en el área de textos.

En el Commodore, el byte de longitud de línea se sustituye por dos bytes que se denominan *dirección de enlace*. Ésta es, simplemente, la dirección real en forma de dos bytes del primer byte de la siguiente línea del programa.

Es interesante destacar que en el BBC y en el Spectrum la dirección de comienzo de la línea siguiente se calcula a partir de la dirección presente más la longitud de línea (lo que es lento pero ahorra un byte); mientras que en el Commodore, la dirección siguiente se almacena como tal (lo cual

Diagrama de conversión
El código norteamericano estándar para intercambio de información (*American Standard Code for Information Interchange*: ASCII) proporciona a los caracteres un valor de código estándar por medio de los números entre 0 y 127. Los códigos del 0 al 31 no asumen caracteres imprimibles, sino que se utilizan para enviarles señales de control a los periféricos como la pantalla y la impresora. Por tanto, el significado de estos códigos varía considerablemente de una máquina a otra, tal como refleja el diagrama. Algunas máquinas, en especial el Commodore y el Spectrum en nuestro diagrama, dejan muchos códigos sin utilizar (señalados aquí mediante un ●). Los códigos del 32 al 127 son para los caracteres imprimibles; los códigos ASCII estándar dentro de esta escala son los más generalizados (con variaciones menores) en la mayoría de los ordenadores. Su manual para el usuario le proporcionará los códigos ASCII para su máquina

CODIGO ASCII	ASCII	COMMODORE	SPECTRUM	BBC MICRO
0	NUL — No hace nada	●	●	Nulo
1	SOH — Inicio de encabezamiento	●	●	Siguiente carácter a impresora
2	STX — Inicio de texto	●	●	Activar impresora
3	ETX — Fin de texto	●	●	Desactivar impresora
4	EOT — Fin de transmisión	●	●	Separar cursores texto-gráficos
5	ENQ — Consulta	Tecla blanco	●	Juntar cursores texto-gráficos
6	ACK — Acuse de recibo	●	PRINT	Activar unidades VDU
7	BEL — Sonar timbre	●	EDIT	Emitir pitido breve
8	BS — Retroceder un espacio	Desactiva tecla CBM	Cursor izq.	Cursor atrás un espacio
9	HT — Tabulación horizontal	Activa tecla CBM	Cursor der.	Cursor adelante un espacio
10	LF — Avance de línea	●	Cursor abajo	Cursor abajo
11	VT — Tabulación vertical	●	Cursor arriba	Cursor arriba
12	FF — Salto vertical	●	Delete	Limpiar área de texto
13	CR — Retorno del carro	RETURN	ENTER	Return
14	SO — Desplazar hacia afuera	Encender minúsculas	Número	Encender modalidad página
15	SI — Desplazar hacia adentro	●	●	Apagar modalidad página
16	DLE — Escape enlace de datos	●	INK	Limpiar área gráficos
17	DC1 — Control del dispositivo 1	Cursor abajo	PAPER	Def. color texto
18	DC2 — Control del dispositivo 2	Reverse on	FLASH	Def. color gráficos
19	DC3 — Control del dispositivo 3	Cursor a base	BRIGHT	Def. color lógico
20	DC4 — Control del dispositivo 4	Delete	INVERSE	Restaurar ausencia color lógico
21	NAK — Recibido negativo	●	OVER	Desactivar unidades VDU
22	SYN — DLE sincrónico	●	AT	Seleccionar modalidad pantalla
23	ETB — Fin bloque transmisión	●	TAB	Reprogramar carácter visualiz.
24	CAN — Cancelar	●	●	Def. ventana gráficos
25	EM — Fin del medio	●	●	Plot m,x,y
26	SUB — Sustituir	●	●	Restaurar ausencia ventanas
27	ESC — Carácter de escape	●	●	Nulo
28	FS — Separador de archivos	Tecla rojo	●	Def. ventana textos
29	GS — Separador de grupo	Cursor der.	●	Def. origen gráficos
30	RS — Separador de registros	Tecla verde	●	Desplazar cursor texto
31	US — Separador de unidades	Tecla azul	●	Desplazar cursor texto a x,y

CODIGO ASCII	ASCII	CODIGO ASCII	ASCII
32	Espacio	80	P
33	!	81	Q
34	"	82	R
35	#	83	S
36	$	84	T
37	%	85	U
38	&	86	V
39	'	87	W
40	(	88	X
41	)	89	Y
42	*	90	Z
43	+	91	[
44	,	92	\
45	–	93	]
46	.	94	^
47	/	95	_
48	0	96	‖
49	1	97	a
50	2	98	b
51	3	99	c
52	4	100	d
53	5	101	e
54	6	102	f
55	7	103	g
56	8	104	h
57	9	105	i
58	:	106	j
59	;	107	k
60	<	108	l
61	=	109	m
62	>	110	n
63	?	111	o
64	@	112	p
65	A	113	q
66	B	114	r
67	C	115	s
68	D	116	t
69	E	117	u
70	F	118	v
71	G	119	w
72	H	120	x
73	I	121	y
74	J	122	z
75	K	123	{
76	L	124	!
77	M	125	}
78	N	126	~
79	O	127	Delete

entraña la utilización de un byte extra pero es más rápido). Esto demuestra que no hay ninguna forma *correcta* de construir un ordenador, sólo existe la manera en que lo hace el diseñador individual. También constituye un buen ejemplo de las cosas que los diseñadores de ordenadores deben tener en cuenta. Ellos saben que tienen que hacer una elección fundamental entre diseñar una máquina que sea lenta pero barata, o una que sea rápida pero cara. Del mismo modo, al escribir un programa en BASIC en máquinas con memoria limitada (el Vic-20 y ZX81 sin ampliar constituyen buenos ejemplos de ello), se debe encontrar un equilibrio entre velocidad de ejecución y eficacia en el uso de la memoria.

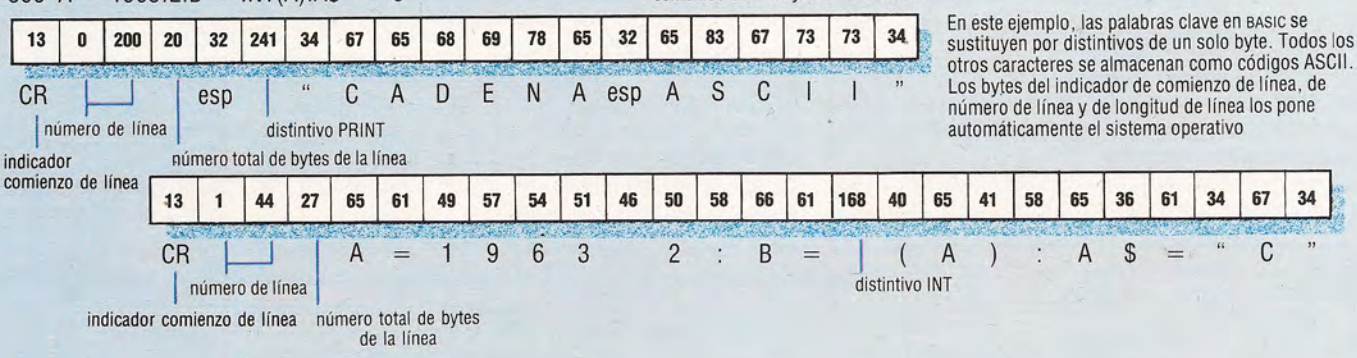
Por último, observe que en el área de textos habrá un indicador de comienzo de línea o final de línea para cada línea del programa en BASIC. En el BBC Micro cada línea empieza con un byte que contiene 13 (ASCII para Retorno de carro), mientras que en una línea del Spectrum éste se halla al final. En el Commodore, la línea de BASIC termina con un byte-cero (el ASCII de "").

Cómo se almacenan los programas en BASIC

Cada máquina posee sus variantes propias en cuanto a la forma en que almacena una línea de texto en BASIC. Observe las técnicas individuales para las líneas de texto que proporcionamos abajo

BBC Micro

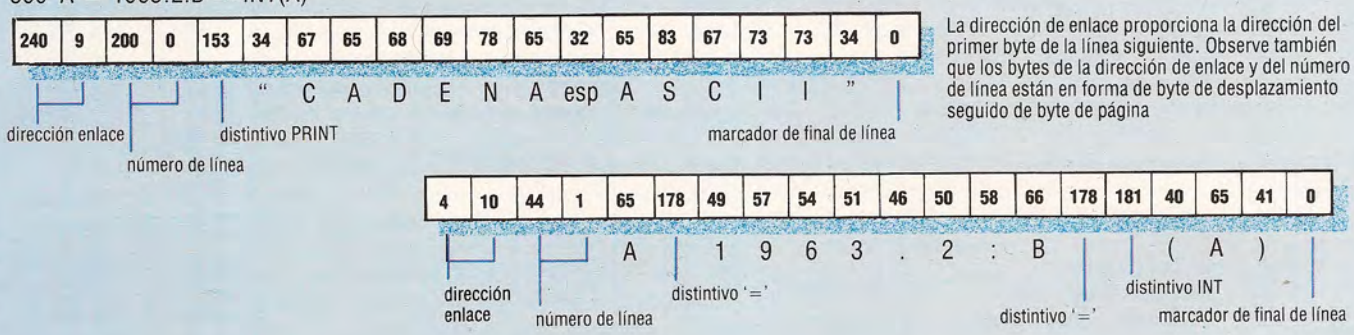
```
200 PRINT "CADENA ASCII"
300 A = 1963.2:B = INT(A):A$ = "C"
```



En este ejemplo, las palabras clave en BASIC se sustituyen por distintivos de un solo byte. Todos los otros caracteres se almacenan como códigos ASCII. Los bytes del indicador de comienzo de línea, de número de línea y de longitud de línea los pone automáticamente el sistema operativo

Commodore 64

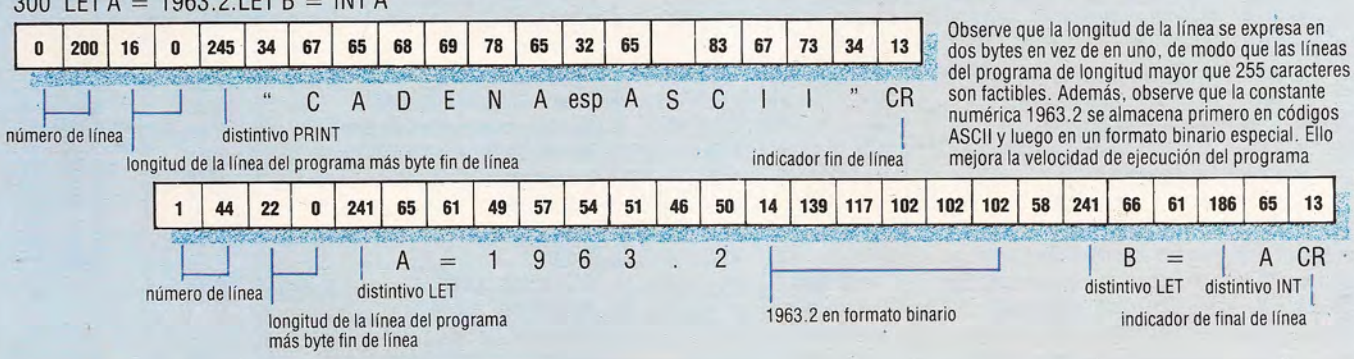
```
200 PRINT "CADENA ASCII"
300 A = 1963.2:B = INT(A)
```



La dirección de enlace proporciona la dirección del primer byte de la línea siguiente. Observe también que los bytes de la dirección de enlace y del número de línea están en forma de byte de desplazamiento seguido de byte de página

Spectrum

```
200 PRINT "CADENA ASCII"
300 LET A = 1963.2:LET B = INT A
```



Observe que la longitud de la línea se expresa en dos bytes en vez de en uno, de modo que las líneas del programa de longitud mayor que 255 caracteres son factibles. Además, observe que la constante numérica 1963.2 se almacena primero en códigos ASCII y luego en un formato binario especial. Ello mejora la velocidad de ejecución del programa

La fórmula del éxito

Creada a fines de 1982, Imagine Software ha sabido aprovechar el auge de los ordenadores personales. Del centenar de empleados con que cuenta, más de cuarenta escriben código de programación

Tal como ocurre en una empresa que triunfa, Imagine debe su éxito en el terreno del software a una mezcla de talentos. Los fundadores, David Lawson y Mark Butler, ambos de Liverpool, representan los dos elementos esenciales de cualquier empresa: la experiencia técnica y la agudeza comercial.

La especialidad de Imagine consiste en escribir software para las máquinas Commodore. En un momento dado, cuatro programas para juegos distintos (*Bewitched* [Hechizado], *Catcha snatcha* [Atrapar al ladrón], *Wacky waiters* [Locos camareros] y el tradicional *Arcadia*) escritos por Imagine para Commodore figuraban en la lista de los diez programas más vendidos. Los esfuerzos que Imagine dedicó a los ordenadores de la Commodore motivaron en 1983 la insólita distinción de ser invitada por ésta a que visitara la fábrica y las oficinas de Norristown (Pennsylvania) para que conociera, antes de su salida al mercado, las máquinas 264 y V364. Estas máquinas están destinadas a los amantes de los juegos y hasta ahora Commodore ha resistido a la tentación de elevar a sus productos en la escala de precios, con el fin de atraer al usuario pequeño empresario. A raíz de la gira, Commodore encargó a Imagine dos nuevos juegos, que se venderán bajo su propio cuño, lo que representa todo un triunfo para una empresa tan joven.

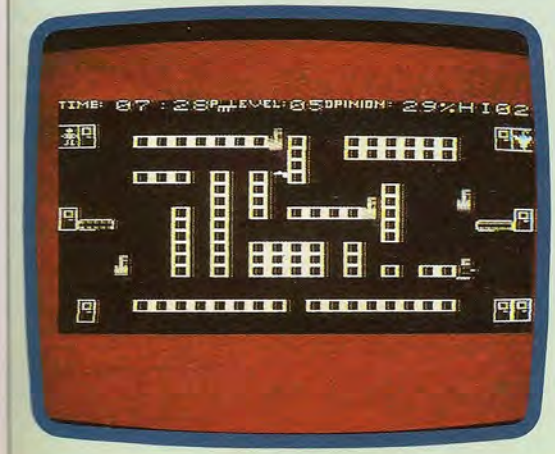
En Gran Bretaña, sin embargo, el que disfruta del apoyo de software más amplio es el Sinclair Spectrum. Pero las relaciones de Imagine con sir Clive Sinclair no han sido siempre inmejorables. Cuando Mark Butler y Dave Lawson se marcharon de Bug-Byte Software, donde habían pasado sus años de formación, programaron una visita a Sinclair, pero ésta acabó sin llegar a acuerdo alguno de trabajo. La política de Imagine es que sus juegos se puedan modificar para adaptarse a una nueva máquina sin que para ello se requiera mucho más que cambiar las posiciones de memoria de unos pocos trozos de código del programa.

El QL se basa en el procesador central Motorola 68000 y actualmente la mayoría de los programadores que poseen experiencia profesional con este chip ya están contratados por firmas profesionales de software o trabajan en los departamentos de informática de las universidades.

Con el fin de encontrar programadores para que trabajaran con el procesador 68000, Imagine puso anuncios para cubrir un buen número de puestos, pero en líneas generales los resultados fueron decepcionantes. Aunque hubo una masiva respuesta, eran muy pocas las personas cuya experiencia fuera superior a unos pocos meses. Muchas ni siquiera habían logrado completar la codificación de un solo juego. ¡No había materia prima para iniciar una nueva generación de proyectos de programación!

Parece ser que Imagine está ahora planeando una nueva estrategia de cara al mercado más lucrativo (y más competitivo) del software para gestión empresarial. En realidad no se trata tanto de una

Catcha snatcha (Atrapar al ladrón)
Escrito para el Commodore Vic-20, este juego de laberinto-persecución versa sobre un detective contratado para trabajar en unos grandes almacenes, que atrapa a ladrones, encuentra mercancías perdidas y entrega niños extraviados a sus padres



Bewitched (Hechizado)
Otro juego de laberinto, pero diferente en el sentido de que sólo se puede penetrar en sus corredores luego de abrir las cerraduras de sus puertas de acceso. Para complicar aún más las cosas, en realidad no todas las puertas se abren, la situación de éstas cambia al azar, y, además, hay fantasmas que acechan

Cortesía de Soft

táctica original, porque fue Apple quien inició los primeros contactos. Aunque en un principio la iniciativa sorprendió a Imagine, ésta pronto comprendió por dónde Apple estaba dirigiendo su software, con una representación en pantalla fácil de utilizar e imágenes pictóricas como "ventanas" e "iconos" enlazadas mediante el dispositivo de entrada del ratón, en donde la acción gráfica y la entrada no efectuada desde el teclado constituyen factores clave.

Imagine tenía un sólido conocimiento de la tecnología en ocho bits de Apple, porque había utilizado ordenadores Apple IIe para escribir programas desde el principio. En consecuencia, estaba en condiciones ideales de sacar partido de este contacto. Asimismo, como parte de un ejercicio de renovación instrumental, Imagine invirtió en varias máquinas Sage IV. El objeto de ello fue el de aumentar la velocidad y el potencial informático. Dave Lawson ha escrito bastante software original utilizado, a su vez, para escribir programas de juegos, que luego se compilan de forma cruzada (se compilan en una máquina y se emplean en otra) para el ordenador personal en cuestión. El Sage IV posee un disco de RAM de gran capacidad, cuyo valor no tiene precio para escribir en lenguaje ensamblador. De cara al tiempo de los programadores, es más económico gastar dinero en una máquina como el Sage, que tener a los programadores sentados mientras esperan programas compilados. Junto con el Sage se proporciona un p-System UCSD, Universidad de California en San Diego, una implementación del PASCAL para imitar al Lisa y al Mackintosh de Apple, que utilizan ambos PASCAL en el *back ground* (zona de baja prioridad en la memoria). Los términos *background* y *foreground* se aplican a máquinas que pueden ejecutar más de un programa a la vez. El programa "foreground" siempre tiene prioridad, pero cada vez que se presente una oportunidad se ejecutará el programa "background".

Como muchas otras casas de software, Imagine ha estado investigando el lenguaje para ordenadores que lleva el nombre más conciso: el C. Éste es uno de los lenguajes más versátiles y, aun más importante, más portátiles de todos los que existen para microordenadores. Su estructura modular lo hace ideal para desarrollar software de sistemas.

En la actualidad no existen muchas personas que posean experiencia tanto en C como en el microordenador Sage. Por eso Imagine está rastreando en las facultades universitarias y en otras casas de software, con el deseo de hallar más personas con experiencia en este campo.

El secreto del triunfo

¿Qué es lo que hace que un programa se convierta en un éxito? Mucho tienen que ver en ello las "corazonadas". Al principio, cuando Mark Butler y Dave Lawson juntaron sus fuerzas para lanzar Imagine, ya tenían una clara idea de lo que la firma donde habían trabajado, Bug-Byte Software, estaba comercializando con todo éxito. Se sentaban y se sometían a sesiones para inspirarse y crear ideas: intentaban ponerse mentalmente en el lugar del entusiasta de los juegos. Lo que es primordial. "Si éste fuera mi primer ordenador, que tendré por un mes, o por seis meses, ¿qué esperaría de este juego? ¿Por qué iba yo a jugar con él? ¿Durante cuánto tiempo jugaré? En un punto del juego, ¿me gustarán los diversos efectos sonoros o un flash de los gráficos?" El diseñador de juegos tiene que pensar como si él mismo tuviera la edad del cliente al que se destina el juego.

Ahora Imagine es demasiado grande como para que dos personas solas generen todas las ideas para los juegos, y actualmente cuenta con un equipo de ocho diseñadores de gráficos que trabajan en la animación y secuencia gráfica de las nuevas obras. Éstas luego son probadas por el personal de la casa, no por los programadores.

Por desgracia para las casas de software, una técnica que han aprendido demasiadas personas sin conocimientos específicos es cómo ahorrarse el desembolso por el software. La reproducción ilegal es un problema acuciante. Se ha calculado que por cada juego en cassette que se vende desde el mostrador se hacen siete copias ilegales, la inmensa mayoría de ellas por el simple trámite de la transferencia de audio (de cinta a cinta). Se pueden tomar medidas preventivas para eliminar o, al menos, reducir esta práctica, pero son tan caras que el costo extra repercutiría negativamente en el consumidor. Sin duda alguna, Imagine está perdiendo ingresos de esta forma, pero así y todo vende muchísimas cassettes y por el momento mantiene sus opciones.

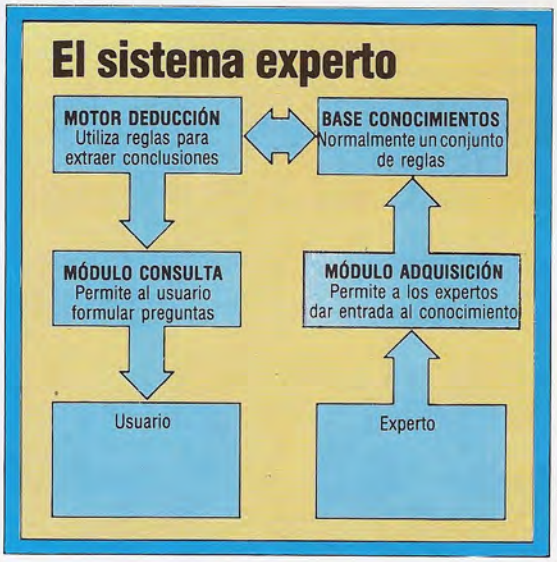
Puede que no pase mucho tiempo antes de que el cartucho y el disco sustituyan por completo a la cassette. Incluso podrían surgir procedimientos completamente nuevos para la distribución del software, en los cuales el código del programa se enviara desde un ordenador central, o anfitrión, y se cargara en el ordenador personal a través de las líneas telefónicas que se utilizan para la televisión por cable. Una firma de software que aspire a crecer en la década actual tiene necesariamente que estar atenta a las modificaciones de este tipo y, en este sentido, Mark Butler, de Imagine, está poniendo todo su talento al servicio de ello, sugiriendo nuevas posibilidades para los juegos empleando la síntesis de voz y manteniéndose al corriente de todas las innovaciones tecnológicas, como el disco láser.

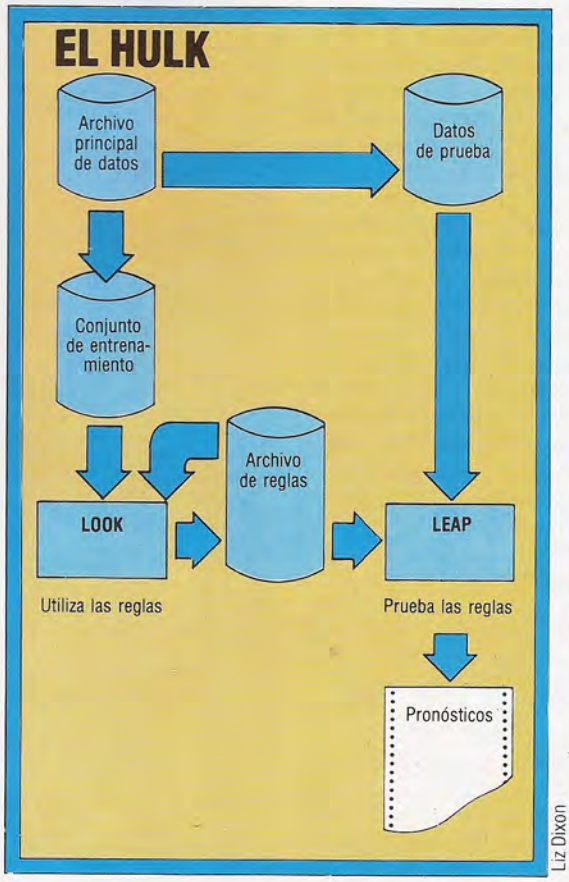
Cortesía de Imagine

Cuarteto "imaginativo"
La pujanza de cualquier firma de software radica en la composición adecuada del equipo de diseño de programas. Unidos específicamente para crear los juegos Psyclapse y Bandersnatch, en la foto aparecen, de izquierda a derecha: Ian Weatherburn, Mike Glover, John Gibson y Eugene Evans

El increíble HULK

Los sistemas expertos, en otros tiempos campo de acción exclusivo de los laboratorios de investigación, se encuentran ahora al alcance de los usuarios de ordenadores personales





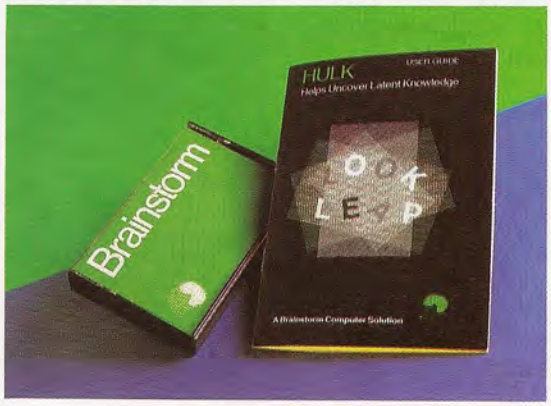
Liz Dixon

La eficacia del HULK
En un sistema experto tradicional, una serie de reglas IF...THEN, retenidas en la base de conocimientos, son aplicadas por el motor de deducción en respuesta a los interrogantes del usuario, que entra en el sistema a través del módulo de consulta. El conocimiento sobre el cual se basan estas reglas se introduce en el sistema experto a través del módulo de adquisición. Con el HULK se construye un conjunto de reglas de decisión a partir de un archivo principal de datos que contiene observaciones a las que da entrada el usuario. Se necesitan dos programas separados: el LOOK, que requiere de un conjunto de entrenamiento de datos para probar las reglas y de un conjunto de prueba de datos para verificar su utilidad, y el LEAP, que emplea el archivo de reglas y los datos de prueba para producir un pronóstico de probabilidades. El HULK se ha descrito como "el sistema experto del pobre"

El término "ingeniería" está experimentando un cambio en su significado. En el siglo XIX, un ingeniero era alguien como Brunel (1806-1859), que transformaba el hierro y el acero en barcos y puentes. En la actualidad han hecho suya esta palabra diversas profesiones de despacho que "diseñan" materias primas como el "conocimiento".

Un ingeniero de conocimiento es alguien que sabe cómo construir un *sistema experto* (SE). Un sistema experto sintetiza el conocimiento organizado acerca de algún campo de la experiencia humana. Consideremos, por ejemplo, el caso del diagnóstico médico: un médico posee un gran acopio de conocimientos basados en los hechos acerca de las enfermedades y de sus signos y síntomas; la experiencia del médico reside en la capacidad de relacionar el estado de un paciente con las descripciones de las condiciones típicas proporcionadas por el libro de texto. Al hacerlo, el médico determina qué síntomas están presentes y coteja su significado con el de los síntomas ausentes y/o la enfermedad ante la cual cree encontrarse. Cuanto mayor sea la capacidad del médico para combinar el conocimiento de los libros con las observaciones reales, tanto más exacta será la técnica de diagnóstico. Los factores limitativos son la capacidad de recordar datos organizados, la capacidad de relacionar casos observados con el patrón de los datos existentes y la capacidad de aplicar este conocimiento en casos en los que los datos son incompletos o no se adaptan muy bien a los casos anteriores. Estos dos primeros factores (la organización y la clasificación de datos) son el punto fuerte de los ordenadores; el último factor es el punto fuerte de los expertos humanos. Si un sistema informático puede sustituir la "percepción" o el "olfato" del experto humano por el análisis estadístico, entonces sus superiores poderes para la organización de datos podrían capacitarlo para superar al experto.

Los sistemas de este tipo son más relevantes allí donde sea necesario el juicio humano por no existir una teoría completa, como en el diagnóstico médico. También se pueden aplicar provechosamente allí donde, aun cuando esté disponible a nivel público el conocimiento requerido, éste sea demasia-



Liz Heaney

Proyecto práctico
Al principio Richard Forsyth concibió el HULK como un plan práctico para sus alumnos del Polytechnic of North London. Si bien escribir el programa sólo le ocupó dos semanas, se necesitaron seis meses para mejorarlo y refinarlo con el fin de hacerlo más accesible al usuario. Es posible que pronto aparezca una versión para el QL

do complicado como para que lo apliquen la mayoría de las personas (p. ej., la legislación acerca de la declaración de la renta o las disposiciones que regulan el derecho a disfrutar de los beneficios de la Seguridad Social). Debido a que para la operación de este tipo de sistemas el conocimiento, frecuentemente expresado como un conjunto de reglas, resulta esencial, también se los conoce como sistemas *basados en reglas* o *basados en el conocimiento*.

Los sistemas expertos han logrado un éxito sorprendente en varios campos. Ya pueden superar en rendimiento a los profesionales humanos experimentados en diagnóstico médico, en prospección de minerales o en muchos otros campos. En virtud de estos notables éxitos, estos sistemas están siendo creados en gran número por los laboratorios de investigación de inteligencia artificial y encontrando aplicaciones en la práctica de la informática general, con importantes consecuencias por la forma en que la gente está construyendo sistemas informáticos de elevado rendimiento. De cara al diseño de software, el enfoque basado en el conocimiento reemplaza la tradición de:

INFORMACION + ALGORITMO = PROGRAMA

por una metodología basada en:

CONOCIMIENTO + DEDUCCION = SISTEMA

que es un cambio evolutivo de consecuencias muy significativas.

De manera que un sistema experto se fundamenta en una *base de conocimientos*. En un SE totalmente desarrollado existen en realidad otros tres componentes básicos: el *motor de deducción*, que forma nuevas reglas para interpretar los datos sobre la base de las reglas existentes y los datos; el *módulo de adquisición de conocimientos*, a través del cual el sistema adquiere conocimientos a partir de su propia experiencia y la de los expertos humanos, y la *interface para el usuario*, que permite que los no expertos tengan la posibilidad de interrogar al sistema y de utilizarlo.

Una base de conocimientos contiene hechos (o aseveraciones) y reglas. Los hechos pueden cambiar rápidamente; por ejemplo, durante el curso de una consulta. Las reglas son la información a más largo plazo acerca de cómo generar nuevos hechos o hipótesis a partir de lo que se conoce actualmente. ¿En qué difiere esto de una base de datos convencional? La diferencia fundamental estriba en que una base de conocimientos es más creativa. Los hechos de una base de datos normalmente son pasivos: o están o no están allí, para ser archivados o recuperados por el usuario en la medida en que éste así lo requiera. Una base de conocimientos, por el contrario, intenta rellenar activamente la información que falta, a la luz de lo que ya "sabe" e independientemente de las exigencias inmediatas de datos.

Las reglas que responden al formato IF...THEN (conocidas como "reglas de producción") son un método válido para expresar el conocimiento de "reglas empíricas". Por ejemplo:

IF (si) el equipo local perdió el último partido jugado en su campo, AND (y) el equipo visitante ganó por dos goles el último partido jugado en su campo
THEN (entonces) la probabilidad de un empate se multiplica por 1 088.

Pronóstico meteorológico

Los archivos de datos del **HULK** los crea el usuario como archivos de programas en BASIC, por lo que resultan fácilmente accesibles para la inspección y edición. El archivo que utilizamos es típico.

Ejecutamos el **HULK** empleando este archivo y se nos solicitó una hipótesis acerca de los datos; la hipótesis fue:

MAÑANA = 1

con lo que queremos significar que nos interesan aquellas muestras en las que la última variable posee el valor 1; en otras palabras, los días en que llovió al día siguiente. Deseamos utilizar el **HULK** para establecer reglas que nos permitan predecir el tiempo que hará mañana a partir del tiempo que hace hoy.
Seguidamente estamos en condiciones de dar entrada a reglas para prueba. Cada regla se aplica al archivo de datos y el éxito que obtiene en cuanto a predecir lluvia se mide y se informa, tanto como regla única como cuando se utiliza en combinación con reglas anteriores. El **HULK** aconseja acerca de si agregar o no la regla al conjunto de reglas, pero la decisión en este sentido queda en manos del usuario.

Rápidamente encontramos tres reglas para pronosticar lluvia para mañana:

1) Si hoy llueve más de 2 mm
AND (y)
2) Si hoy el sol brilla menos de 3,5 horas
AND (y)
3) Si la temperatura máxima es superior en menos de 6 grados a la temperatura mínima de hoy
THEN (entonces)
uno puede tener una certeza de un 83 % al pronosticar lluvia para mañana (siempre que, por supuesto, los datos del tiempo relativos a este mes sean una muestra representativa de los de todo el año).

Aunque éste es un archivo de programas en BASIC, no está pensado para ejecutarse; es simplemente una forma conveniente de almacenar datos.

```
1 TIEMPO,30,5
```

La línea 1 describe el archivo: contiene su nombre (TIEMPO), el número de muestras de datos (30) y el número de datos (5) por muestra. El archivo contiene los datos del tiempo para un mes: cada día se describe en lo que se refiere a temperatura mínima y máxima (en centígrados), lluvia diaria (en mm), horas de sol y una variable booleana cuyo valor es 1 si lloverá al día siguiente y 0 si no lloverá.

```
100 TMINIMA
200 TMAXIMA
300 LLUVIA
400 SOL
500 MANANA
```

Desde la línea 100 a la 500 se proporcionan los nombres de variables para los datos de cada muestra

```
1001 D01,54,110,175,32,1
1002 D02,42,125,041,62,1
1003 D03,75,112,077,11,1
1004 D04,27,105,018,43,0
1005 D05,30,120,000,95,0
1006 D06,44,106,000,55,0
1007 D07,48,094,000,51,1
1008 D08,68,092,035,48,1
1009 D09,64,102,043,1,1
     ----- -----
1028 D28,58,154,000,20,1
1029 D29,67,093,054,42,0
1030 D30,45,095,000,58,1
```

Entre las líneas 1001 y 1030 se incluyen los datos, con una muestra diaria por línea del programa, y empezando cada línea con una etiqueta que identifica esa muestra. Los valores del HULK han de ser números enteros, así que todos los datos se han multiplicado por diez para conservar la precisión:
La línea 1001, por ejemplo, en realidad debería decir:
1001, D01,5.4,11.0,17.5,3.2,1 y así sucesivamente para el resto del archivo

MAÑANA = 1

SOLICITANDO REGLA **1**

LA REGLA ES: **LLUVIA > 20**

TABLA DE CONTINGENCIA	ÉXITO	FRACASO
REGLA VERDADERA	**12**	**2**
REGLA FALSA	**5**	**11**

PROMEDIO DE ÉXITO = **76,3 %**

BITS POR MUESTRA

ANTES	**LLUVIA > 20**	**1,00**
DESPUÉS	**LLUVIA > 20**	**0,85**

SE LE ACONSEJA [CONSERVAR] LA REGLA

LA REGLA [SE AÑADE] AL CONJUNTO DE REGLAS

Bits por muestra
Esta es una medida de cuántas de las muestras de datos están contribuyendo al éxito de la regla

Mike Brownlow

Para inferir deducciones existen dos estrategias de alto nivel fundamentales: la "encadenación hacia adelante" y la "encadenación hacia atrás". En términos generales, la encadenación hacia adelante implica examinar los datos por orden para formular hipótesis, mientras que la encadenación hacia atrás intenta hallar datos para demostrar o refutar una hipótesis ya formulada. La encadenación hacia adelante pura conduce a un cuestionario descentrado del sistema del tipo "¿Y si...?", mientras que la encadenación hacia atrás pura suele ser inexorable con su interrogatorio dirigido a un objetivo.

La mayoría de los sistemas de éxito utiliza una mezcla de ambas estrategias. Si un procedimiento deductivo trabaja básicamente hacia atrás o hacia adelante, habrá de tratar con datos inciertos. Los especialistas en ordenadores han intentado simplificar y comprimir el mundo en que vivimos con el fin de delimitarlo y hacerlo compatible con los confines rígidos del ordenador, y este cometido nunca ha sido fácil. Ahora la investigación en sistemas expertos nos ha proporcionado un medio para tratar con lo incierto; en otras palabras, con el mundo real en vez de con alguna abstracción idealizada que nuestro sistema de datos nos obligue a utilizar.

En realidad tenemos demasiadas formas de tratar con lo incierto. Están la lógica bayesiana, la lógica polivalente y los factores de certidumbre, por nombrar sólo tres: estas lógicas reemplazan la certeza de "IF (SI) X ES VERDADERA, THEN (ENTONCES) Y ES VERDADERA" por la prudente deducción estadística de "IF (SI) X ES VERDADERA EL 65 % DE LAS VECES, THEN (ENTONCES) LAS PROBABILIDADES DE Y OSCILAN ENTRE EL 50 Y EL 70 %. Se han probado toda clase de esquemas, y lo curioso es que todos parecen funcionar. Una explicación para este hecho podría basarse en que la organización del conocimiento importa más que los valores a él asignados. La mayoría de las bases de conocimiento incorporan la redundancia para que el sistema experto llegue a conclusiones correctas por varios caminos.

Los paquetes de software diseñados para facilitar el acceso del usuario al diseño de sistemas basados en el conocimiento están apareciendo ya en el mercado. En Gran Bretaña se puede adquirir el HULK (*Helps Uncover Latent Knowledge*: ayuda a descubrir el conocimiento latente), que comercializa Brainstorm Computer Solutions. Sólo funciona en el BBC Modelo B y en los ordenadores Torch; no obstante, es muy probable que aparezca una versión para el QL, de Sinclair.

El HULK permite que el usuario elabore y verifique un conjunto de reglas de decisión, que después se pueden usar para predicción o clasificación.

Por ejemplo, supongamos que un agricultor ha realizado en un ordenador personal mediciones detalladas acerca de la altura, el color de las hojas, etc., de varios centenares de naranjos, registrando, además, aquellos árboles que sufrieron alguna afección antes de la época de la recolección y aquellos que se conservaron sanos. El sistema se podría utilizar para desarrollar reglas que relacionaran las características del árbol con su estado de salud. Luego esas reglas podrían identificar a los naranjos en peligro y, por consiguiente, mejorar las expectativas para la recolección del año siguiente. Este agricultor quizá no sabría nunca cuáles eran esas reglas, pero el sistema las aplicaría a los datos relativos a los árboles tal como se le proporcionaron.

Por otra parte, usted podría retener información acerca de los resultados de los partidos de fútbol de la temporada y desarrollar un procedimiento para catalogar los diferentes encuentros como probables triunfos, empates o derrotas, sobre la base de diversos indicadores conocidos antes del saque inicial. Los datos necesarios para tomar decisiones, como el rendimiento reciente del equipo local, los resultados previos de este encuentro, las posiciones de los equipos en la tabla de clasificación, todo está fácilmente disponible. Pero si no se contara con la ayuda de un SE se ocuparía mucho tiempo en organizar esta información y sería difícil correlacionarla. Existen muchas aplicaciones para el HULK: todo cuanto necesita es un conjunto de datos.

El paquete HULK consta de dos programas principales: el LOOK (*Logical Organiser of Knowledge*: organizador lógico de conocimiento) y el LEAP (*Likelihood Estimator and Predictor*: estimador y pronosticador de probabilidades). Éstos permiten que el usuario desarrolle un conjunto de reglas a partir de un archivo de datos de observaciones retenido en disco (p. ej., los resultados de los encuentros anteriores y otros factores de decisión) y luego aplicar ese conjunto de reglas a otro archivo de datos de observaciones incompletas (p. ej. los factores de decisión para los partidos del próximo domingo) con el objeto de hacer pronósticos respecto a los mismos. El conjunto de reglas es la base de conocimientos: expresa el conocimiento del sistema acerca de los datos. En el HULK, consiste en reglas de decisión en cuanto a lo probable, que se pueden usar para clasificación o predicción.

La base de conocimientos crece a través de la interacción del usuario y el ordenador: las reglas las propone el usuario y las verifica el ordenador; el usuario descarta aquellas que no mejoran el rendimiento global del sistema.

Para utilizar el LOOK uno necesita un conjunto de datos o, mejor aún, dos (un gran conjunto de datos se puede dividir en dos partes con este fin). Uno se denomina *conjunto de entrenamiento* para probar las reglas, y el otro recibe el nombre de *conjunto de prueba* para confirmar su utilidad sobre datos no vistos.

Una vez los datos están en el archivo, se emplea el LOOK para probar sus ocurrencias mediante la proposición de nuevas reglas, de a una cada vez. Hace pasar cada regla por el conjunto de entrenamiento y le dice en cuánto mejora (si es que lo mejora) el marcador de predicciones de las reglas ya presentes. Luego aconseja si es conveniente conservar o descartar la nueva regla. La decisión final queda en manos del usuario.

Después de crear las reglas mediante el LOOK, se puede utilizar el LEAP para aplicarlas a muestras desconocidas. Las reglas se combinan para dar una estimación de una única probabilidad para cada muestra del conjunto de datos de prueba. La salida del LEAP incluye una lista de muestras ordenadas de acuerdo a su probable resultado. Lo que pueda ser este resultado (MARCADOR EMPATE, o ALTO RENDIMIENTO DE ESTE NARANJO) depende de la naturaleza de los datos de muestra y de lo que el usuario intentaba pronosticar mediante los mismos. El HULK proporcionará la configuración del módulo de adquisición y la base de conocimientos, dejándole al usuario el control del motor de deducción.

Ampliación flexible

El sistema de ampliación del BBC Micro ha mejorado aún más gracias a que la Acorn ha diseñado un nuevo DOS y un controlador para unidades de disco

En la actualidad, virtualmente todas las unidades de disco compatibles con el BBC Micro son minifloppies estándar de 5 ¼ pulgadas, unidades de capacidad simple o doble, pero están apareciendo unidades de 8 pulgadas. Lamentablemente, el DFS no es una pieza estándar en los ordenadores BBC y se debe comprar por separado a Acorn. Se vende en forma de un chip de ROM que encaja en un conector del tablero de circuito impreso del ordenador. Otros fabricantes ofrecen chips DFS alternativos por un precio algo inferior al de Acorn.

Debido a que el DFS está contenido en ROM, la mayoría de las órdenes de disco no requieren memoria interna para llevar a cabo las instrucciones. El DFS se amplía mediante órdenes y rutinas proporcionadas como "utilidades" en un disco que viene con la unidad.

Las unidades de disco se acoplan a través de un cable plano a un conector de 34 vías (que lleva la indicación *disk drive*) en la cara inferior del ordenador. Este cable transporta todos los datos del disco a y desde las unidades, tanto dobles como simples. El control de las unidades de disco se ejerce a través de un chip controlador de disco 8271 que convierte los datos de ocho bits en paralelo del ordenador a la forma en serie para salir a la unidad de disco seleccionada, y viceversa. Se admiten cuatro estándares de formato: 40 pistas u 80 pistas en una sola cara y 40 u 80 pistas en doble cara, siempre a densidad simple. Cada pista se divide en 10 sectores, que contiene cada uno 256 bytes, dando una capacidad total de unos 100 Kbytes por cara en un formato de 40 pistas y 200 Kbytes por cara en el de 80 pistas.

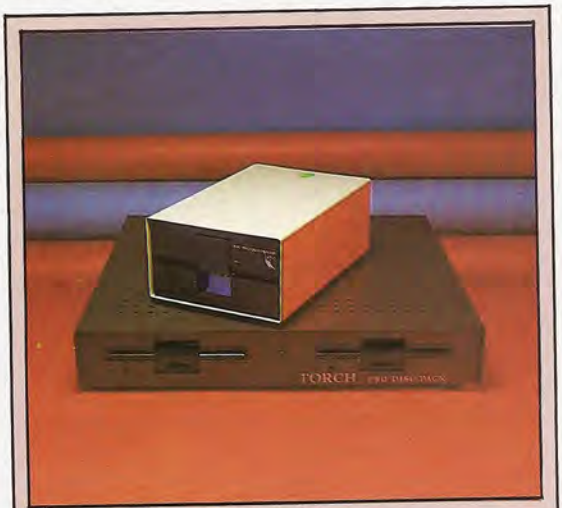
Las unidades se identifican mediante un número; una única unidad es 0 y una segunda unidad es 1. En el caso de las unidades que emplean las dos caras de un disco, ambas caras se tratan como unidades separadas, numeradas 0 y 2. Una segunda unidad numerará sus dos caras 1 y 3.

El catálogo del disco (o cara) ocupa dos Kbytes y está contenido en los dos primeros sectores de la primera pista. Dicho catálogo retiene información acerca del nombre y los identificadores del disco y se puede dividir en un máximo de 27 directorios seleccionables independientes con las listas de los archivos correspondientes, de manera que estos últimos se pueden almacenar por categoría. No hay previsto ningún mapa de disponibilidad de bloques (BAM); en cambio, hay una orden *COMPACT que localiza cualquier hueco que hubiera dejado algún archivo borrado y reacomoda el almacenamiento de los archivos por orden, dejando todo espacio libre después del final del último archivo.

Los archivos de programas y de datos se pueden guardar en disco del mismo modo que en cinta. De hecho, con el DFS todas las órdenes para manipulación de cassettes se dirigen automáticamente al disco a partir del encendido. Para utilizar una cassette, con el DFS acoplado, dé entrada a la orden RETURN — * TAPE. Para reasignar el disco dé entrada a: RETURN — * DISK. Además de las órdenes para manipulación de archivos estándar, el DFS facilita órdenes y rutinas para manipulación de datos, incluido un sistema archivador de acceso aleatorio, órdenes HELP y mensajes de error.

Al igual que el excelente BASIC BBC, el DFS es un poderoso dispositivo para manipular datos. Incluye muchas rutinas que por lo general se consideran fuera del alcance de los sistemas operativos en disco para ordenadores personales y, además de ello, son fáciles de utilizar. La estructura del sistema fomenta la administración eficaz del disco y la transferencia de datos es rápida. Las cifras varían de un fabricante a otro, aunque, por término medio, la carga de un archivo de programas de 20 Kbytes tarda alrededor de cinco segundos. El inconveniente del sistema, aparte de su costo relativamente elevado, es que el DFS sólo puede almacenar 31 nombres de archivo para cada cara de disco. Habida cuenta de la posible capacidad de 200 Kbytes por cara y la flexibilidad que caracteriza al sistema de directorio, esto representa un auténtico inconveniente.

Exclusiva y automática
La Hobbit constituye un sistema exclusivo de cinta flexible diseñado para el BBC Micro. Al estar completamente controlada por software, todas las funciones de avance, rebobinado, reproducción y grabación se realizan de forma automática



Las unidades de disco BBC
Para sacar el máximo partido de un BBC Micro es esencial una unidad de disco. Las dos aquí ilustradas son de las más populares para el Modelo B: la Acorn 100 y la Torch Z80. Para poderlas utilizar, primero se deben conectar en interface con el ordenador a través de unidades ROM de sistema operativo en disco, que se instalan en el interior de la máquina

Ian McKinnell

Órdenes del disco BBC

Siempre que sea necesario enunciar el archivo con el cual esté relacionada una orden, la especificación completa es:

COMMAND:DV.DR.NOMARCHI

donde COMMAND es la orden para manipulación de archivos requerida; DV es el número de unidad de disco (0-3); DR es el identificador de directorio (A-X o $); y NOMARCHI consta de un máximo de siete caracteres que identifican el archivo (no se deben incluir '#', '*', '.' ni ':'). De no especificarse el número de unidad ni el identificador de directorio, la unidad por omisión es 0 y el directorio por omisión es $. Los identificadores por omisión se pueden cambiar utilizando las órdenes *DRIVE, *DIR & *LIB. El DFS permite asimismo el empleo de estos caracteres "wildcard": '#' y '*'. En algunas órdenes éstos se pueden utilizar para especificar todas las unidades, todos los directorios o todos los nombres de archivo que empiecen con la misma letra. En las explicaciones que proporcionamos, la anterior especificación para archivos se abrevia como <FSP>. Además de las órdenes para cassette estándar disponibles para el disco desde el momento del encendido, y de las órdenes de acceso directo, el DFS facilita estas órdenes para manipulación de discos:

***FORM40, *FORM80 y *VERIFY**
Estas órdenes están almacenadas como utilidades en un disco que se proporciona con la(s) unidad(es) de disco; dan formato a un disco de 40 u 80 pistas e informan acerca del éxito de la operación.

ACCESS
*ACCESS <FSP> L protege al archivo de un borrado o sobreescritura. *ACCESS <FSP> anula esta protección.

***BACKUP, *DESTROY y *ENABLE**
*BACKUP FuenteDV Destino DV copia el contenido completo del disco de la unidad fuente en el disco de la unidad de destino, sobreescribiendo, por consiguiente, el disco de destino. *DESTROY <FSP> elimina un archivo. De haber *wildcards* (tarjetas ajenas) incluidas en <FSP>, mediante una sola orden se podrían borrar varios archivos. Dado que BACKUP y DESTROY son tan drásticas en cuanto a sus efectos, se debe dictar una orden *ENABLE para que el DFS las obedezca.

***BUILD <FSP>**
Crea un archivo ASCII con la especificación de archivo a partir de todas las subsiguientes entradas desde el teclado hasta acabar con la tecla ESCAPE.

***CAT DV**
Visualiza el catálogo de la unidad de disco especificada.

***COMPACT DV**
Desplaza los espacios libres de un disco de la unidad especificada hasta el final del último archivo, en un bloque continuo.

***COPY**
Copia el o los archivos especificados (utilizando un nombre de archivo wildcard) de un disco a otro.

***DELETE <FSP>**
Borra el archivo individual especificado del catálogo de un disco. El archivo puede sobreescribirse después.

***DIR DR**
Establece el directorio por omisión corriente en el directorio especificado. Todos los archivos subsiguientes almacenados mediante *SAVE o SAVE se le asignarán al directorio establecido.

***DRIVE DV**
Establece la unidad por omisión corriente.

***DUMP <FSP>**
Visualiza un listado en hexadecimal del archivo especificado.

***EXEC <FSP>**
Lee todos los datos de un archivo como si a éste se le hubiera dado entrada desde el teclado. Es útil para ejecutar una secuencia de órdenes que se emplee con frecuencia. Los archivos que se pueden leer mediante *EXEC se crean mediante *BUILD.

***HELP**
Referida a la operación de la unidad de disco, *HELP DFS visualiza una lista parcial de las órdenes DFS estándar y de su construcción, y *HELP UTILS visualiza una lista de las demás órdenes DFS estándar.

***INFO <FSP>**
Visualiza información extra relativa al o a los archivos especificados (utilizando *wildcards*) no visualizados mediante *CAT, tales como: posición de memoria, dirección de ejecución, longitud en bytes y localización de sector.

***LIB:DV.DR**
Establece el directorio especificado como la "biblioteca" (en inglés, *library*). Permite la utilización de una forma breve de orden (*NOMARCHI) que busca el directorio corriente de la biblioteca para el programa en lenguaje máquina mencionado, lo carga en la memoria y lo ejecuta inmediatamente como si se hubiera empleado la orden *RUN completa.

***LIST <FSP>**
Visualiza el archivo ASCII especificado, incluyendo los números de línea.

***LOAD <FSP>**
Lee el archivo especificado en la memoria, en las posiciones de las que se tomara originalmente.

***OPT 1**
Instrumenta un sistema de mensajes en el que se visualiza cada vez que se accede a un archivo la información dada por *INFO. Esta facilidad se logra mediante *OPT 1 1. Para inhabilitar esta configuración, utilice *OPT 1 0.

***OPT 4**
Modifica la opción de comienzo automático al encenderse o con (SHIFT) BREAK para la unidad de disco corriente seleccionada, donde: *OPT 4 0 desactiva el arranque automático; *OPT 4 1 carga (LOAD) el archivo !BOOT; *OPT 4 2 ejecuta (RUN) !BOOT; y *OPT 4 4 ejecuta (EXEC) !BOOT.

***RENAME <FSP viejo> <FSP nuevo>**
Esta orden modifica el nombre de un archivo y lo desplaza a un directorio diferente. No desplaza archivos de una unidad a otra.

***RUN <FSP>**
Lee un archivo en código de lenguaje máquina en la memoria y lo ejecuta inmediatamente. Se utiliza con archivos que no estén contenidos en la biblioteca corriente.

***SAVE**
Copia un bloque especificado de la memoria del ordenador y lo escribe en disco en la unidad y el directorio corrientes. Se construye así:

```
*SAVE "NOMBRE" CCCC FFFF EEEE
RRRR
```

o bien

```
*SAVE "NOMBRE" CCCC + LLLL EEEE
RRRR
```

donde CCCC es la dirección de comienzo del bloque de memoria; FFFF es la dirección de final del bloque de memoria; EEEE es la dirección de ejecución del programa almacenado; RRRR es la dirección a la cual se le leerá el programa; y LLLL es la longitud del archivo expresada en bytes (opción para FFFF). Todos los números se dan en hexadecimal. RRRR y EEEE se pueden omitir, y las direcciones de recarga y ejecución pasan, por omisión, a CCCC.

***SPOOL <FSP>**
Abre el archivo especificado para recibir toda la información visualizada como un archivo de textos. Permite almacenar un programa en BASIC como un archivo ASCII, en lugar de distintivarlo.

***TITLE "NOMBRE DISCO"**
Cambia el nombre del disco en la unidad corriente por el nombre especificado.

***TYPE <FSP>**
Visualiza un archivo ASCII excluyendo los números de línea.

***WIPE <FSP>**
Es idéntica a *DESTROY, con la excepción de que *ENABLE no es necesaria.

La gran prueba

La etapa final de toda operación de ensamblaje es ponerlo a prueba, pero esto no incluye conectarlo, pues podría destruirse uno de los dispositivos más delicados

El primer paso para probar cualquier ensamblaje de componentes en un circuito en funcionamiento se refiere a la eficacia de las junturas soldadas. Una juntura efectuada a una temperatura demasiado baja puede parecer aceptable exteriormente, pero por dentro quizá no establezca conexión alguna. Un suave tirón permitirá comprobar adecuadamente este extremo: en el caso de que la juntura esté realizada de forma correcta no se le separará entre las manos; por otra parte, en el supuesto de que se separe, es mejor que la juntura fracase en esta etapa que más adelante.

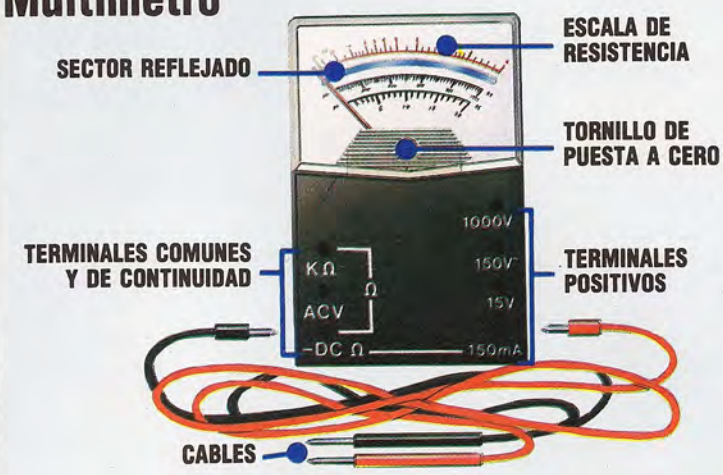
Ahora estamos preparados para aplicar algún *tester*. Uno de los más sencillos se puede comprar en las ferreterías o tiendas de recambios para automóviles. Se emplean para determinar cuándo los puntos de contacto del distribuidor están abiertos y cerrados.

Sin embargo, una alternativa mejor es la que representa un pequeño multímetro. En su papel de ohmiómetro puede probar no sólo la continuidad sino también la resistencia. La unidad de resistencia se denomina *ohmio* en honor del físico alemán del siglo XIX Georg Ohm (1789-1854), que fue quien descubrió el fenómeno. La resistencia es una función de la superficie de la sección transversal de un cable que transporta una corriente, pero también se puede crear artificialmente mediante la utilización de unos componentes llamados *resistencias*. Para nuestra tarea, una juntura efectuada correctamente ofrecerá una resistencia despreciablemente pequeña al paso de la pequeña corriente empleada por el medidor o *tester* de continuidad, y, como resultado de ello, la luz se encenderá y brillará con intensidad, o bien, la aguja del dial se ladeará por completo.

Toda reacción inferior indica que la juntura es pobre y se debe realizar otra vez.

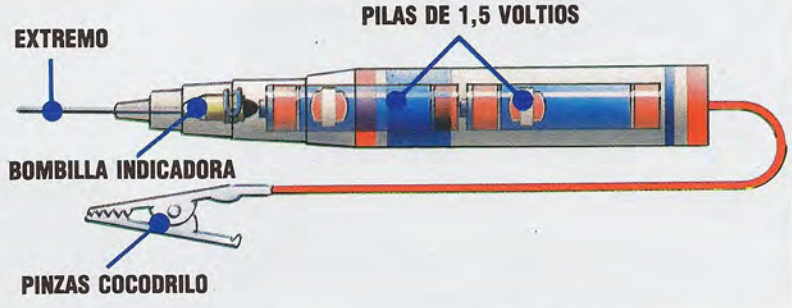
Además de medir la conductividad, el multímetro posee otras dos funciones: la medición de la corriente en amperios y del potencial eléctrico en voltios. Estas dos unidades guardan una estrecha relación entre sí: en efecto, la diferencia de potencial existente entre dos puntos de un circuito que transporta un amperio de corriente y gasta un vatio de energía es un voltio.

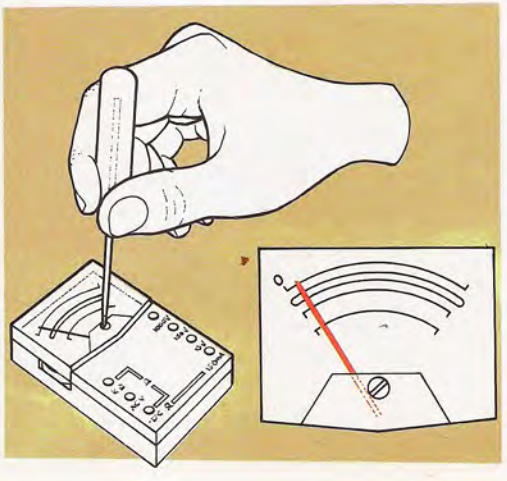
Multímetro



Típicamente, los multímetros prueban la continuidad y la resistencia, medida en ohmios; la magnitud del flujo de corriente, medida en amperios (amps) y la cantidad de potencial (en ocasiones llamada *tensión*), medida en voltios. El precio de los multímetros varía. No obstante, sólo hay dos métodos de representar sus resultados: analógico o digital. En general, los instrumentos de bobina móvil que visualizan sus datos mediante el movimiento (deflección) de una aguja a través de una escala de forma análoga al aumento o la disminución del valor que se está midiendo, son considerablemente más baratos que las versiones digitales

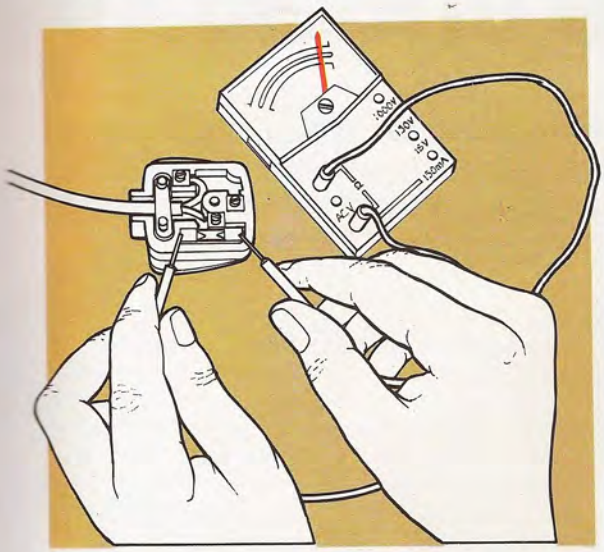
Tester de circuitos





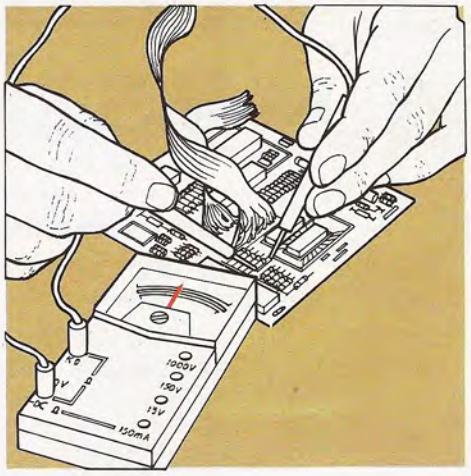
El medidor puesto a cero
Los medidores analógicos, que requieren que la aguja indicadora se mueva físicamente a través de un dial, deben disponer de algún medio de ajuste. Normalmente asume la forma de un tornillo, montado en el punto de apoyo de la aguja. Un sector reflejado por detrás de la aguja le permite al observador estar seguro de una lectura exacta. Además, el circuito de medición de resistencia también debe ser ajustable, para registrar otras variaciones e imprecisiones

Continuidad

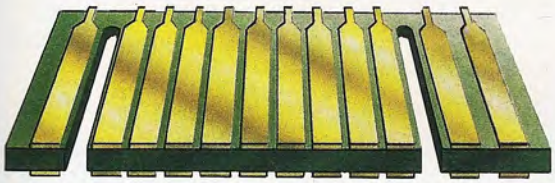


Continuidad y resistencia
Uno de los ejemplos más comunes de fallo de continuidad inducido y deliberado es el del fusible existente en todos nuestros enchufes de 13 amperios con tres patillas. El fusible está diseñado para quemarse y romper el circuito si se llegara a sobrecargar y, en ausencia de un *tester* de continuidad, el remedio normal cuando un aparato no funciona consiste en sustituir el fusible viejo por uno nuevo. Pero, ¿y si no era el fusible? El *tester* de continuidad nos lo dirá enseguida. Además, podemos utilizar un multímetro para medir el valor de una resistencia

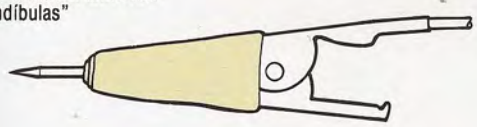
Resistencia



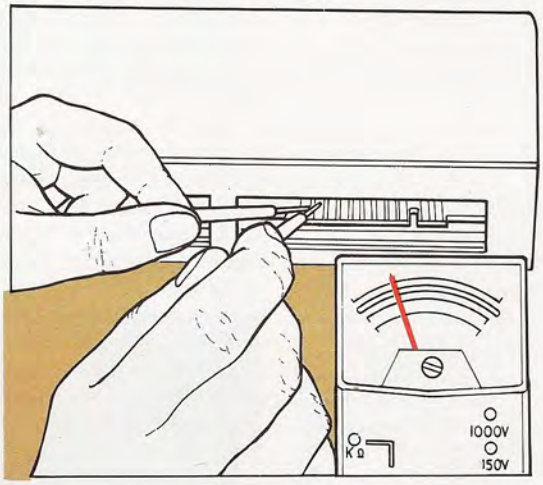
Flanco de subida
Los conectores terminales como el que vemos en la ilustración son el medio físico mediante el cual el ordenador se conecta con sus periféricos. Algunas máquinas, como el Spectrum y el ZX81, poseen una sola de estas puertas. Otras, como el BBC Micro, poseen varias. Los conectores terminales son sólo un tipo de puerta de E/S, pero son populares porque forman parte integrante del tablero de circuito impreso



Pinzas cocodrilo
Las pinzas cocodrilo, bastante toscas, se pueden convertir en un cable más refinado pegando con cinta aislante un trozo de hierro de soldar entre sus "mandíbulas"

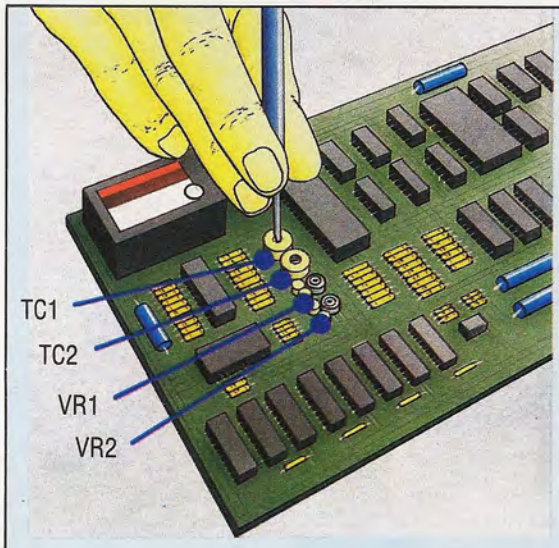


Voltaje
Un buen ejemplo del empleo del multímetro como *tester* de voltaje consiste en localizar la fuente de alimentación eléctrica del conector marginal de su ordenador. Consulte su manual y ponga la patilla de 0 voltios y la de +5 o +9 voltios. Ponga en contacto el cable negativo del medidor con 0 voltios y el positivo con +5 o +9 voltios. La aguja le señalará con exactitud cómo están la regulación de voltaje y el sistema de circuitos de control del interior de su ordenador y su fuente de alimentación eléctrica



ATENCIÓN

la garantía de su ordenador personal (en caso de que aún estuviera en vigor) podría anularse si alguien que no fuera el fabricante o su agente autorizado abriera la carcasa



Sintonice su Spectrum

Como observábamos en la p. 530, los usuarios de los primeros Spectrum pueden mejorar la pobre calidad de imagen en el televisor. Las dos resistencias variables que vemos (VR1 y VR2) controlan respectivamente el equilibrio rojo-verde y azul-amarillo. TC1 (un condensador de ajuste) y su compañero TC2 controlan la claridad de los caracteres y la intensidad del color.
La carcasa del Spectrum está sujeta con cinco tornillos estrella visibles situados en la cara inferior (tome nota del aviso sobre la caducidad de la garantía); para dejar al descubierto el tablero basta con quitar estos tornillos.
Los usuarios de la versión 3 del Spectrum no pueden realizar este ajuste ni ningún otro. Un Spectrum versión 3 se puede identificar por el disipador, placa de aluminio visible a la izquierda del conector terminal

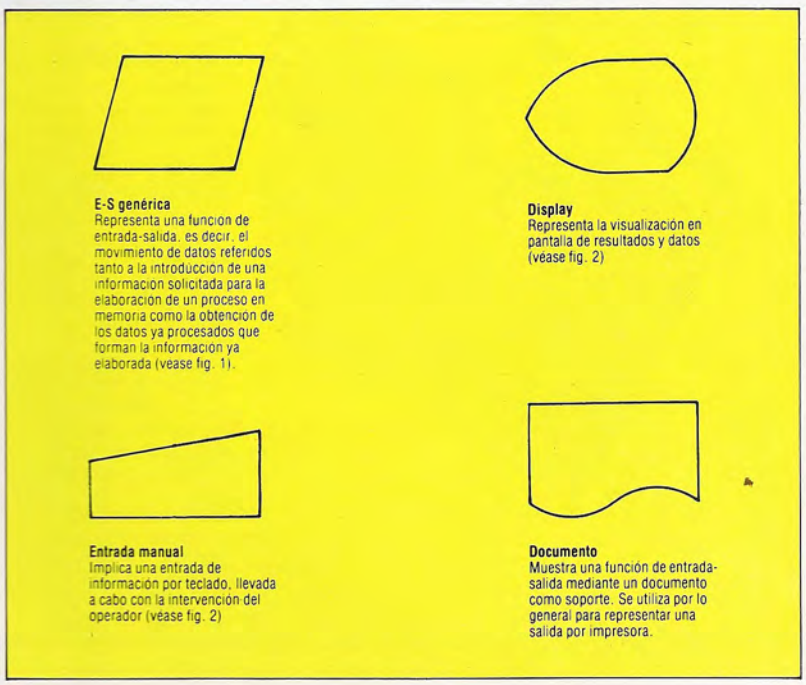
Símbolos de entrada y salida

Entenderemos su uso mediante un sencillo ejemplo contable

Supongamos el siguiente ejemplo: "Los clientes de una firma ven modificadas sus fichas (donde se guardan datos referidos tanto a razones sociales como a movimientos de géneros, pagos, etc.), conforme a los respectivos comprobantes, que obran en poder de las empresas y son diariamente timbrados. En ellos aparece la cantidad de género en movimiento que, si se refiere a una remesa, se deberá sumar a la existencia de la ficha correspondiente, mientras que si se trata de una devolución, la existencia se restará. En cualquier caso se mostrará la ficha como final de proceso, debidamente actualizada".

Una vez analizado el problema, comprobamos que un tipo de operación aparece repetido, ya que podemos distinguir dos entradas diferentes: la de la ficha correspondiente al cliente y la del comprobante. Dado que no se informa expresamente de la naturaleza de dichas entradas, éstas se considerarán genéricas.

Así, cuando leemos *tomar comprobante*, se entenderá por ello dar la cifra que en él aparezca, o sea, la cantidad correspondiente a la existencia en el caso de la ficha del cliente. Por otra parte, en la fase de *salida*, se pide que aparezca la nueva configuración de la ficha debidamente actualizada tras el proceso que tiene lugar una vez se ha decidido si se trata de una remesa o no.



E-S genérica
Representa una función de entrada-salida, es decir, el movimiento de datos referidos tanto a la introducción de una información solicitada para la elaboración de un proceso en memoria como la obtención de los datos ya procesados que forman la información ya elaborada (véase fig. 1).

Display
Representa la visualización en pantalla de resultados y datos (véase fig. 2)

Entrada manual
Implica una entrada de información por teclado, llevada a cabo con la intervención del operador (véase fig. 2)

Documento
Muestra una función de entrada-salida mediante un documento como soporte. Se utiliza por lo general para representar una salida por impresora.

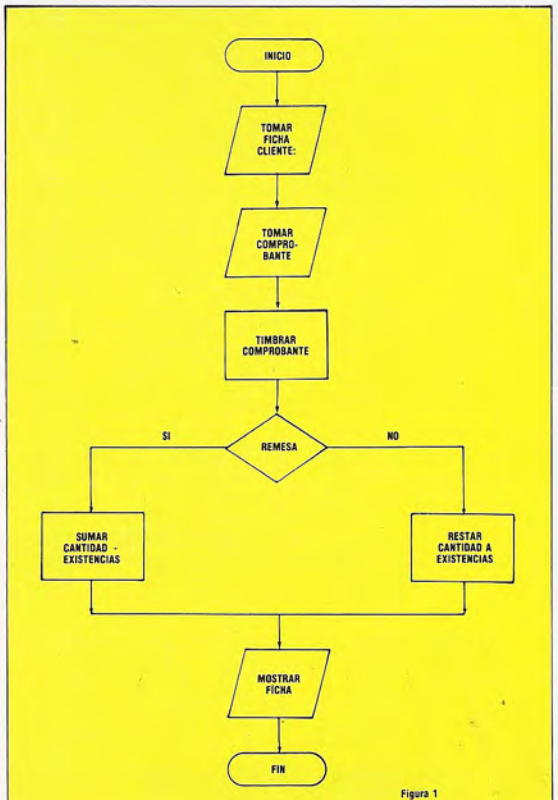


Figura 1

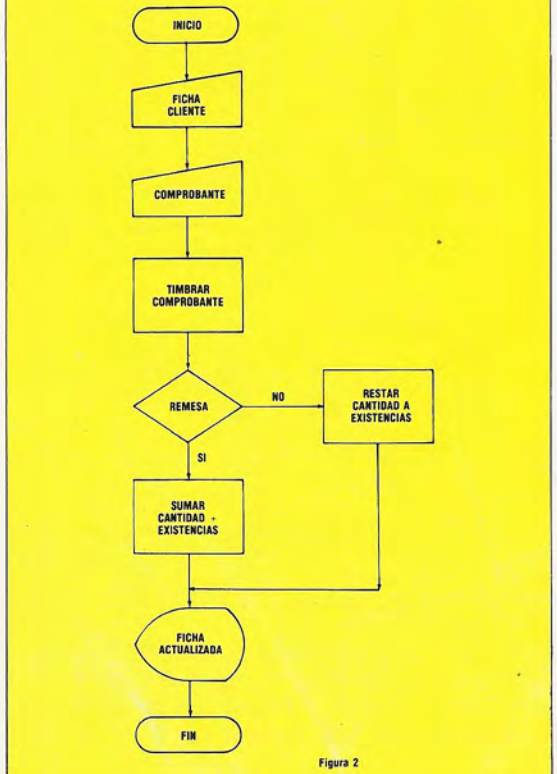


Figura 2

Informática de calidad

Si se afirma que la revolución del microordenador empezó el día en que la IBM entró en liza, recordemos que su entrada, en 1981, es muy reciente. Pero revolucionó el mercado

La reputación de IBM en los mercados del miniordenador y del ordenador de unidad principal no es, por cierto, la de una empresa innovadora en cuanto a hardware. Su software y su documentación, aunque con frecuencia profusamente detallados y exactos, se podrían mejorar en términos de presentación y sencillez de uso. Sin embargo, la empresa es celebrada por la solidez de sus equipos y el IBM PC es, ciertamente, robusto. Al igual que casi todo el hardware IBM, cuesta también considerablemente más que sus competidores.

Lo cual no parece ser un factor negativo a la hora de las ventas, pues, a pesar de unos precios que hacen parecer baratos a muchos ordenadores, la máquina se convirtió enseguida en una de las más populares. Se le ha hecho el definitivo cumplido de ser copiada e imitada al menos tanto como el Apple y, por cierto, más rápidamente.

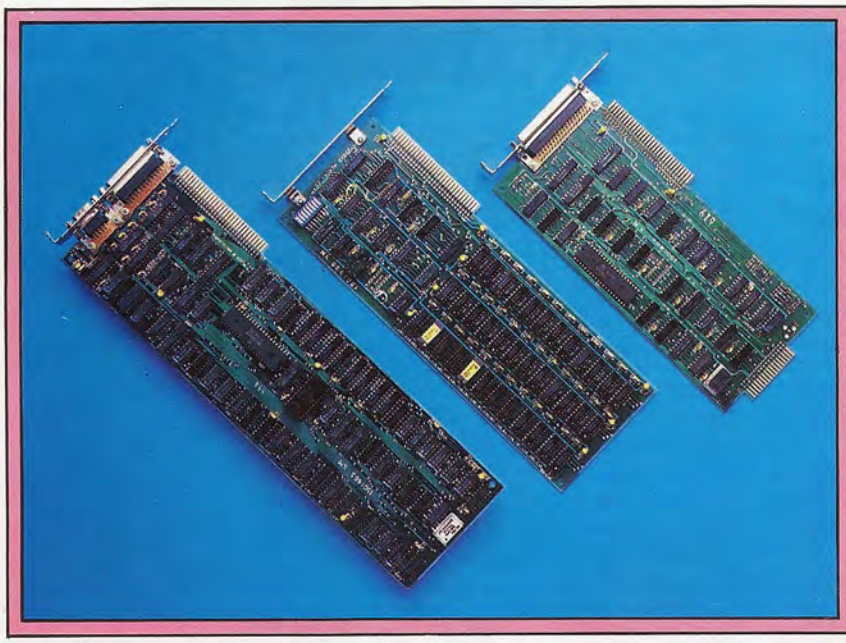
IBM explica que el elevado precio del PC es un reflejo del nivel de asistencia que ofrece la empresa. Dicha asistencia existe en realidad, si usted está dispuesto a pagar el 11,2 % del costo del producto por año para mantener un contrato de servicio. El costo de la mayoría de los contratos de servicio independientes es superior al menos en un 2 % y pocas empresas pueden ofrecer unidades de recambio en cuestión de momentos, de modo que tal vez exista algún mérito adicional en adquirir máquinas construidas por una empresa tan grande como IBM.

Las especificaciones del IBM PC no son descollantes. Posee un procesador 8088, descrito como una CPU de 16 bits, pero tiene las líneas de dirección y de datos multiplexadas para economizar patillas del chip y esto significa que no es rápida. De hecho, su rendimiento suele ser sólo alrededor del 25 % más rápido que una mediana máquina de ocho bits.

Tal como se suministra, el modelo básico necesita una ampliación para aprovechar al máximo su potencial, porque no posee mucha memoria (no la suficiente para ejecutar programas complicados) y casi no posee facilidades de entrada y salida. En

Chris Stevens

Diseño ergonómico
El diseño físico y el trazado del IBM PC reflejan la dilatada experiencia de la empresa en el campo de ordenadores y productos para oficina: son discretos y ergonómicamente sólidos. Las tres unidades principales (teclado, procesador y monitor) están separadas para facilitar su acomodamiento. La máquina viene con una unidad monocromática como estándar, pero existe a la venta un monitor en color



Tableros de ampliación
En su forma más elemental, el IBM PC apenas si constituye un desafío para aquellas máquinas que, como el Apple II, llevan mucho más tiempo establecidas en el mercado; pero cuando se amplía (de izq. a der.) ya sea con un tablero para gráficos en color, o con un tablero para ampliación de memoria o bien un sofisticado controlador de entrada-salida, la máquina comienza a parecerse mucho más a un serio ordenador personal para gestión empresarial

Unidades de disco
La máquina está equipada con una sola unidad de disco, de una sola cara y de densidad simple, pero ésta se puede mejorar progresivamente hasta una doble cara de densidad doble

Tableros controladores de disco

Fuente de alimentación eléctrica

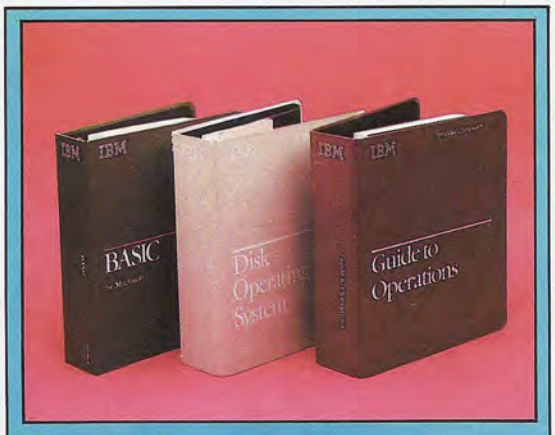
consecuencia, la mayoría de los compradores se encuentran con que deben agregar al menos una ficha multifunción, que cuesta tanto como muchos ordenadores personales.

Los gráficos son impresionantes pero no se suministran con la máquina; el usuario necesita una ficha para gráficos. Ésta se vende en dos versiones (monocromática y en color) y se compone de un gran bloque de memoria (para retener la imagen de pantalla) y de componenentes electrónicos que producen la señal de video. Las fichas las controla la CPU principal, si bien está surgiendo un nuevo tipo de ficha de visualización que posee un procesador de visualización en video especializado. Éste libera al procesador principal de la tarea de actualizar las pantallas y, por consiguiente, acelera el proceso. Otra vez es de lamentar que sea tan caro.

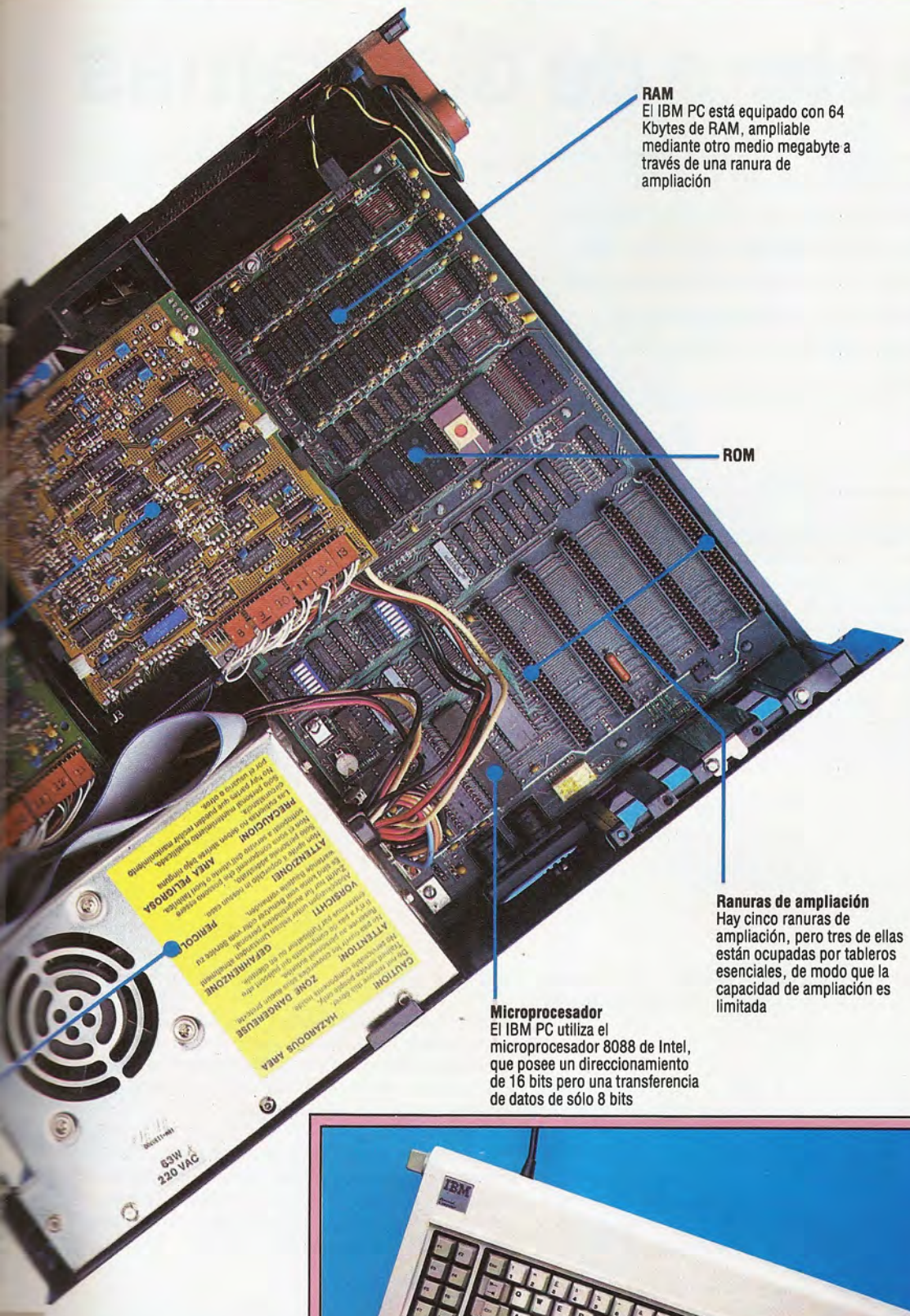
El PC posee ranuras de ampliación, pero sólo hay cinco y, por tanto, se ha de pensar cuidadosamente en cómo se han de utilizar. El resultado es que virtualmente no existen a la venta fichas "baratas y geniales" que ofrezcan funciones limitadas a un precio reducido; casi todas ellas son grandes, complejas y capaces de realizar muchas tareas diferentes, a menudo de forma simultánea, y son caras. Dado que el IBM PC no sirve para mucho sin dichas mejoras, su costo se debería tener en cuenta a la hora de considerar la posibilidad de adquirirlo.

Existen, por supuesto, modelos alternativos del PC (como la versión XT de disco rígido) que proporcionan muchas de estas facilidades como estándar, pero los precios son mucho más elevados. En realidad se pueden comparar con el del Lisa de Apple, una máquina mucho más avanzada.

Como muestra de que ni siquiera IBM es inmune a las tendencias de la moda, el PC ha sido remodelado como un "portátil", pero con un peso mínimo de 14 kg este término resulta bastante forzado. La remodelación implicó reemplazar las dos unidades de disco de altura estándar por la unidad de doble cara, que deja entonces una de las aperturas disponible para un monitor de 9 pulgadas. Una versión del teclado más ligera y más pequeña se ajusta frontalmente a la máquina y se aloja en una nueva carcasa.



Software IBM PC
Una de las principales razones para comprar un IBM PC (que es la misma por la cual tantos fabricantes de ordenadores de todo el mundo lo han copiado casi exactamente) es la selección de software comercial disponible. Éste funciona bajo el control del PC-DOS (sistema operativo en disco), desarrollado por Microsoft sobre la base del CP/M, aunque existe un número de alternativas (el CP/M-86 y UCSD p-system, p. ej.) y entre ellos estos sistemas operativos admiten una amplia variedad de lenguajes, como COBOL, FORTRAN, PASCAL y BASIC. La gama de software comercial es tan amplia como la de cualquiera de los sistemas de microordenadores existentes, incluyendo óptimos paquetes de procesadores de textos, hojas electrónicas, bases de datos y de gestión. Además, dado que en Estados Unidos el ordenador personal IBM PC se tiene en mucha más estima que en el resto del mundo, se dispone de un gran número de juegos producidos por empresas de software norteamericanas



IBM PC

DIMENSIONES
140 x 500 x 400 mm

CPU
Intel 8088

CAPACIDAD DE MEMORIA Y VELOCIDAD
64 K de RAM, ampliables a 576
40 K de ROM 4,7 MHz

CARACTERISTICAS DE LA PANTALLA
25 líneas de 80 caracteres

INTERFACES Y PUERTAS
Centronics en paralelo y cinco ranuras

LENGUAJES DISPONIBLES
BASIC, más una selección disponible bajo PC-DOS, que incluye COBOL, FORTRAN, etc.

TECLADO
79 teclas tipo máquina de escribir

DOCUMENTACION
Responde al estándar normal que se puede esperar de IBM y de los proveedores de software que ha escogido

VENTAJAS
El IBM PC básico se puede utilizar de forma bastante exitosa y después ampliarlo y mejorarlo convirtiéndolo en uno de los microordenadores más potentes de cuantos existen

DESVENTAJAS
Es muy caro si se compara con las muchas imitaciones del PC que aparecieron tras su lanzamiento. El servicio y el software son asimismo más caros que los de máquinas más sencillas

Teclado
Como se podría esperar de uno de los más grandes fabricantes de máquinas de escribir, terminales de ordenador y otros dispositivos activados por teclado, el teclado del IBM PC es virtualmente impecable. Está separado de la unidad procesadora, para que el usuario pueda colocarlo en la posición que desee, su perfil es sumamente plano y de inclinación ajustable, y utiliza teclas esculpidas dispuestas en cinco filas

Chris Stevens

El Peanut
El PC Junior de IBM, al que durante la fase de desarrollo se aludía en clave como el *Peanut* (cacahuete), es una versión notablemente rebajada de la máquina más grande. Quizá la innovación más interesante fuera el que se utilizara un enlace infrarrojo entre teclado y procesador, en lugar del cable habitual. El PC Jr está equipado con dos ranuras para cartucho

Lectura de diagramas

Una valiosa ayuda para la simplificación de los circuitos lógicos son los diagramas de Karnaugh. Las más complejas expresiones algebraicas se vuelven diáfanas con este método

Los *diagramas de Karnaugh* (también llamados *diagramas k*) son en realidad ampliaciones de los diagramas de Venn que ya conocemos (véase p. 526). Nos permiten representar las expresiones lógicas de forma gráfica. Un diagrama k asume formas ligeramente distintas según el número de letras (o variables) diferentes que haya en la expresión a simplificar, y es de mayor utilidad para las expresiones que contengan dos, tres o cuatro variables.

Dos variables: Cada uno de los cuadros de un diagrama k de dos variables (habrá $2^2 = 4$ cuadrados) representa una función AND, tal como refleja este diagrama:

	A	$\bar{A}$
B	A.B	$\bar{A}$.B
$\bar{B}$	A.$\bar{B}$	$\bar{A}$.$\bar{B}$

Para representar la expresión $AB + A\bar{B}$ como un diagrama k, colocamos unos en los cuadrados pertinentes:

	$\bar{A}$	A
B	1	0
$\bar{B}$	1	0

Aquí tenemos otros tres ejemplos, que representan respectivamente las expresiones $\bar{A}B$, $AB + \bar{A}\bar{B}$ y $AB + \bar{A}B + A\bar{B}$:

	A	$\bar{A}$
B	0	0
$\bar{B}$	0	1

	A	$\bar{A}$
B	1	0
$\bar{B}$	0	1

	A	$\bar{A}$
B	1	1
$\bar{B}$	1	0

Tres variables: En este caso, el número de cuadrados se multiplicaba por dos ($2^3 = 8$ cuadrados). El diagrama k básico de tres variables es:

		A	$\bar{A}$	
B		A.B.C	$\bar{A}$.B.C	C
B	$\bar{B}$	A.$\bar{B}$.C	$\bar{A}$.$\bar{B}$.C	
B	$\bar{B}$	A.$\bar{B}$.$\bar{C}$	$\bar{A}$.$\bar{B}$.$\bar{C}$	$\bar{C}$
B		A.B.$\bar{C}$	$\bar{A}$.B.$\bar{C}$	

He aquí dos expresiones, $AC + \bar{A}B\bar{C}$ y $AB + \bar{A}C$, representadas como diagramas k:

		A	$\bar{A}$	
B		1	0	C
$\bar{B}$		1	0	
$\bar{B}$		0	0	$\bar{C}$
B		0	1	

		A	$\bar{A}$	
B		1	1	C
$\bar{B}$		0	1	
$\bar{B}$		0	0	$\bar{C}$
B		1	0	

$$
\begin{aligned}
&ABC + A\bar{B}C + \bar{A}B\bar{C} \\
&= AC(B + \bar{B}) + \bar{A}B\bar{C} \\
&= AC + \bar{A}B\bar{C}
\end{aligned}
$$

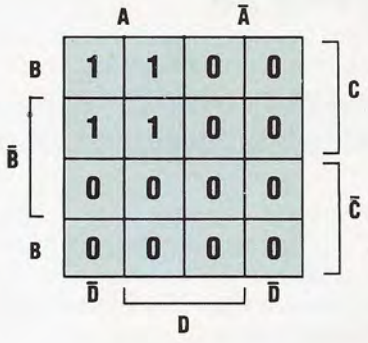
$$
\begin{aligned}
&ABC + AB\bar{C} + \bar{A}BC + \bar{A}\bar{B}C \\
&= AB(C + \bar{C}) + \bar{A}C(B + \bar{B}) \\
&= AB + \bar{A}C
\end{aligned}
$$

Observe que ambas expresiones se han simplificado utilizando la ley booleana según la cual un conjunto A relacionado mediante OR con su negativo ($\bar{A}$) da como resultado 1 (el conjunto universal o identidad).

Cuatro variables: Cuando comenzamos a tratar con cuatro variables, los mapas empiezan a hacerse más complicados (tienen $2^4 = 16$ cuadrados), pero no obstante son bastante fáciles de interpretar de acuerdo a la cuadrícula básica:

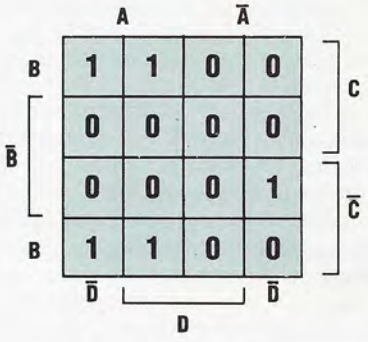
		A		$\bar{A}$		
B		A.B.C.$\bar{D}$	$\bar{A}$.B.C.D	A.B.C.D	$\bar{A}$.B.C.$\bar{D}$	C
B	$\bar{B}$	A.$\bar{B}$.C.$\bar{D}$	A.$\bar{B}$.C.D	$\bar{A}$.$\bar{B}$.C.D	$\bar{A}$.$\bar{B}$.C.$\bar{D}$	
B	$\bar{B}$	A.$\bar{B}$.$\bar{C}$.$\bar{D}$	A.$\bar{B}$.$\bar{C}$.D	$\bar{A}$.$\bar{B}$.$\bar{C}$.D	$\bar{A}$.$\bar{B}$.$\bar{C}$.$\bar{D}$	$\bar{C}$
B		A.B.$\bar{C}$.$\bar{D}$	A.B.$\bar{C}$.D	$\bar{A}$.B.$\bar{C}$.D	$\bar{A}$.B.$\bar{C}$.$\bar{D}$	
		$\bar{D}$	D	D	$\bar{D}$	

He aquí un diagrama k acompañado de una simplificación:



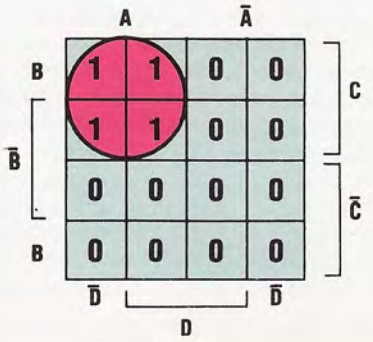
$$\begin{aligned} & ABC\overline{D} + ABCD + A\overline{B}C\overline{D} + A\overline{B}CD \\ &= ABC(\overline{D} + D) + A\overline{B}C(\overline{D} + D) \\ &= ABC + A\overline{B}C \\ &= AC(B + \overline{B}) \\ &= AC \end{aligned}$$

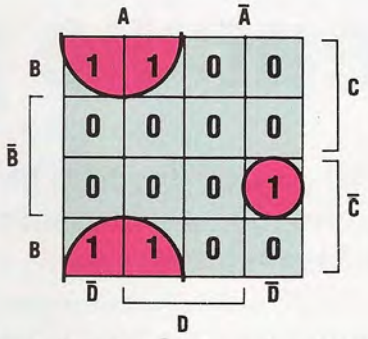
He aquí otro ejemplo:



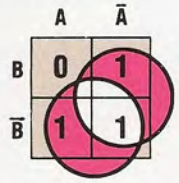
$$\begin{aligned} & ABC\overline{D} + ABCD + \overline{A}\overline{B}C\overline{D} + AB\overline{C}\overline{D} + AB\overline{C}D \\ &= ABC(\overline{D} + D) + \overline{A}\overline{B}C\overline{D} + AB\overline{C}(\overline{D} + D) \\ &= ABC + \overline{A}\overline{B}C\overline{D} + AB\overline{C} \\ &= AB(C + \overline{C}) + \overline{A}\overline{B}C\overline{D} \\ &= AB + \overline{A}\overline{B}C\overline{D} \end{aligned}$$

Si analizamos con más detenimiento la disposición de los unos en estos dos diagramas k, podremos descubrir cierta regularidad. En el primer ejemplo, todos los cuadrados con AC en sus expresiones poseen un 1. En el segundo ejemplo, lo mismo sucede con todos los cuadrados AB. Esto sugiere que una forma más sencilla de simplificar expresiones booleanas consiste simplemente en inspeccionar un diagrama k. Consideremos los siguientes:





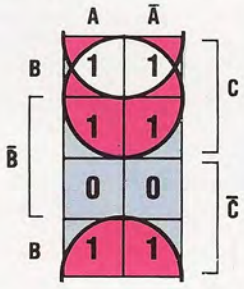
Con un poco de práctica, se pueden detectar grupos de dos, cuatro u ocho unos para formar términos más simples. Por ejemplo, consideremos esta expresión: $\overline{A}B + A\overline{B} + \overline{A}\overline{B}$.



Utilizando un diagrama k de dos variables, podemos detectar dos grupos de unos. Un grupo representa todos los casos NO(B) y el otro representa todos los casos NO(A), de manera que podemos simplificar la expresión a $\overline{A} + \overline{B}$. Esta expresión se puede volver a simplificar, mediante la ley de Morgan: $\overline{A.B}$. ¿Se puede llegar a la conclusión de forma más directa inspeccionando el diagrama k?

Un ejemplo más difícil lo ofrece esta expresión de tres variables:

$A\overline{B}\overline{C} + A\overline{B}C + \overline{A}BC + AB\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C$



El grupo de cuatro unos en la parte superior del diagrama k representa todos los posibles casos en los que C es verdadera. Las filas superior e inferior del mapa representan todos los posibles casos en los que B es verdadera. Por consiguiente, la expresión simplificada es: B + C.

En el próximo capítulo del curso proseguiremos con nuestra investigación acerca de la utilización de los diagramas k para simplificar expresiones booleanas que comprenden cuatro variables. Luego le mostraremos cómo se utilizan estos diagramas en el proceso del diseño de circuitos. Y ello nos permitirá unificar todos los aspectos del curso que hemos analizado hasta el momento.

Ejercicio 4:
Dibuje diagramas k de tres variables para simplificar las siguientes expresiones booleanas:

a) $\overline{A}B.C + \overline{A}.\overline{B}.C + \overline{B}.\overline{C} + \overline{A}.B.\overline{C}$
b) $A.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + A.B.\overline{C}$

Sencillo y lógico

El BASIC de Commodore no es una versión muy avanzada, pero tiene una lógica y una simplicidad notables y su editor de pantalla es de los mejores

Todas las máquinas CBM admiten variables con nombres largos, pero el intérprete sólo explora los dos primeros caracteres del nombre, de modo que, por ejemplo, tanto FUSION como FUERZA son admisibles, pero equivalentes. Por consiguiente, la salida de este fragmento:

```
100 FUSION = 17: FUERZA = 2*FUSION
200 PRINT FUSION, FUERZA
```

es:

```
 34       34
```

Esto se aplica a todos los tipos de variables: de coma flotante (p. ej., NUMERO), de enteros (p. ej., NUMERO%), en serie (p. ej., NUMERO$) y de matriz (p. ej., NUMERO$(62,47)). Los tipos de variables son convencionales, pero en el Vic-20 las variables de enteros son inutilizables porque la máquina no admite aritmética de enteros; el tipo entero se conservó simplemente para mantener la compatibilidad con otras máquinas Commodore que admiten aritmética de enteros.

Una negativa consecuencia de estas reglas sobre los nombres de las variables es que un nombre que pareciera válido podría ser invalidado porque sus dos primeras letras conformaran una palabra reservada (START, p. ej., es equivalente a ST como nombre de variable, y ST es una palabra reservada).

Las variables de matriz pueden tener hasta 255 dimensiones y su extensión sólo está limitada por la cantidad de RAM disponible. El primer elemento de cualquier matriz es el elemento(0), de modo que DIM EX(6) crea una matriz de siete elementos: EX(0), EX(1), EX(2),..., EX(6). Aquí la sentencia DIM es innecesaria, porque si el intérprete se encuentra con una variable de matriz de dimensión simple para la cual no se haya ejecutado ninguna sentencia DIM, se sobreentiende, por omisión, una dimensión 10; si el subíndice de dicha matriz es mayor que 10, entonces se producirá un error BAD SUBSCRIPT (subíndice malo). Ésta es una facilidad arbitraria que no favorece una buena práctica de programación: el intérprete debe reacomodar la memoria cada vez que se encuentra con una sentencia DIM (o la primera referencia a una matriz no DIMensionada), de modo que todas las matrices se deben DIMensionar al mismo tiempo al comienzo del programa, antes de que se emplee ninguna variable simple. Si esto no se hace no sucederá nada grave, pero incidirá ligeramente en la velocidad de ejecución.

Debido a la forma en que trabaja la mayoría de los intérpretes de BASIC, la ejecución de los programas se puede acelerar inicializando las variables del programa utilizadas más comúnmente por su orden de importancia; esto se puede hacer con sentencias de asignación o con la DIM. Una línea como:

```
10 DIM A$(10,24),K,L,MARCADOR
```

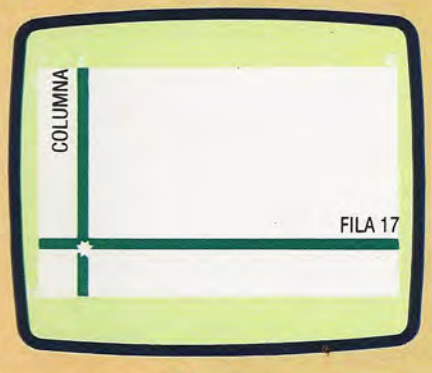
no tendrá ninguna consecuencia obvia, pero se trata de una forma rápida de colocar las variables K,L y MARCADOR arriba de la tabla de símbolos, haciéndolas, por lo tanto, más accesibles al intérprete y aumentando la velocidad de ejecución.

Una mirada a la lista de las palabras clave en un manual para el usuario CBM (del que hablaremos más adelante) revela unas pocas omisiones, y algunos añadidos, al juego Microsoft completo. Tal vez la omisión más importante sea INKEY$ y las adiciones más significativas TIME$ y STATUS.

INKEY$, la función para exploración del teclado, se sustituye por la sentencia GET. Al igual que INKEY$, hace que se explore el primer carácter del buffer del teclado y devuelve su valor ASCII. GET se suele usar más a menudo en sentencias como:

```
150 GET GT$:IF GT$ = "" THEN 150
```

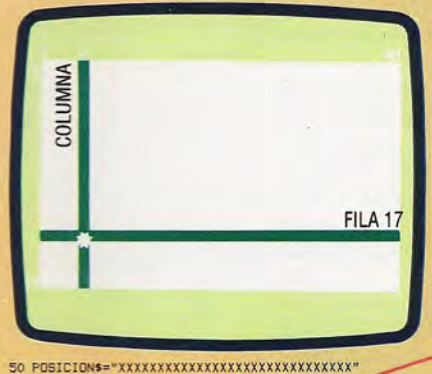
Posicionamiento del cursor



La capacidad para incluir órdenes para el cursor en una cantidad en serie puede hacer fácil el diseño de gráficos con el Commodore, especialmente cuando se utilizan de forma conjunta con la poderosa orden para edición en pantalla

```
100 PRINT AT(17,4)"*"
```

Si el BASIC de Commodore admitiera la orden PRINT AT, sería sencillo posicionar el cursor



Pero como no lo admite, debemos utilizar la configuración de cursor programable:

Inicialice POSICION$ después coloque los parámetros de fila y columna como en esta expresión

```
50 POSICION$="XXXXXXXXXXXXXXXXXXXXXXXXXX"
100 PRINT LEFT$(POSICION$,17)TAB(4-1)"*"
```

Si tuviera que hacer mucho trabajo de formato en pantalla, valdría la pena colocar el orden para posicionamiento del cursor dentro de una subrutina y luego inicializar las variables FILA y COLUMNA con la posición de pantalla requerida, antes de llamar a la subrutina

```
50 POSICION$="XXXXXXXXXXXXXXXXXXXXXXXXXX"
100 FILA=17:COLUMNA=4:GOSUB 1000:PRINT"*"
500 END
1000 PRINT LEFT$(POSICION$,FILA)TAB(COLUMNA-1);:RETURN
```

Liz Dixon

que tendrá el efecto de detener la ejecución del programa hasta que se pulse una tecla, en cuyo caso GT$ contendrá el carácter correspondiente a la tecla pulsada. Puede que no siempre sea éste el caso, porque GET explora el buffer del teclado y no el teclado propiamente dicho, de modo que si cuando se efectúa GET el buffer contiene algunos caracteres, entonces la ejecución del programa no esperará a que el usuario pulse una tecla. Ello se puede demostrar mediante este programa:

```
50 FOR K = 1 TO 100:PRINT K:NEXT K
60 PRINT "PULSE CUALQUIER TECLA"
150 GET GT$:IF GT$ = "" THEN 150
200 PRINT "USTED HA PULSADO LA TECLA";GT$
```

Si ejecuta este programa verá que en la pantalla aparecen los números del 1 al 100, seguidos del mensaje para la entrada, y después nada hasta que usted pulsa una tecla. Sin embargo, si pulsa una tecla mientras se están imprimiendo los números, entonces esa pulsación penetrará en el buffer y allí la encontrará la sentencia GET, de modo que no se producirá ninguna pausa en la ejecución del programa. Esto puede ser fastidioso y posiblemente desastroso, por ejemplo, en los juegos, donde se producen muchas pulsaciones frenéticas de teclas. La solución consiste en restaurar el buffer justo antes de que se ejecute la sentencia GET, mediante la inserción en el programa de:

```
149 POKE KBPTR,0
```

donde KBPTR es la dirección del contador de cola de teclado (198 en el Commodore 64).

Normalmente GET no genera un cursor centelleante en la pantalla, pero POKE FLASH,0 —donde FLASH es la dirección de la bandera de activación del centelleo de cursor (204 en el Commodore 64)— proporcionará uno.

TIME$ (que se suele abreviar TI$) es el reloj del sistema; en el momento del encendido se inicializa a "000000" y a partir de entonces marca el tiempo en horas, minutos y segundos hasta "235959" (23 horas 59 minutos 59 segundos desde la inicialización), momento en que se restaura a "000000". Está a disposición del usuario al igual que cualquier otra variable, y se puede establecer en cualquier hora legal, tal como: TI$ = "000000" o como TI$ = "084503".

Relacionado con TI$ está TI, su equivalente numérico. TI marca la hora desde la inicialización expresada en sesentavos de segundo (denominados *jiffys*), de modo que su valor oscila entre 0 y 5183999 (60*60*60*24-1). TI depende de TI$ y no se puede inicializar por sí mismo; sólo se inicializa TI$.

STATUS (que se abrevia ST) es una variable definida por el sistema. Cuando se detecta un error en un dispositivo de entrada-salida, el valor de ST será un número que indique el tipo de error detectado. Éste se escribiría así:

```
330 IF ST > 0 THEN GOSUB 30000:REM ERROR EN
    MANIPULACION DE ARCHIVO E/S


30000 REM MENSAJES DE ERROR
30100 IF ST = 16 PRINT "ERROR IRRECUPERABLE
    DE LECTURA"
```

CMD es un miembro muy útil del juego de instrucciones Commodore. Tiene el efecto de desviar la salida de la pantalla a un canal de salida seleccionado. Posee muchas aplicaciones útiles, como por ejemplo:

```
OPEN 4,4:CMD 4:LIST
```

Esto LISTa el programa corriente a la impresora en vez de a la pantalla; cuando el listado esté completo se ha de ejecutar:

```
PRINT#4:CLOSE 4
```

CMD se puede utilizar en un programa para copiar la pantalla en la impresora. Supongamos que en su programa GOSUB 3000 hace que se imprima (PRINT) en la pantalla (en vez de que se coloque —POKE— en la memoria de pantalla) un mensaje o algunos datos. Para poder copiar esa visualización en la impresora, digite:

```
OPEN4,4:CMD 4:GOSUB3000:PRINT#4:CLOSE4
```

Aunque se acepta un signo de interrogación (?) como abreviatura de la palabra clave PRINT, no se puede abreviar PRINT# por ?#; para hacer esto debe utilizar pR (p seguida de r con tecla de cambio).

Las abreviaturas de las palabras clave Commodore son tan antiguas como las propias máquinas Commodore, pero oyendo a los usuarios de ordenadores Spectrum se podría pensar que la idea la concibió Sinclair. Casi todas las palabras clave se pueden abreviar con su letra inicial seguida de la segunda letra mayúscula. Cuando dos palabras clave o más poseen las mismas dos primeras letras, entonces la abreviatura consistirá en las dos primeras letras más la tercera cambiada: READ, RESTORE y RETURN, por ejemplo, se abrevian rE, reS y reT.

El editor de pantalla, y el sólido a la vez que cómodo sistema operativo sobre el que se sustenta, constituye la configuración individual más significativa de las máquinas Commodore. Se viene utilizando desde el lanzamiento del PET y sigue siendo uno de los mejores editores de pantalla para micros que existen. Para editar una línea de programa, por ejemplo, LISTe la línea, desplazar el cursor directamente hasta cualquier punto de la línea, editar el texto y pulsar RETURN. No importa dónde esté el texto en la pantalla ni dónde esté el cursor en la línea: cuando pulsa RETURN el texto de la línea en pantalla que contiene el cursor entra en el sistema como si usted lo hubiera digitado.

Una sutileza de este sistema de edición es la facilidad que otorga para copiar. Supongamos que tiene que dar entrada a estas dos líneas:

```
100 IF  INT(NUMERO/INDICE—TASA) = 5  THEN
    3000
200 IF INT(NUMERO/INDICE—TASA) = 7 THEN
    3800
```

Sólo necesita digitar la primera línea y pulsar RETURN; luego, con ese texto en la pantalla, desplaza el cursor hacia arriba, cambia el número de línea 100 por 200, cambia 5 por 7, cambia 3000 por 3800 y vuelve a pulsar RETURN. En la memoria la línea 100 permanece tal como estaba y su texto editado en la pantalla se convierte en la línea 200. Este proceso se puede repetir tantas veces como lo desee. Ésta no es más que una de las fascinantes facilidades que uno puede obtener con el editor, pero demuestra lo fácil y directa que resulta su utilización.

Superestafa

En 1971 Jerry Schneider estafó 1 000 000 de dólares en equipos a la empresa Los Angeles Pacific Telephone and Telegraph. Recuperando manuales y equipos de los depósitos de desperdicios de esa compañía, se hizo pasar por periodista y consiguió los códigos de acceso para el ordenador IBM 360 de la empresa. Schneider hacía pedidos discretos y revendía la mercancía. Tras pasar 40 días en prisión, se colocó como consultor en seguridad de ordenadores.

Líneas de Assembler

Vamos a revisar los principales recursos utilizados para tratar con la memoria y veremos, además, las diferencias entre los programas para el 6502 y el Z80

Cuando se ejecuta (RUN) un programa, lo primero que hace el sistema operativo es inspeccionar los indicadores de comienzo de texto en BASIC con el fin de determinar en qué lugar de la memoria reside el programa a ejecutar. Para hacer esto, sin embargo, el sistema operativo tiene que almacenar las direcciones de los indicadores. Entonces ¿por qué el OS no almacena sencillamente las direcciones indicadas?

La razón principal es la flexibilidad. El sistema operativo, como recordará (véase p. 84), es un programa permanente residente en la ROM, y cualquier dato que contenga (como las direcciones de memoria) es igualmente permanente. Supongamos que durante un tiempo determinado se fueron lanzando distintas versiones de un ordenador y que, mientras en la versión 1 fue conveniente tener el comienzo del texto en BASIC en el byte 2048, en la versión 2 se hizo necesario cambiarlo de lugar y colocarlo en el byte 4096. Ello significará que la última máquina no será capaz de utilizar el sistema operativo de la versión anterior, debido a la diferente posición del área para textos en BASIC. Además, habría que crear nuevas ROM para cada nueva versión de la máquina, lo que resulta costoso; y bien podría ser que el software escrito para una versión no se pudiera emplear con la otra. Por el contrario, si las ROM del sistema operativo sólo contienen las direcciones de los indicadores, entonces se pueden utilizar estas mismas para todas las versiones de la máquina y de un modelo a otro sólo se necesitará cambiar el contenido de los mismos. La posición de los indicadores propiamente dichos puede permanecer constante, porque el sistema operativo requiere un bloque de memoria relativamente pequeño para espacio de trabajo y almacenamiento de datos (por lo común, alrededor de 1 000 bytes). Fijar la posición de este bloque (habitualmente las primeras cuatro páginas de la memoria) y diseñar o rediseñar el sistema alrededor del mismo no supone un gran impedimento para el equipo de diseño. Sin embargo, tener fija, pongamos por caso, la posición del área para textos en BASIC (un bloque de entre 3 000 y 40 000 bytes) sí supone una gran limitación.

La práctica estándar

Es práctica generalizada almacenar las direcciones de los indicadores en la forma que se conoce como *lo-hi* (apócope de *low-high*: "bajo-alto", en inglés). Si el byte 43 y el byte 44, por ejemplo, han de señalar la dirección 7671 (página 29, desplazamiento 247), entonces el byte 43 contendrá 247 (el desplazamiento o byte *lo*—bajo— de la dirección), mientras que el byte 44 contendrá 29 (la página o byte *hi*—alto— de la dirección). Esto puede confundir al principio, pero resulta conveniente para el microprocesador. También es lógico que el byte *lo* de la dirección se almacene en el byte *lo* del indicador, y el byte *hi* de la dirección en el *hi* del indicador.

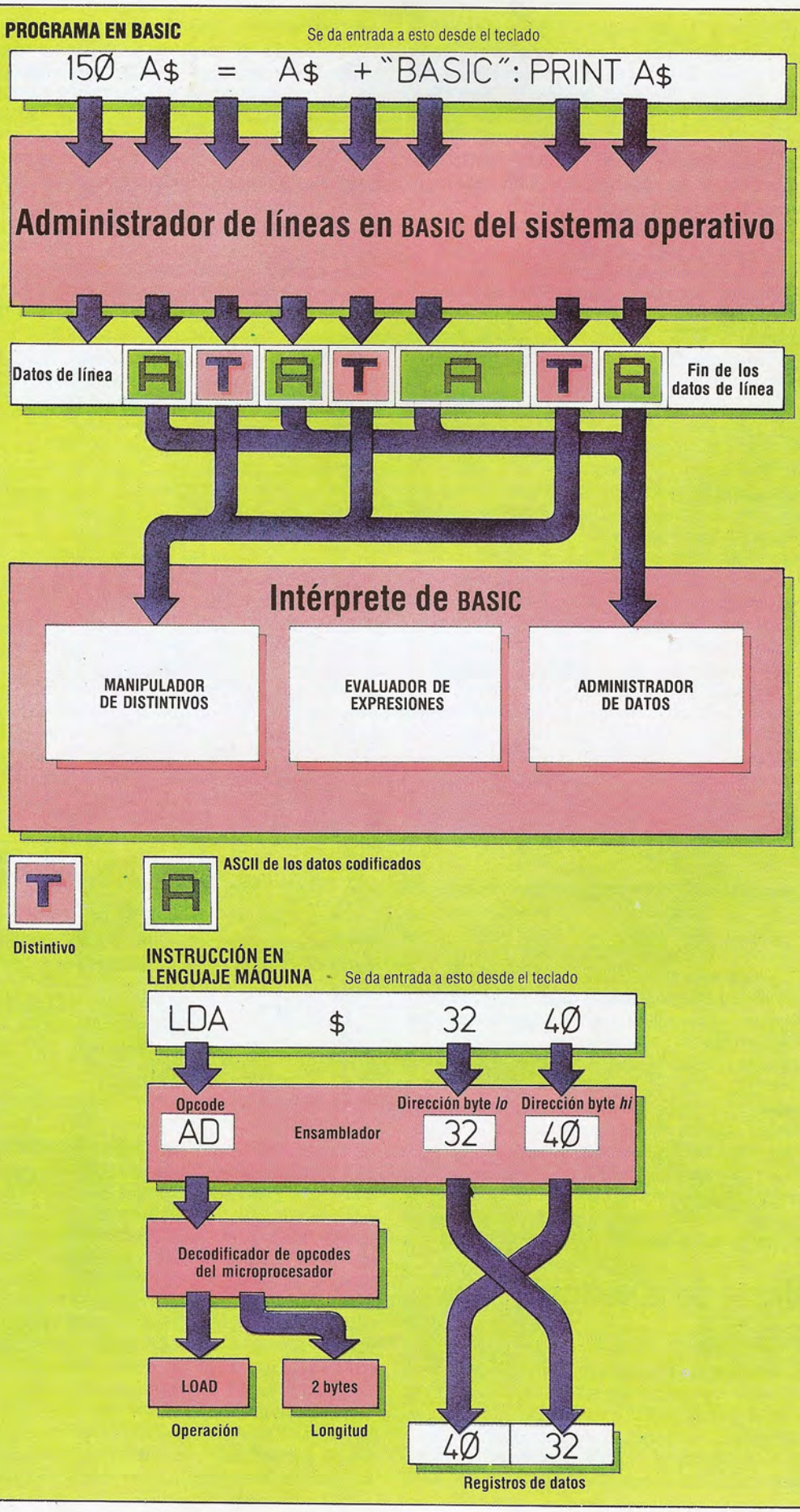
Si repetimos el ejemplo anterior utilizando números hexas en vez de números decimales, podremos apreciar la gran ventaja del sistema hexadecimal (de ahora en adelante, las direcciones y otros números siempre se escribirán en hexa precedidos por el signo $). Los bytes señaladores son $2B y $2C, y la dirección a la que señalan es $1DF7. Por consiguiente, $2B contiene F7 (el byte *lo* de la dirección), mientras que $2C contiene $1D (el byte *hi* de la dirección). Observe que cuando la dirección está en hexa los dos dígitos hexas a la derecha son el byte *lo*, y los dos dígitos a la izquierda son el byte *hi*, lo que tiene mucho más sentido que utilizar números decimales.

Es importante destacar que el BBC y el Spectrum se desvían de esta norma, ya que almacenan los números de línea del programa como números de dos bytes en forma *hi-lo* (alto-bajo) en lugar de *lo-hi* (bajo-alto). Cierto que son parámetros del programa en vez de direcciones de bytes, pero así y todo están al reves de la convención habitual.

Otra práctica corriente en el direccionamiento de memoria es la de denominar los indicadores mediante la dirección del byte *lo* solamente a pesar de que son cantidades de dos bytes. Podríamos decir, por ejemplo, que en el Commodore 64 el byte 43 señala el comienzo del texto en BASIC. En este caso se sobreentiende, sin embargo, que el byte 43 y el byte 44 son conjuntamente los indicadores.

Otro aspecto a considerar son los distintivos, o *tokens* (véase p. 556). Tienen una significación doble para los programadores en lenguaje máquina: representan órdenes en inglés de múltiples caracteres (como PRINT o RESTORE) mediante códigos numéricos de un único byte; y utilizan además desplazamientos. En BASIC, una orden es una palabra, pero para el sistema operativo ejecutarla no es una sola operación. La orden PRINT, por ejemplo, exige que se hallen en la memoria o evalúen los datos a imprimir, para después enviarlos carácter por carácter en código ASCII. Estas diversas tareas se llevan a cabo mediante una subrutina del programa del intérprete de BASIC. Cuando el intérprete encuentra en una línea de programa el distintivo PRINT, toma el valor de ese distintivo para localizar y ejecutar la correspondiente subrutina.

Supongamos que en nuestra versión de BASIC sólo hay tres órdenes: INPUT, PRINT y STOP; y éstas tienen asignados los distintivos $80, $81 y $82, respectivamente. Además, vamos a suponer que las subrutinas del intérprete que ejecutan estas órde-



Paso a paso

Este recuadro muestra cómo se traduce y se ejecuta una línea de programación en BASIC y una instrucción en lenguaje máquina

El sistema operativo transmite los datos de la línea en la forma habitual, sustituyendo las sentencias del BASIC por los correspondientes distintivos

Aquí se digita RUN

El intérprete de BASIC busca en la línea los distintivos y los datos relacionados con los mismos, utilizando el valor del distintivo para localizar la subrutina de manipulación del sistema operativo apropiada

El ensamblador traduce las expresiones mnemotécnicas del lenguaje Assembler en opcodes de un byte y almacena el operando de 2 bytes en forma *lo-hi*

Cuando se ejecuta la instrucción, el microprocesador decodifica el opcode en códigos de longitud y de operación, para así tratar como operando al número correcto de bytes que siguen al opcode

Kevin Jones

nes empiezan en los bytes $D010, $EA97 y $EC00 respectivamente, y que estas tres direcciones están almacenadas en forma *lo-hi* en los seis bytes a partir de $FA00 y hasta $FA05, que nos proporcionan una tabla de tres indicadores de dos bytes. Pues bien, cuando nuestro intérprete imaginario encuentra un distintivo ($81, p. ej.) procede a restarle $80, multiplica el resultado por dos y lo suma a $FA00. El resultado final en este caso es $FA02, que es el byte *lo* del indicador para la subrutina PRINT. Si se hubiera encontrado con un distintivo diferente a $81, entonces el algoritmo descrito habría devuelto la dirección del indicador para la subrutina correspondiente. De esta manera, la orden PRINT se reemplaza por un distintivo, $81, que es un desplazamiento de una tabla de indicadores que dirige al intérprete hasta la parte en cuestión de su propio programa.

Tenemos aquí una medida de la "distancia" entre el BASIC, un lenguaje denominado de alto nivel, y el lenguaje máquina, o lenguaje de bajo nivel. A nosotros el BASIC nos parece comprensible porque utiliza palabras de un código tan asimilable como es la lengua inglesa, la lógica algebraica y los números y series. Cuando sustituimos las palabras por distintivos y el resto por códigos ASCII, se parece ya a algo que el microprocesador puede manipular.

Por último, estudiemos la noción de *contexto*. En el área para textos en BASIC hemos visto la extendida utilización de los códigos: códigos ASCII para representar caracteres y números, distintivos para representar órdenes y (en el Spectrum) códigos binarios especiales para representar datos numéricos. Todos estos códigos se reducen a números binarios en la escala entre 00000000 y 11111111 (de $00 a $FF, de 0 a 255 en decimal) contenidos en bytes individuales de memoria e interpretados de acuerdo a su contexto. Dentro del área para textos en el BASIC del Commodore 64, la línea de programa:

```
200 rem*******left$*********
```

podría tener tres bytes conteniendo el número decimal 200: uno para el byte *lo* de la dirección de enlace, otro para el byte *lo* del número de línea y el tercero en la representación del distintivo de "left$". Cada byte tiene el mismo aspecto que los otros, y sin embargo significa algo diferente. Sólo sus expectativas le dirán a usted cómo interpretar ese valor en distintas situaciones.

Y aquí es donde realmente abordamos el inicio de estas lecciones de lenguaje máquina. Dijimos entonces que todo lo que hay almacenado en un ordenador está en algún tipo de lenguaje máquina. Parte de ello era familiar (como los códigos ASCII), parte desconocido (como los distintivos) y el resto quedaba sin explicar (como los programas en lenguaje máquina). Toca, pues, explicar estos mismos programas en lenguaje máquina.

Códigos de operación (opcodes)

Los programas en lenguaje máquina se reducen a unas cuantas secuencias de bytes situados en algún lugar de la memoria, que representan a su vez una mezcla de instrucciones para el microprocesador y datos sobre los cuales ha de operar éste. De modo semejante a lo que sucede con los demás bytes de la memoria, sólo el contexto puede separar los bytes de datos de los bytes de instrucciones, por lo que primero debemos considerar el formato de las instrucciones de un programa en lenguaje máquina.

Una instrucción en lenguaje máquina empieza con un código que indica la operación a realizar. Se denomina *código de operación* (abreviatura inglesa: *opcode* u *opc*) y puede tener uno o dos bytes de longitud. El opcode puede ser una instrucción cuya ejecución no necesite datos, pero las más de las veces va seguido de uno o dos bytes de datos. Un byte individual de datos puede ser una constante numérica o un código ASCII, mientras que dos bytes de datos a continuación de un opcode siempre son una dirección (almacenada en la forma byte *lo*-byte *hi*). Con la descripción anterior surgen de inmediato las diferencias existentes entre los microprocesadores: el BBC Micro utiliza un MOS Tech 6502A, el Commodore 64, un MOS Tech 6510 (muy similar al 6502A, de modo que en el futuro hablaremos generalmente sólo del 6502), y el Spectrum posee el Zilog Z80A. MOS Tech y Zilog produjeron sus microprocesadores aproximadamente al mismo tiempo (a comienzos de los años setenta), después que Intel lanzara el primer microprocesador, en 1971. Por consiguiente, tanto el 6502 como el Z80 comparten un mismo criterio de diseño, pero difieren sustancialmente en los detalles. En particular, los códigos de lenguaje máquina del Z80 son completamente distintos de los códigos de lenguaje máquina del 6502. Así, por ejemplo, los opcodes del 6502 siempre son de un byte de largo y pueden ir seguidos por uno o dos bytes de datos o por ninguno; pero los del Z80 pueden ser de dos bytes de largo, seguidos también, a su vez, por uno o dos bytes de datos o por ninguno.

Al ser enviado un opcode al microprocesador, el programa interno de la CPU lo decodifica en códigos de operación y de longitud, y es esta última información la que le permite al microprocesador interpretar los bytes que siguen al opc. Por ejemplo, para el 6502 la secuencia de bytes hexas:

```
A9 0E 8D 01 4E 60 44 52 41 54
```

representa tres instrucciones, seguidas de cuatro bytes de códigos ASCII. Esto se podría reescribir:

```
A9 0E
8D 01 4E
60
44
52
41
54
```

que muestra cómo la primera instrucción es el opc A9, seguido siempre por un byte de datos; la siguiente instrucción es el opc 8D, también seguido por dos bytes de datos; mientras que la siguiente es el opc 60, que no necesita ningún dato pues sólo hace que la ejecución del programa se bifurque, de modo que los siguientes bytes de datos el procesador no los examina para nada. Si al microprocesador se le envía el primer byte, A9, cuando espera recibir un opc, entonces a partir de ese momento todo funciona bien. La información de cada opc asegurará que el procesador recoja el número correcto de bytes de datos para cada opc, y el siguiente byte se tratará como el siguiente opc. Sin embargo, si al procesador, esperando un opc, se le envía el segundo byte, 0E, tratará a éste como un opc, dando por resultado que la secuencia se interprete:

```
0E 8D 01
4E 60 44
52
```

que significa: opc 0E, que necesita dos bytes de datos; después opc 4E, que también pide dos bytes de datos, después opc 52, que no es un opc legal, ocasionando en el procesador el equivalente de un error de sintaxis. Esto demuestra cómo un error de interpretación inicial genera una serie de graves errores lógicos en la ejecución del programa.

Esto también demuestra claramente algunos otros puntos importantes acerca del lenguaje máquina: que no es muy amable con el usuario (al menos, al comienzo) en cuanto que resulta difícil de leer y de escribir; que es terriblemente secuencial, sin que haya nada, excepto el orden, que diferencie a una instrucción de otra; y que es literal sólo en la medida en que lo puede ser una máquina, obedeciendo instrucciones erróneas con la misma presteza que obedece las instrucciones correctas, y rechazando sólo errores de sintaxis.

Parte de esta hosquedad se puede evitar recurriendo a técnicas mnemónicas alfabéticas escritas en lugar de opcodes numéricos mientras se está componiendo el programa, y recurriendo sólo a los opcodes cuando el programa se está cargando realmente en la memoria. Estos recursos mnemotécnicos forman el *lenguaje ensamblador* (Assembler), y el proceso de traducción a opcodes numéricos se conoce como *ensamble* o *ensamblamiento*. Observe que existe una correspondencia biunívoca entre el conjunto de expresiones mnemotécnicas del lenguaje ensamblador y el de opcodes: aquél es de nivel más alto que el código de lenguaje máquina, la diferencia entre ambos es mínima.

Si reescribimos el fragmento de código de lenguaje máquina anterior en forma de lenguaje ensamblador 6502, éste adoptaría la forma siguiente:

```
0000 A9 0E      LDS #$0E
0002 8D 01 4E   STA $4E01
0005 60         RTS
```

mientras que la misma secuencia de operaciones en lenguaje ensamblador Z80 sería de esta manera:

```
0000 3E 0E      LD A,$0E
0002 32 01 4E   LD ($4E01),A
0005 C9         RET
```

La primera columna muestra las direcciones hexas en la memoria del primer byte de la línea; el opc A9 en el listado 6502, por ejemplo, está en el byte 0; el byte de página 4E en ambos listados está en el byte 4, y así sucesivamente. La siguiente columna puede contener uno, dos o tres bytes y muestra el listado en lenguaje máquina. La tercera columna empieza con una expresión mnemotécnica del lenguaje ensamblador y muestra la versión en este lenguaje del de máquina. No se moleste en este momento en tratar de descifrarlo todo: es suficiente por ahora haber visto un listado en lenguaje Assembler, y observar las diferencias entre las versiones Z80 y 6502. Mejor también que note ya que la dirección de la segunda línea aparece en forma *lo-hi* convencional en código de lenguaje máquina, pero en forma *hi-lo* "normal" en lenguaje ensamblador.

En el próximo capítulo del curso comenzaremos a examinar con detalle los opcodes y echaremos una mirada a la arquitectura del microprocesador.

Conversión a hexadecimal

Para convertir el programa Mempeek de la página 539 de modo que el contenido de los bytes se visualice en hexadecimal en lugar de en decimal, introduzca las siguientes modificaciones:

BBC Micro

Agregue:

```
3000 DEF PROCHXPRINT(NUMDEC)
3100 LOCAL X$
3200 X$ = "0123456789ABCDEF"
3300 HB = INT(NUMDEC/16):LB = NUMDEC-HB# 16
3400 B$ = MID$(X$,HB + 1,1) + MID$(X$,LB + 1,1) + " "
3500 PRINT B$;
3600 ENDPROC
```

y cambie la línea 600 por:

```
600 PROCHXPRINT(PK%)
```

Spectrum

Agregue:

```
10 LET X$ = "0123456789ABCDEF"
3000 REM*********S/R BYTE HEXA*********
3100 LET HB = INT (PK/16): LET LB = PK-HB*16
3200 LET B$ = X$(HB + 1) + X$(LB + 1) + " "
3300 PRINT B$;
3400 RETURN
```

Y cambie la línea 600 por:

```
600 GOSUB 3000
```

Commodore 64

Agregue:

```
10 LET X$ = "0123456789ABCDEF"
3000 REM*********S/R BYTE HEXA********
3100 HB = INT(PK/16):LB = PK-HB*16
3200 B$ = MID$(X$,HB + 1,1) + MID$(X$,LB + 1,1) + " "
3300 PRINT B$;
3400 RETURN
```

y cambie la línea 600 por:

```
600 GOSUB 3000
```

Estas modificaciones harán que el contenido de la memoria se visualice en hexadecimal. Aun así, a la dirección de comienzo y al número de los bytes se les debe dar entrada en decimal

Inversión para el futuro

Xerox, el mayor vendedor del mundo de aparatos de reproducción gráfica, se ha lanzado también a la conquista del campo de la automatización de oficinas

A principios de la década de los setenta, la conocida y prestigiosa empresa Xerox (tanto, que muchos dicen "xerocopias" en vez de "fotocopias") planificó un programa de investigación a gran escala para hacer realidad un sueño: el de tener disponible la información en la oficina al punto, como la electricidad o el agua corriente. Xerox creó un nuevo equipo de investigación con cheque en blanco y veló por su máxima libertad de funcionamiento estableciéndolo en Palo Alto (California), en la otra punta del país, alejado de las oficinas centrales de Xerox en Rochester (New Hampshire).

Chris Stevens

El precedente del Lisa
Uno de los mayores éxitos del PARC fue el desarrollo del STAR, un sistema de programación que utiliza el lenguaje SMALLTALK. El STAR opera combinando programas y datos en el mismo archivo para procesamiento. La tecnología del Lisa de Apple le debe mucho a esta innovación; de hecho, la mayoría de los miembros del equipo de desarrollo del Lisa se reclutaron en el PARC

El traslado al condado de Santa Clara (California) dio sus frutos. Situado cerca del campus de la Universidad de Stanford, que contaba con un floreciente departamento de ciencia informática especializado en el estudio de la inteligencia artificial, el Centro de Investigación de Palo Alto (PARC: *Palo Alto Research Center*) atrajo a varios de los mejores cerebros de la informática. En esta comunidad cerrada, los estudiantes de talento podían pasar fácilmente de la investigación académica a la investigación comercial. El PARC se convirtió en el centro de la cultura de ordenadores, produciendo una jerga que sólo resultaba comprensible a los iniciados. Varios productos de Xerox recibieron apodos durante la misma fase de creación. La serie de micros 820, por ejemplo, recibió el nombre en clave de "Worm" (gusano), dando por hecho que habría de "comerse el Apple" (manzana).

El ímpetu primordial del nuevo equipo de investigación estaba encaminado a desarrollar una red de área local (LAN: *Local Area Network*). Ahora este término ya se ha convertido en un lugar común, pero cuando Xerox construyó su primera red experimental, en Hawaii, a finales de los años sesenta, se trataba de un concepto revolucionario. Las conexiones entre una máquina de unidad principal y un terminal exigían un costoso cableado para comunicaciones de alta velocidad, y había problemas con los tendidos de cables más allá de los 20 metros. Se pudo utilizar la red pública de teléfonos conmutados, pero ésta limitaba el intercambio de datos a 9 600 baudios.

En Palo Alto el objetivo consistía en una red de velocidad razonable que enlazara entre sí ordenadores más pequeños, de modo que el usuario dispusiera de un potencial informático local para su propia máquina, así como del acceso a ordenadores más grandes, a grandes almacenamientos en disco y a otros periféricos caros como plotters e impresoras. Ésta fue la base del concepto Ethernet LAN.

En el sistema Ethernet, las conexiones se efectuaron con cable coaxial normal, que es capaz de transportar 10 millones de bits por segundo y tiene aptitud para transportar información digitalizada de sonido y de gráficos, así como datos. Además, el sistema podía alcanzar hasta 500 metros sin necesidad de amplificadores repetidores. Se podía enchufar cualquier dispositivo nuevo derivándolo de la red existente, lo que daba la máxima flexibilidad.

La red física es pasiva: los datos, de la clase que sean, se transmiten alrededor de la red y un *transceptor* actúa como el extremo frontal de cada dispositivo, determinando si el mensaje está destinado a ese dispositivo. De ser así, el transceptor decodifica el mensaje y lo presenta en una forma que pueda ser utilizada por el dispositivo, sea éste un microordenador, una impresora, un plotter, etc.

Para mediados de la década de los setenta el Ethernet ya estaba funcionando. Xerox creyó que si podía conseguir la ayuda de otros fabricantes el sistema se convertiría en un estándar para la comunicación entre ordenadores. Presentó sus diseños a la IBM, y ésta se negó a participar. Sin embargo, a la Digital Equipment Corporation le faltó tiempo para unirse al proyecto. En 1975 Xerox también se aseguró la cooperación del fabricante de chips Intel, que construyó el chip transceptor.

El Ethernet fue puesto a prueba en Suecia, en un complejo experimental de oficinas y fábricas, y al cabo de unas exitosas pruebas fue adoptado por otros fabricantes. Ahora se ha convertido en un estándar internacional de carácter oficial y fabricantes como Hewlett-Packard e ICL, de Gran Bretaña; Siemens, de Alemania, y Olivetti, de Italia, han decidido adoptarlo. Xerox alentó la aceptación del estándar vendiendo las heliografías por el precio total de mil dólares. Todos los productos Xerox se pueden conectar al Ethernet.

Acceso fácil

El servicio Micronet 800 les permite a los usuarios acceder de manera directa a software del sistema central de Prestel y contar con un punto de intercambio de "correo electrónico"

Ian McKinnell

Micronet 800, servicio que funciona en Gran Bretaña, se llama así en razón de que su sitio original era la página 800 de la base de datos de Prestel, sistema de videotex de la British Telecom. Gracias a él, numerosos usuarios de ordenadores personales pueden tener acceso directo a una considerable variedad de software que puede operarse y, tal vez más importante aún, almacenarse en cinta o disco para su utilización posterior, a menudo gratuitamente. Sin embargo, los abonados necesitan hardware y software adicionales.

Puesto que uno accede a una base de datos remota a través de la red telefónica pública, el primer requisito es un modem (modulador-demodulador), dispositivo de hardware que traduce las señales del ordenador en señales aptas para ser transmitidas telefónicamente. Existen dos alternativas: los modems de cable y los acopladores acústicos. La diferencia básica entre ambos radica en que el modem de cable se conecta al sistema telefónico mediante un enchufe, mientras que el acoplador acústico funciona con el auricular del teléfono (material corriente de la British Telecom), convirtiendo las señales en tonos audibles y viceversa.

Se puede utilizar todo modem compatible con el Prestel para acceder al Micronet 800, es decir, que

Conexión con el Micronet
Un BBC Micro puede conectarse al Micronet empleando cualquier sistema compatible Prestel. El modo más sencillo de hacerlo consiste en comprar un paquete completo de hardware (un modem o acoplador acústico), software (una ROM, cassette o disco) y abonarse a Micronet/Prestel en Prism Microproducts. En este BBC, la ROM Micronet está situada de modo que el ordenador está preparado para el Micronet en cuanto usted conecta

debe transmitir a 1 200 baudios y recibir a 75 baudios. Existen tres tipos principales de modems. Los modems 1000 y 2000, aptos para la mayoría de los ordenadores personales para los que se dispone de software de Micronet, ofrecen transmisión en dos direcciones (transmisión y recepción en 1 200 baudios, útil para la comunicación entre usuarios de microordenadores así como con Prestel) y la velocidad estándar de 75/1 200 baudios. En el caso del modem 2000, el cambio está controlado con software. El Acoustic Modem, también disponible para diversas máquinas, se limita a operar a 75/1 200 baudios. Su ventaja reside en que no necesita un punto de conexión especial y puede utilizarse con cualquier teléfono, incluso con uno público, siempre, naturalmente, que el pago previo sea lo bastante elevado.

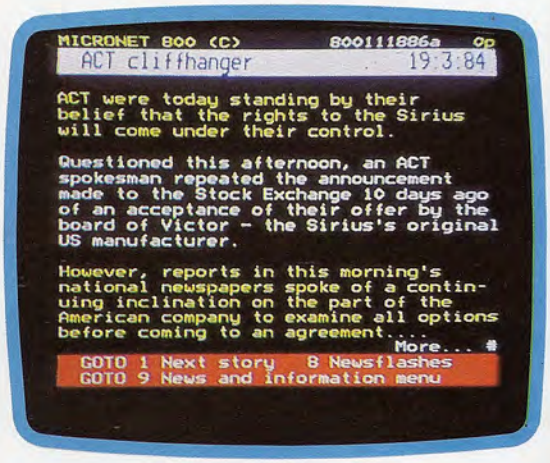
El tercer tipo de modem está hecho a la medida para el Spectrum de Sinclair y se conoce como VTX5000. Al igual que los modems 1000 y 2000, ofrece dúplex completo (transmisiones simultáneas en ambas direcciones) en 75/1 200 baudios y semidúplex (una dirección por vez) en 1 200/1 200 baudios, pero también incluye todo el software necesario para las comunicaciones en forma de ROM, procedimiento mucho más barato que el que se ofrece a los usuarios de otros tipos de microordenadores, que han de comprar software para operar además de hardware adicional.

Dicho software se presenta en dos formas: en

1 Índice general de contenidos
Este menú de contenidos ofrece una buena visión general de la información disponible y le permite elegir qué página abordará a continuación

2 Cartas
Una de las áreas más populares es la página de cartas del usuario; a menudo tanto las cartas como las respuestas de Micronet son un poco crípticas para los novatos

3 Juegos gratuitos
Puede pasar programas de Micronet a su propio ordenador; unos juegos son gratuitos, pero hay otros que se pagan



4 Noticias
Micronet proporciona noticias. Puede actualizar las páginas prácticamente todos los días, por lo que a menudo está adelantado en relación con los semanarios

5 Carlos y Diana
Gallery es una serie de obras maestras de teletexto; también hay juegos, chistes y otros elementos de esparcimiento

6 El tiempo
Abonarse a Micronet le da acceso a muchas otras páginas informativas de Prestel, por ejemplo, la página meteorológica

Liz Heaney

disco o —para el BBC Micro— como ROM sustituible. Los usuarios del BBC pueden optar por la optimización del sistema instalando la ROM en el enchufe de la derecha. En otros ordenadores las funciones son las mismas en cuanto el programa ha sido cargado desde el disco. Puesto que la operación de los sistemas es muy semejante, en líneas generales nos referiremos al Micronet tal como se aplica en el BBC Micro.

El software de decodificación, creado por Scicon —una de las empresas de software más arraigadas en Gran Bretaña—, es necesario para que los microordenadores individuales operen según los protocolos de Prestel, que Micronet utiliza en todo momento (el abonado a Micronet automáticamente cumple los requisitos para acceder a toda la base de datos de Prestel, con su caudal de información sobre todos los temas).

Conexión al sistema

Todos los sistemas de videotex están basados en menús. Al usuario se le presentan una serie de opciones, cada una de las cuales representa una salida de la configuración a la que accede. Cuando el usuario ingresa en el sistema operativo de Micronet, la primera serie de configuraciones solicita códigos de identificación. Se dividen en dos partes y se componen de un identificador personal de diez dígitos y del número de abonado a Prestel. Puesto que Prestel permite que el usuario encargue bienes y servicios, cuyo precio podría ser cargado en su cuenta, dichos números funcionan por derecho propio como tarjeta de crédito. En el caso de Micronet, los números determinan a quién se le cobra el costo del software recibido. Como es lógico, se ha pensado mucho para generar este sistema de seguridad, que parece ser infalible. Por ejemplo, el usuario puede ocultar su número de código de los curiosos incorporando diez asteriscos en su lugar, después de lo cual el sistema le exige que vuelva a incorporarlo, pero esta vez no lo visualiza.

Una vez conectado al sistema, el usuario puede acceder a la base de datos, recibir y enviar mensajes, entregar a voluntad juegos o software comercial... y también recibir, ya que Micronet es un sistema en dos direcciones con alcance no sólo para vender software sino para comprarlo.

Y así ingresa en la base de datos de Micronet propiamente dicha. A fin de ser accesible desde una serie de menús, la estructura de una base de datos ha de ser jerárquica —una disposición tipo "tronco y ramas"—, y Micronet no constituye una excepción a la regla. En cuanto el usuario ha conectado con éxito con el ordenador de Micronet a través del sistema telefónico, la primera opción que se le presenta define el área general en que trabajará: 10 titulares como **What's New** (Qué hay de nuevo), **Computermart** (Mercado informático), **Talking Back** (Réplicas) o **Mailbox/Telex** (Buzón/Télex). Si bien los títulos por sí solos explican de qué temas se ocupan, existen dos "salidas": una que conduce a la página de **Help** (Ayuda) y otra que lleva al Prestel propiamente dicho.

Una de las características principales del Micronet consiste en su capacidad para distribuir software. ¿Cómo se cumple esta operación? El menú adecuado se obtiene a través de **Telesoftware**, y al seguir ese camino se le proporciona al usuario una lista de los tipos de ordenador para los que se dispone de software (cada uno cuenta con su camino de salida propio), así como un menú que ofrece gráficos de los "diez principales" de los ítems más populares, la explicación de cómo vender software a través de Micronet, líneas de **Help** y un anuncio.

Si seguimos la ruta del BBC Micro, llegamos a una configuración que permite elegir entre paquetes individuales de software (juegos, paquetes de gestión, programas de servicios), últimas novedades, *bestsellers*, programas gratuitos y configuraciones **Help**. Al seleccionar una de las cuatro opciones que ofrecen programas aparece una lista de títulos con breves descripciones. La elección de uno de ellos lleva a Micronet a visualizar la configuración conocida como **Downloader Menu** (menú de compra). Esto define las opciones disponibles, que consisten en recibir y utilizar el programa, desconectar el sistema o hacerlo funcionar y continuar conectado al Micronet. Permanecer conectado puede resultar caro desde el punto de vista de las tarifas telefónicas en el caso de que haga usted algo más que guardar en cinta o disco el programa que acaba de adquirir.

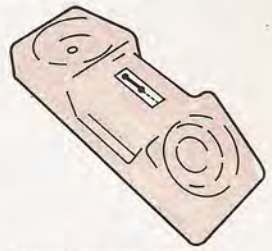
Aunque es verdad que Micronet ofrece software gratuito, la cantidad es limitada, a pesar de que Micronet sostiene que dispone de más de cien programas en cualquier momento dado. Muchos de los programas gratuitos son producciones de aficionados, por lo que no siempre puede esperarse sonido y símbolos gráficos de alta calidad. Los demás programas disponibles tienen diversas tarifas, ya que el precio lo fija el autor.

Puesto que hay que pagar gran parte del software, quizá debería contemplarse este aspecto de Micronet como un sistema alternativo de distribución de programas más que como un servicio público. A decir verdad, los esfuerzos de Prism por encontrar un método de distribución más eficaz no se agotan en Micronet: la empresa es el representante británico del sistema norteamericano Romox, que programa cartuchos de ROM especialmente destinados a la venta al por menor.

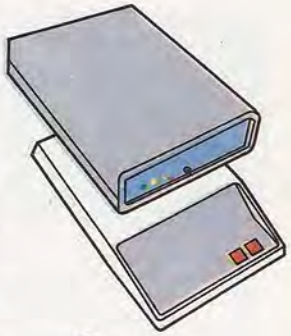
Las restantes funciones del Micronet son semejantes a las del sistema Prestel, del que forma parte: y así, **Mailbox** se encuentra reproducido en el sistema mayor, como las páginas de noticias y anuncios.

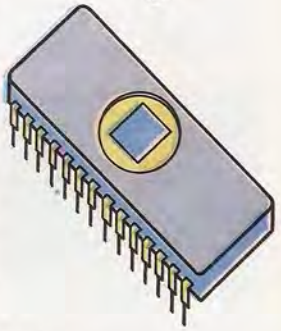
El camino a través de la jerarquía, tal como lo determina la salida escogida por el usuario, se define por un número de identificación de la configuración. Si se conoce su número de identificación, es posible evitar la en ocasiones larga ruta de acceso secuencial conectando directamente cualquier configuración. Se trata de una característica útil para el usuario con experiencia, que probablemente se sentirá frustrado al tener que verse obligado a recorrer página tras página de una información que no le interesa.

Si tenemos en cuenta la cantidad y calidad de los servicios que ofrece, el Micronet no es caro. La suscripción cuesta 1 libra semanal a los usuarios privados que operan desde sus casas y algo menos de 2 libras a los usuarios comerciales y a las escuelas. A ello se añade el costo del modem y, desde luego, la llamada telefónica local que conecta al usuario con el ordenador de Micronet o Prestel... más el costo adicional que se deba pagar por cualquier programa que uno compre.

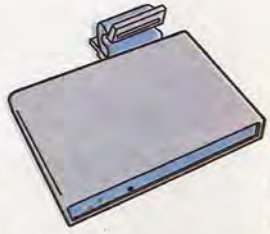


Acopladores acústicos
El acoplador acústico ofrece el modo más sencillo de conectar máquinas como el BBC, el RML 380Z y el Apple a Micronet. Necesitará el software adecuado para su máquina, si bien la lista de ordenadores a los que se puede tener acceso con este software crece constantemente





Modems
Los Modems 1000 y 2000 se conectan directamente con la red telefónica y por ende ofrecen una vía de acceso más fiable al Micronet. El software adecuado para el BBC puede ofrecerse en un chip de ROM, de modo que está disponible desde el instante mismo en que lo conecta



Para el Spectrum
El VTX 5000 es un modem diseñado específicamente para el Spectrum y se conecta al conector de ampliación

Tony Duncan-Smith

Ruedas de fuego

Con la introducción de la unidad de disco para sus modelos 32 y 64, Dragon se mantiene a la altura de sus más serios competidores

Las unidades Dragon emplean discos de densidad única de 13,3 cm. Se forman magnéticamente, según órdenes, con 40 pistas divididas en 18 sectores, cada uno de los cuales posee un espacio de almacenamiento de 256 bytes. En consecuencia, la capacidad total de almacenamiento por disco es de $40 \times 18 \times 256 = 184\,320$ bytes o 180 Kbytes. El DOS (*Disk Operating System*: sistema operativo de disco) necesita parte del espacio de almacenamiento para identificación de archivo, manejo de disco y directorio, lo que deja alrededor de 175 Kbytes de espacio formateado.

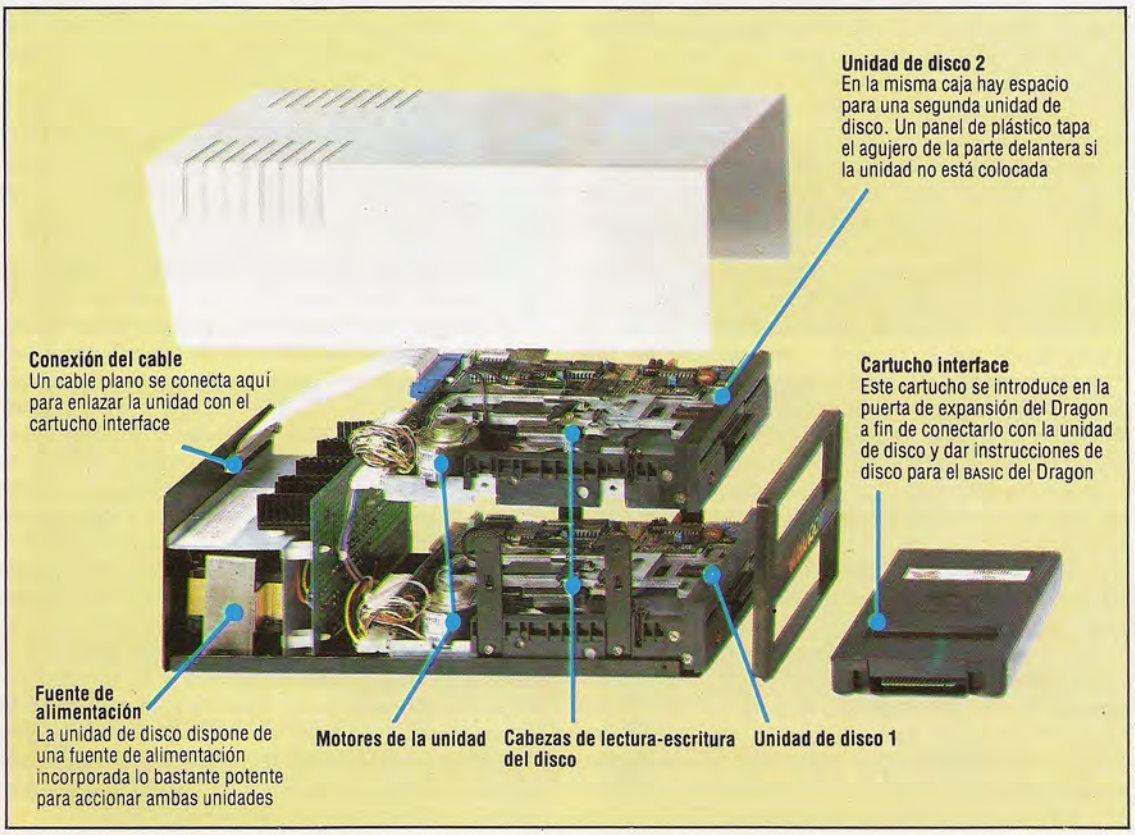
El directorio consiste en la lista de todos los archivos contenidos en un disco. Según instrucciones, visualiza los nombres de los archivos, los tipos de archivo, la cantidad de bytes del archivo y el número de bytes disponibles. Pueden existir cuatro tipos de archivo: programa BASIC, de datos, binario y de seguridad (*backup*). El tipo de fichero almacenado se indica añadiendo .BAS, .DAT, .BIN y .BAK al nombre correspondiente a cada fichero. El DOS de Dragon no menciona de qué forma los archivos de un disco se distribuyen en la memoria y cómo se accede a ellos pero, puesto que no hay órdenes para cerrar el espacio libre de un disco, probablemente existe una forma de BAM (*Block Availability Map*: mapa de disponibilidad de bloques).

Aunque parezca extraño, el DOS de Dragon no puede acceder aleatoriamente a los datos, de un byte por vez. Éste es un medio útil para un programa que requiere leer a menudo pequeñas cantidades de datos, por ejemplo, una base de datos. En su lugar, el DOS utiliza lo que se denomina *acceso directo simulado*, que aparentemente hace lo mismo, salvo que se requieren unas pocas líneas adicionales de código en BASIC para ejecutarlo.

El precio relativamente elevado y la reducida capacidad de almacenamiento hacen que este sistema de disco sólo compense parcialmente su costo. Tal vez consciente de ello, Dragon Data piensa incrementar su variedad de unidades a fin de incluir, a un precio razonable, unidades de 80 pistas, distribuidas en dos lados, y doble densidad. Para el Dragon 64 se está introduciendo como opción un sistema operativo de RAM residente cargado a partir del disco. Denominado OS9, requiere 16 Kbytes de RAM, pero cuenta con un avanzado sistema operativo 6809 que incluye un DOS excepcionalmente eficaz, y en discos separados dispone de PASCAL, C, COBOL y BASIC 09 estructurado.

Opciones de lenguaje
El primer grupo de software de disco de Dragon funciona según el sistema operativo OS9 e incluye lenguajes informáticos avanzados como el PASCAL y el compilador C, que vemos aquí

Paquete sofisticado
El OS9 también permite utilizar sofisticados programas de oficina. Este paquete de tratamiento de textos incluye un verificador de ortografía con un diccionario de 42 000 palabras



Fuerza motriz
Las unidades de disco Dragon se presentan en una caja con su fuente de alimentación incorporada y espacio para dos unidades. Un cable plano la conecta a un cartucho interface que luego se enchufa en el Dragon. Los discos funcionan según el sistema operativo Dragon o el más costoso y profesional tipo software OS9. Cada disco puede albergar 175 K de información

Ian McKinnell

Órdenes del Dragon

Cuando se escriben órdenes, los parámetros que describen el archivo suelen ser los mismos. En este caso, el formato de instrucción utilizado con más frecuencia es:

COMMAND"D:NOMARCHI.TYP"

donde D selecciona una unidad (1-4) y hace caso omiso de la unidad omitida; NOMARCHI consta de hasta ocho caracteres que especifican el archivo, y .TYP es el código de identificación de éste, que puede ser especificado por el usuario; si no lo es, el DOS asume el valor omitido: .BAS. Este formato está representado por FSP.

DRIVE N
En este caso, N puede ser de 1 a 4. Esto selecciona la unidad omitida.

DSKINIT
Esta orden formatea el disco especificado:

DSKINIT D,S,T

En este caso D selecciona la unidad; S el lado del disco (ya sea 1 o 2) a ser formateado, siendo 1 el omitido; T selecciona el número de pistas a formatear (40 u 80), siendo 40 el omitido. Con un sistema estándar de una sola unidad, basta con teclear DSKINIT (ENTER) y el disco insertado se borrará y se formateará correctamente.

DIR D
Esta orden visualiza el directorio de la unidad especificado, D, del siguiente modo:

```
DIR
PROGNAME .BAS 1654
M/C BINARY .BIN 1389
PROGDATA .DAT 2581
PROGNAME .BAK 1654
167322 FREE BYTES
```

ARCHIVOS DE PROGRAMAS

SAVE
Esta orden almacenará programas o archivos binarios en disco. SAVE FSP almacena un archivo de programas. No hace falta especificar .BAS ya que se trata del valor omitido. La orden:

SAVE FSP,CCCC,FFFF,XXXX

almacena un archivo binario. En este caso CCCC es la dirección de inicio en decimal del código a almacenar; FFFF es la dirección final y XXXX es la dirección a partir de la cual se ejecuta el programa.

LOAD
Leerá el programa o archivos binarios del disco con la orden:

LAOD FSP

Si el archivo especificado es binario, FSP puede suplirse por, CCCC. Ello indicará la nueva posición de inicio en la memoria del archivo binario.

RUN FSP
Inmediatamente cargará y ejecutará el programa en BASIC especificado.

CHAIN FSP
Cargará y ejecutará un programa sin modificar las variables almacenadas. Resulta útil en los programas que comparten información. Añadir #,CCCC tendrá el mismo efecto que en la orden LOAD descrita.

FREE D
Visualiza la cantidad de bytes libres en una unidad específica.

COPY
Duplica el archivo ARANTIG como ARNUEVO del siguiente modo:

COPY ARANTING TO ARNUEVO

Si los números de la unidad no se especifican, la copia se hace en el mismo disco en la unidad omitida.

RENAME
Cambia el nombre pero no el tipo de un archivo. Ambos FSP deben referirse a la misma unidad o asume el omitido

MERGE FSP
Esta orden superpondrá un archivo de programas en BASIC específico en otro contenido en la memoria, haciendo así que los programas se fundan. El programa contenido en el disco tendrá preferencia si los números de línea coinciden.

KILL FSP
Borrará un archivo específico del disco.

PROTECT
Esta orden evita que un archivo sea borrado o que se le escriba encima mediante cualquier otra instrucción salvo DSKINIT:

PROTECT ON FSP

También hará que se visualice un campo P invertido con el nombre del archivo en el directorio del disco. La orden PROTECT OFF FSP retirará la protección

BACKUP
Copiará todo el contenido de un disco en una unidad (DA), pasándolo a un disco de otra (DB) con la instrucción:

BACKUP DA TO DB,S,T

S y T tienen exactamente los mismos valores que para la orden DSKINIT, permitiendo hacer copias en discos y unidades de distinto formato. El usuario de discos simples puede limitarse a digitar BACKUP (ENTER), después de lo cual se visualizan las instrucciones para alternar discos de fuente y de destino.

VERIFY
ON y OFF controlan esta orden, que comprueba que el contenido de un archivo almacenado es igual al original.

ARCHIVOS DE DATOS

FWRITE
Se utiliza para crear y escribir un archivo de datos que contenga listas de variables. Para cada FWRITE se abre un canal al archivo, permitiendo añadir nuevos datos a los ya almacenados. Se escribe así:

FWRITE"NOMARCHI";VAR

En este caso NOMARCHI es el archivo a escribir, o a crear y escribir, y VAR es una lista de variables que contiene los datos a almacenar. Comas y dos puntos ponen fin a las cadenas de variables en los archivos de datos, a menos que se tenga el propósito de que sean leídas como cadenas por FLREAD. La orden:

FWRITE"NOMARCHI",FROM SB,FOR TB;VAR

escribe VAR al NOMARCHI comenzando en el byte SB, extendiendo la longitud de la lista hasta un total de bytes TB. Simultáneamente sólo pueden estar abiertos diez archivos a los que se accede por FWRITE.

CLOSE D
Cierra los canales de los archivos abiertos por FWRITE, FREAD y FLREAD.

CREATE"NOMARCHI",LA
Crea un archivo NOMARCHI, de LA bytes.

FREAD
Esta orden se construye como FWRITE. VAR es leído en la memoria o, si tomamos el segundo ejemplo dado, VAR es leído a partir del byte SB. La agúja de lectura es adelantada entonces por los bytes SB+TB.

FLREAD
Se forma como FREAD, pero comas y dos puntos no se leen como terminaciones.

EOF
Se utiliza para indicar la última entrada válida en un archivo que se está leyendo. Ej.:

EP = EOF ("NOMARCHI")

donde EP es 0 hasta que el puntero de lectura lee el registro final y cambia EP a 1.

LOC"NOMARCHI"
Mostrará la posición del puntero de lectura como el número del siguiente byte a leer de un archivo específico.

SWRITE
Almacenará datos en el sector S de la pista T como las cadenas A$ y B$ hasta un máximo de 128 bytes cada una, así:

SWRITE D,T,S,A$,B$

SREAD
Recuperará los datos almacenados por SREAD empleando el mismo formato. A A$ y B$ se les pueden adjudicar diferentes nombres de variables.

Caminos tortuosos

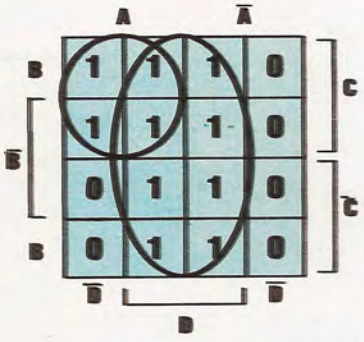
Examinaremos en el presente capítulo los diagramas de Karnaugh cuando se usan para expresiones booleanas con cuatro variables y cómo se aplican en el diseño de circuitos

Cuatro variables. En los casos de expresiones booleanas que suponen cuatro variables, los diagramas de Karnaugh —y las expresiones mismas— pueden parecer sumamente difíciles. Pero si aplicamos las ideas sencillas que establecimos al estudiar los diagramas de Karnaugh de dos y tres variables, pronto comprobaremos cómo se nos hacen familiares y se convierten en fáciles de manipular.

Supongamos, por ejemplo, que nos piden que simplifiquemos la siguiente expresión:

$$AB\overline{C}\overline{D} + ABCD + \overline{A}BCD + A\overline{B}\overline{C}\overline{D} + \overline{B}CD + A\overline{B}CD + \overline{A}\overline{B}CD + B\overline{C}D$$

Vemos que se necesita un diagrama de Karnaugh de cuatro variables y que, si bien esta expresión consta de ocho componentes, en realidad necesitaremos llenar 10 del diagrama (los términos $\overline{B}CD$ y $B\overline{C}D$ representan dos casos cada uno). Así, el diagrama es:



En el diagrama vemos que el grupo central de ocho unos representa todas las combinaciones posibles que contienen la D. El grupo de cuatro unos en el extremo superior izquierdo abarca todos los casos posibles en que están incluidas A y C. En consecuencia, la expresión se simplifica en A AND C OR D (expresada así: A.C + D).

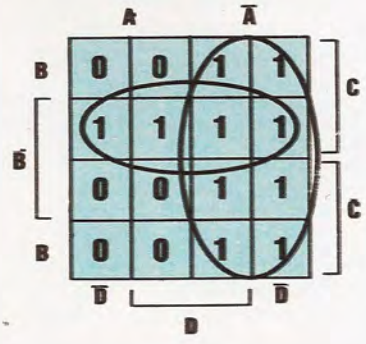
En ocasiones es necesario transformar una expresión a fin de que pueda ser representada en un diagrama de Karnaugh, como en el siguiente ejemplo:

$$\overline{A + B + \overline{C}} + \overline{A}.B + \overline{B + \overline{C}}$$

En este caso primero debemos aplicar la ley de Morgan (véase p. 526) antes de dibujar el diagrama correspondiente. Se reformula así:

$$\overline{A}.\overline{B}.\overline{\overline{C}} + \overline{A}.B + \overline{B}.\overline{\overline{C}}$$

y el diagrama producido por esta expresión es el que se representa a continuación:



En el diagrama de Karnaugh se ve que esta expresión queda simplificada en: $\overline{A} + \overline{B}.\overline{C}$. Recurriendo una vez más a la ley de Morgan, la expresión queda finalmente simplificada como:

$$\overline{A.(B + \overline{C})}$$

Diseño de circuitos

Ejemplo 1: Treinta días

Probablemente conoce la cantinela: "Treinta días trae noviembre con abril, junio y septiembre...". Supongamos que cada mes del año está codificado en un código binario de cuatro bits: 0001 para enero hasta el 1100 para diciembre. Nuestra tarea consiste en diseñar un circuito que acepte el código de cuatro bits como entrada y un 1 de salida si la entrada del mes tiene treinta días.

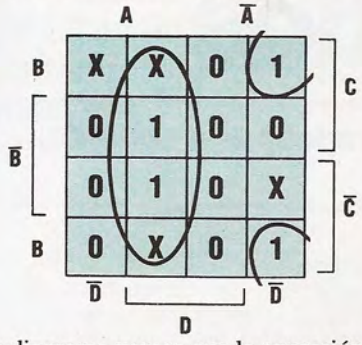
La tabla de verdad de dicho circuito queda así:

MES	ENTRADAS				SALIDA
	A	B	C	D	S
	0	0	0	0	X
ENE	0	0	0	1	0
FEB	0	0	1	0	0
MAR	0	0	1	1	0
ABR	0	1	0	0	1
MAY	0	1	0	1	0
JUN	0	1	1	0	1
JUL	0	1	1	1	0
AGO	1	0	0	0	0
SEP	1	0	0	1	1
OCT	1	0	1	0	0
NOV	1	0	1	1	1
DIC	1	1	0	0	0
	1	1	0	1	X
	1	1	1	0	X
	1	1	1	1	X

La salida X de la tabla de verdad significa una entrada nula. Supondremos que el circuito *no* recibirá esas señales. A partir de la tabla de verdad, para S = 1, podremos formar la siguiente expresión booleana con los bits binarios de la entrada:

$$S = \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.B.C.\overline{D} + A.\overline{B}.\overline{C}.D + A.\overline{B}.C.D$$

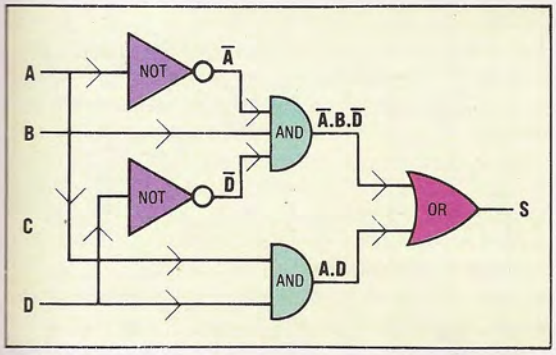
Al dibujarla en un diagrama k junto con las condiciones de "entrada nula" (X), tenemos:



Con este diagrama vemos que la expresión se reduce a:

$$A.D. + \overline{A}.B.\overline{D}$$

Así, nuestro circuito de señales del "mes de treinta días" puede construirse:



Ejemplo 2: Números impares

Puesto que los números del 0 al 15 pueden codificarse mediante cuatro dígitos binarios (del 0000 al 1111), se nos pide que diseñemos un circuito que acepte el código de cuatro bits como entrada y cuya salida dé un 1 si la misma representa un número impar mayor que dos.

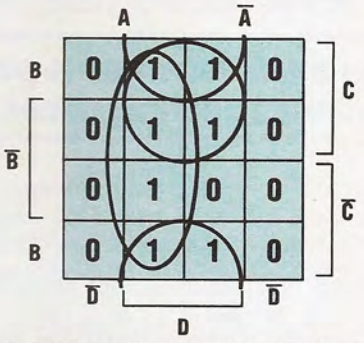
Haremos primero la tabla de verdad:

NÚMERO DECIMAL	ENTRADAS A	B	C	D	SALIDA S
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

A partir de esta tabla de verdad obtenemos la siguiente expresión de álgebra booleana, para todas las condiciones donde S es verdad (= 1):

$$S = \overline{A}.\overline{B}.C.D + \overline{A}.B.\overline{C}.D + \overline{A}.B.C.D + A.\overline{B}.\overline{C}.D + A.\overline{B}.C.D + A.B.\overline{C}.D + A.B.C.D$$

El diagrama de Karnaugh de esta expresión es:



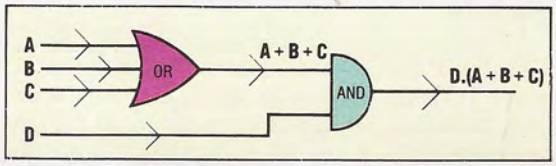
A partir del diagrama, es posible emplear tres grupos de cuatros, representados así:

$$S = A.D + C.D + B.D$$

y puede simplificarse aún más utilizando la propiedad distributiva hasta obtener:

$$S = D.(A + B + C)$$

En consecuencia, el circuito puede diseñarse:



En el próximo capítulo repasaremos los aspectos más importantes del curso de lógica y ofreceremos un amplio conjunto de ejercicios de repaso.

Ejercicio 5

1) Simplifique las siguientes expresiones de álgebra booleana utilizando diagramas de Karnaugh:

a) $A.B.C + A.\overline{B}.\overline{C} + \overline{A}C + \overline{A}B.C + A.B.\overline{C}$

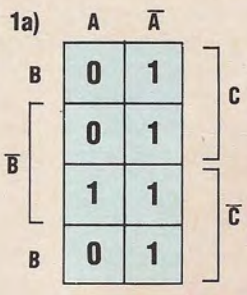
b) $\overline{B + C} + B.\overline{C} + A.C$

c) $A.\overline{B}.D + \overline{A}.D + A.B.C.D + A.B.\overline{C} + \overline{A}.B.\overline{C}.\overline{D}$

2) Diseñar un circuito que acepte las representaciones binarias de los números enteros entre 0 y 7, ambos inclusive. El circuito ha de dar una salida si la entrada numérica es impar o múltiplo de 3 (p. ej., 3 o 6). Dibuje una tabla de verdad y obtenga la expresión simplificada; dibuje un circuito lógico que lleve a cabo esa función.

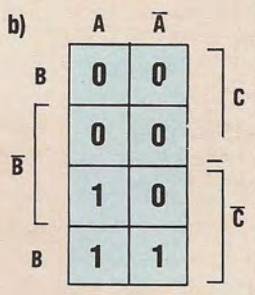
Soluciones al ejercicio 4 de la página 573

1a)



$\overline{A} + \overline{B}.\overline{C}$

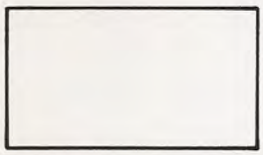
b)



$(A + B).\overline{C}$

Bucles incondicionados

Con un bucle tenemos la posibilidad de repetir una serie de operaciones cuantas veces deseemos



Operación
Con este símbolo se representa una elaboración, o sea la ejecución de una operación o de un conjunto de ellas, tales como cálculos aritméticos, cambios o sustituciones de valores, modificación de datos o de su posición, etc.

Veamos ahora un ejemplo con una operación aritmética. Se nos entregan tres fichas; cada una contiene una cantidad. Debe visualizarse el total de la suma de dichas cantidades y también el cuadro del contenido de cada ficha.

En primer lugar, puede verse que se necesitan una serie de campos destinados a contener los datos que se deben visualizar como final del proceso. En el ejemplo de la figura 1 se han empleado los siguientes: TOTAL para designar el campo en el que se van sumando las diferentes cantidades y C1, C2 y C3 para indicar los cuadrados de las fichas correspondientes.

Así, el proceso queda como sigue: leer una ficha, la cantidad que en ella figura se acumula en el campo TOTAL; después se calcula el cuadrado de dicha cantidad, que queda guardado en el campo correspondiente, y una vez realizadas estas operaciones con las tres fichas, como final de proceso, se visualizan los campos que guardan los tres cuadrados y el de la suma, que contará en ese momento con la suma de las tres fichas.

Este desarrollo del proceso se podría considerar válido en ejemplos tan sencillos como éste, en que se cuenta con un número reducido de fichas, pero supongamos que hemos de trabajar con un número superior de fichas (50 o 100). Entonces la labor, además de extensa, sería reiterativa, ya que se observa que una misma parte del esquema se repite: precisamente, aquella que corresponde a la lectura de la ficha, a la suma de los valores y al cálculo del cuadrado.

La figura 2 nos muestra cómo reproducir una misma parte del proceso. Tras la entrada de la cantidad perteneciente a una ficha, se realiza la suma de dicha cantidad en el campo TOTAL, procediéndose después a averiguar el cuadrado y a visualizar ambos campos. Acto seguido, la línea de flujo retorna al punto inicial, en el que el proceso comienza de nuevo. Ahora bien, en dicho proceso no hay un final lógico, con lo que se obtendrá una repetición ilimitada, ya que al no alcanzar jamás un final, se convertirá en un círculo cerrado en cuyo interior se realiza una parte del proceso. Esta forma de repetición se denomina *bucle incondicionado*. Este nombre se explica por el hecho de que cuando se alcanza el punto en que aparece la línea conductora, el flujo automáticamente sigue su dirección, con lo que se convierte en un reciclaje infinito. Sólo podrá detenerse por la intervención del operador mediante una ruptura voluntaria del ciclo. Esta ruptura será objeto de una próxima lección.

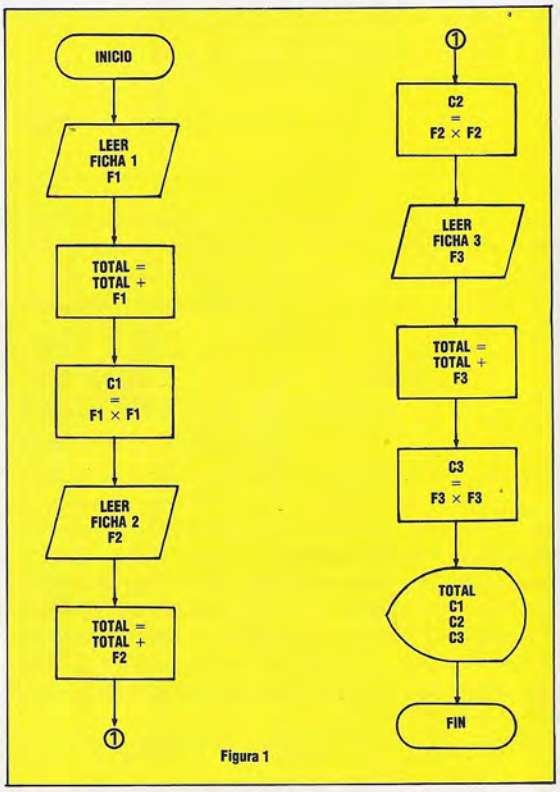


Figura 1

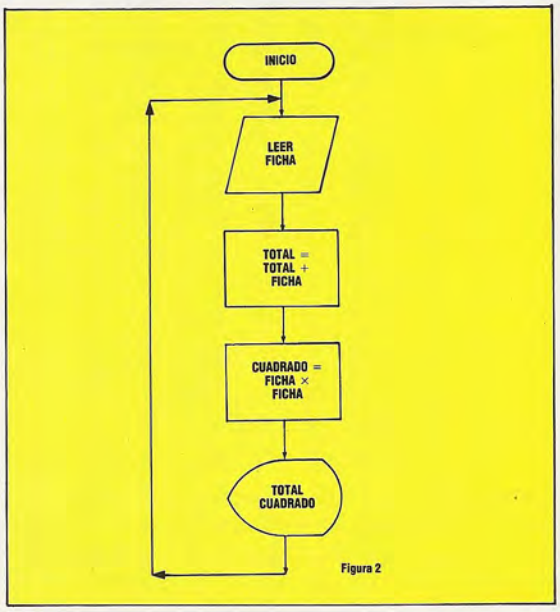


Figura 2

Un diseño austero

El AIM 65 es un microprocesador creado especialmente para aumentar sus conocimientos acerca de cómo funcionan los ordenadores

No todos los microordenadores se alojan en carcasas vistosas, cuidadosamente diseñadas. Diversas máquinas, desde las más sencillas hasta las más sofisticadas, se venden como tableros al desnudo. Uno de los más polifacéticos es el Advanced Interactive Microcomputer, fabricado por la firma Rockwell, más conocido como AIM 65 y concebido como ayuda educativa y de experimentación.

En su presentación más sencilla, el AIM 65 es un tablero al desnudo, desprovisto de carcasa. No obstante, desde el punto de vista operativo, es muy versátil y constituye la única máquina experimental que lleva incorporada una impresora. La mayoría de los sistemas experimentales carecen de un teclado propiamente dicho y no poseen la útil visualización LED de 20 caracteres y 16 segmentos que el AIM proporciona al usuario.

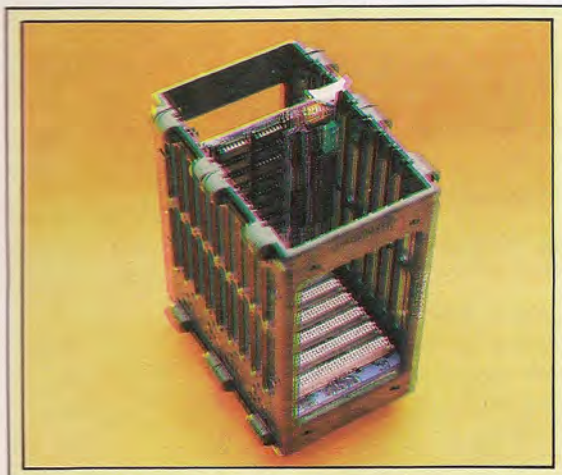
Si bien estas tres características no son decisivas, las máquinas más corrientes —con teclado hexadecimal, visualización de ocho caracteres y siete segmentos y sin impresora— resultan mucho más difíciles de operar. Los requisitos básicos son canales de input/output y memoria, y en este aspecto el AIM es más típico con sus cuatro Kbytes estándar de RAM y 12 Kbytes de ROM; sin embargo, en este sentido está mejor dotado que otras máquinas experimentales, que sólo cuentan con un Kbyte de RAM y dos, cuatro u ocho Kbytes de ROM.

En comparación con los ordenadores de oficina y personales, la mayoría de los cuales poseen un mínimo de 16 Kbytes de RAM y a menudo mucho más, sólo cuatro Kbytes pueden parecer escasos. Pero el AIM 65 también cuenta con dos buenos y sólidos conectores de ampliación, que le permiten añadir más tableros, como, por ejemplo, el tablero de RAM estática de 32 Kbytes. En general, los sistemas experimentales se utilizan en aplicaciones que no requieren una gran capacidad de memoria y es sorprendente lo mucho que pueden hacer con los aproximadamente tres Kbytes que quedan disponibles una vez satisfechos los requisitos de operación de la máquina.

Como el AIM 65 está construido alrededor del microprocesador 6502, todas las partes del mapa de la memoria tienen límites fijos. Ello exige que los 512 bytes inferiores sean de RAM para albergar la

Chris Stevens

Veamos el AIM...
Un sistema AIM completo puede parecer algo desordenado, si bien por razones de elegancia el teclado y el tablero principal pueden presentarse en una carcasa. Es lo ideal si utiliza el AIM en su forma básica de 4 K. Para su ampliación, es posible enchufar directamente una ficha adicional en la parte posterior del AIM. Si aún es insuficiente, ha de usar un chasis de ampliación y conectarlo al ordenador mediante las dos fichas y el cable de cinta que aparecen en primer plano. La caja metálica azul es la fuente de alimentación



Tablero de memoria ampliado
Es posible ampliar el AIM casi hasta el infinito. Éste es un chasis de ampliación que puede contener todavía ocho tableros de circuitos más (también hay para cuatro y dieciséis). Son tableros de tamaño estándar y se acoplan con el conector estándar Euro de 96 patillas. Se pueden comprar fichas para ampliación de memoria (hasta 128 K de RAM y ROM), una interface IEEE estándar y fichas para poder conectar el AIM a unidades de disco flexible, monitores e impresoras de tamaño normal

Conector de aplicación
Este conector terminal tiene dos puertas para conectar el AIM a cualquier dispositivo por controlar

Conector de electricidad
La energía eléctrica necesaria para el AIM (5 v para el ordenador y 24 v para la impresora) se conecta aquí

zona de memoria de información transitoria, que ocupa las posiciones de bytes $0100 a $01FF. La página cero, que va del $0000 al $00FF, es una zona especial utilizada por el 6502 como "seudorregistros". Puesto que los decodificadores de dirección no necesitan incluir la dirección de página (binario 00000000) en la instrucción, en realidad ésta es más corta y mucho más rápida que las regulares. Esta característica forma parte del secreto de la potencia y la popularidad del 6502, pues da al chip 256 registros definibles por el usuario.

Como ya se ha señalado, el AIM tiene una buena cantidad de ROM en comparación con otras máquinas experimentales. Está situada en el extremo superior de la memoria, también en este caso porque es allí donde la localiza el 6502. Dispone de una selección de lenguajes y programas de utilidad, si bien todos dependen del monitor ROM, un bloque de cuatro Kbytes para utilidades de sistema de bajo nivel que reside entre $E000 y $EFFF. Cuenta con un editor de línea, espaciador paso a paso (que ejecuta el programa de a una instrucción por vez, de modo tal que es posible examinar el contenido de la memoria y, probablemente, cambiarlo en cada etapa) y localizador (que visualiza el contenido del registro de posiciones del programa a cada paso durante su ejecución), así como las características usuales de alteración de registro y memoria.

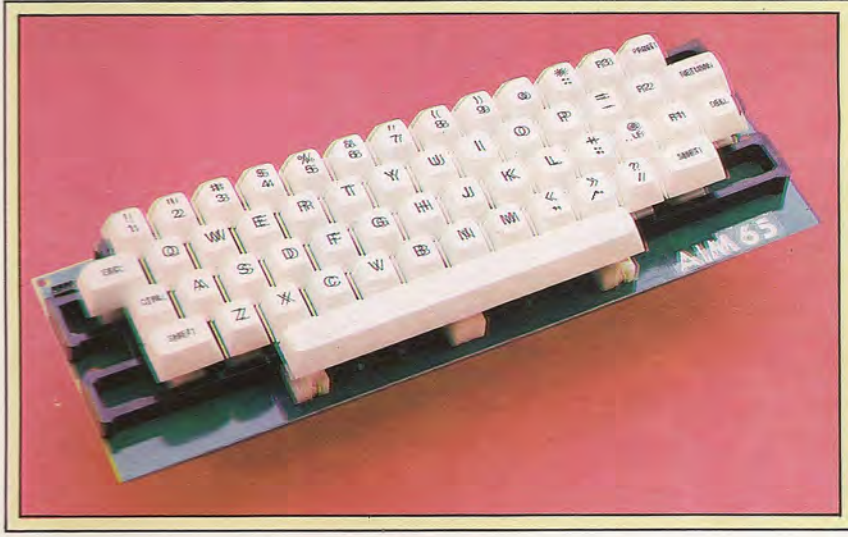
El microordenador se presenta con cinco conectores de ROM, en los que se pueden enchufar diversas opciones de firmware (programas grabados en ROM). Entre éstas existe una versión de BASIC, con números de coma flotante de cinco bytes y funciones aritméticas definidas en una línea.

Para el comprador potencial del AIM 65, que probablemente está interesado en el control y supervisión de procesos industriales u otras aplicaciones afines, resultan mucho más interesantes los otros conjuntos de ROM. Incluyen un ensamblador y un interesante pero poco conocido lenguaje denominado PL/65, que se parece un poco al ALGOL o PL/1 y se compila en un código fuente ensamblador (véase p. 596). Puede ser manipulado, con el fin de "pulirlo", y ensamblado por el ensamblador.

El AIM 65 también dispone de INSTANT PASCAL, un lenguaje poco corriente que ha sido incomprensiblemente desatendido. A diferencia de casi todas las demás versiones de PASCAL, es interactivo e interpretado y ofrece casi todas las ventajas de la estructuración así como la conveniencia y la flexibilidad del BASIC. Sin embargo, el PASCAL es un lenguaje extenso y no encaja a menos que el AIM 65 se amplíe. También dispone de FORTH, que ofrece prácticamente las mismas ventajas. Puesto que exige del ordenador menos que el PASCAL, puede utilizarse sin ampliación.

Impresora
El AIM se sale de los cánones usuales puesto que lleva incorporada una impresora térmica de 20 columnas. Pero resulta práctica para ciertas aplicaciones de control que exijan un permanente registro de los pasos

Interruptor de reactivo
Se emplea para volver a poner en marcha el ordenador



Teclado
El AIM tiene un eficaz teclado que permite programar fácilmente la máquina estándar en BASIC y otros lenguajes; muchos sistemas equivalentes sólo disponen de teclados hexadecimales (sólo 0-9, A-F y unas pocas teclas más) y, por ende, no son fácilmente utilizables. La tecla PRINT del extremo superior derecho se emplea conjuntamente con la tecla CTRL para conectar y desconectar la impresora incorporada. Si la impresora está conectada, todo lo que aparece en la pantalla se imprime en la impresora

Decodificador del teclado
Un chip RIOT 6532 controla el teclado

Chris Stevens

AIM 65

DIMENSIONES

292 x 267 x 60 mm

CPU

6502

MEMORIA

ROM de 4 K, con enchufes para un máximo de 20 K, y RAM de 4 K, ampliables a 64 K

PANTALLA

No hay visualización de video, sino una visualización LED de 20 caracteres y 16 segmentos. Se necesita una ficha de control para establecer una representación visual normal

INTERFACES

Dos puertas bidireccionales de ocho bits, cada una de las cuales tiene dos líneas de control, más bus de sistema

LENGUAJES DISPONIBLES

Se presenta con miniensamblador y editor de línea; también puede disponer de ensamblador completo, BASIC, FORTH, PL/65 e INSTANT PASCAL

TECLADO

53 teclas tipo máquina de escribir, con tres de función

DOCUMENTACION

Manuales de instalación y de hardware que constituyen un modelo de claridad y amplitud y contienen todos los detalles que el usuario pueda necesitar

VENTAJAS

La más importante radica en la extremada flexibilidad de la máquina, atribuible a su capacidad de ampliación y su acceso a diversos lenguajes. La documentación es superlativa

DESVENTAJAS

No presenta ninguna grave. Hay una relativa falta de software de alto nivel, pero hay que reconocer que el AIM 65 es un ordenador de investigadores más que una máquina de aplicaciones

Además del conector de ampliación, que es portador de todas las señales principales —incluida la información y las líneas de dirección, las señales horarias y las líneas de energía—, existen dos chips Versatile Interface Adapter 6522, uno de los cuales se utiliza para controlar la impresora, la interface del teletipo y la interface de la cassette. El otro no está comprometido y aparece en el conector de aplicaciones J2 como dos conexiones bidireccionales de línea ocho más dos. También se ofrece un cronómetro: el 6532 RAM I/O Timer (RIOT), pero este complejo y avanzado dispositivo se destina exclusivamente al manejo del teclado.

En tanto máquina experimental, el AIM 65 no se limita a ejecutar en BASIC o cualquier otro lenguaje de alto nivel. Se prevé que los usuarios programarán sus propias ROM especializadas, que luego podrán conectar en los enchufes de ROM, dedicando así la máquina a un trabajo especializado.

En líneas generales, el AIM 65 es un ordenador de tablero único robusto, flexible y bien sustentado y diseñado. Cuenta con medios suficientes para llamar la atención de todo el que necesite una máquina pequeña, pero muy útil, sin las características de un ordenador de oficina estándar, pero con mayor flexibilidad de la propia de un ordenador personal.

Departamento de contabilidad

Esta nueva serie analizará el mercado de paquetes de gestión, abierto tanto a pequeños usuarios del comercio como a empresas y sociedades

La calificación "paquetes de gestión" abarca un amplio campo de aplicaciones y, además de la diversidad en los servicios, se ofrecen en múltiples proporciones. La primera diferencia en proporción importante es la existente entre el software de cassette dirigido al usuario de ordenadores personales y el software de disco. Como la tecnología de almacenamiento en disco da a los programadores acceso a medios de lectura y escritura rápida, el software de gestión debería basarse en discos, pero no todo el mundo está dispuesto a pagar precios adicionales bastante altos para adquirir un sistema de disco de libre rotación.

Otra distinción es la que existe entre microordenadores de un solo usuario y máquinas de usuarios múltiples, es decir, los microordenadores que pueden sustentar cierta cantidad de terminales separados. Comenzaremos por los sistemas más sencillos y más adelante abordaremos las aplicaciones de las máquinas de usuarios múltiples.

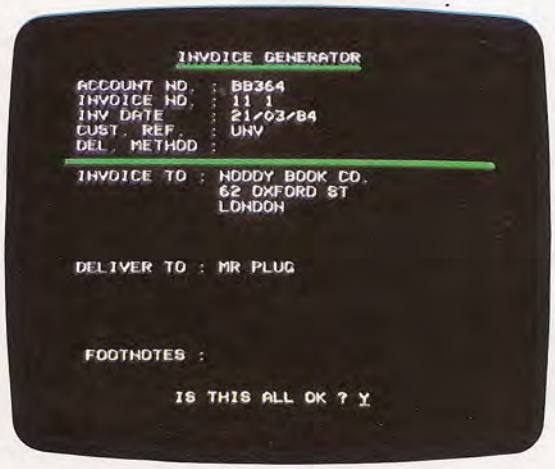
El propósito del software de gestión es que sea utilizado por el profano más que por el especialista en informática. En virtud de ello, ya se trate de un programa diseñado para el sistema de cassette o para el más amplio sistema de disco de usuarios múltiples, por lo general adoptará un enfoque "interactivo". Esto significa que a lo largo de todas las etapas de operación del sistema el usuario es guiado por una serie de orientaciones y mensajes que aparecen en la pantalla. Por lo común, las diversas operaciones se dispondrán como un conjunto de opciones numeradas en forma de "menú". Seleccionar una función del menú principal podría llevar al usuario a un menú secundario o incluso terciario de elecciones. A continuación la pantalla visualizará una solicitud de datos, generalmente información parecida a la que se maneja en un sistema manual de contabilidad. Este uso "sin esfuerzo" es uno de los factores del enorme éxito que los programas aplicados a la gestión están teniendo.

Las tres divisiones funcionales que abordaremos en esta serie son el libro de ventas, el libro de compras y el libro mayor. El libro de ventas consigna los ingresos por ventas realizadas durante el ejercicio económico. El libro de compras registra los gastos por compras necesarias para el tráfico comercial. El libro mayor ofrece una visión global del estado del negocio y su situación financiera en cualquier momento dado.

Los programas de gestión se componen de archivos. Cada archivo contiene cierto número de registros, y cada registro cierto número de campos. La distinción entre archivos, registros y campos es notable. Por ejemplo, un programa de libro de ventas tendrá un archivo maestro de clientes que se compone de los registros de cada uno de los clientes. Los detalles sobre cada cliente, tales como nombre y señas, se dividen en campos dentro del registro.

Además, un programa contable debe seguir un modo preciso en el tratamiento de los datos. Ejemplos de las rutinas de programación necesarias son las rutinas de entrada (que permiten introducir los datos en el sistema) y las rutinas aritméticas, que manipulan valores numéricos en campos numéricos dentro y a través de los registros.

En los "macropaquetes" de gestión (diseñados para ser utilizados con sistemas de discos) se regis-



Liz Heaney

Administración del dinero
Un remedio, aceptable ante las limitaciones de las cassettes, sería la venta de programas independientes que, en un momento dado, puedan aglutinar la información. Este programa de facturación funciona por sí mismo, dejándole a usted la tarea de actualizar manualmente los registros de existencias. Si lo prefiere, puede utilizar archivos generados por los programas adjuntos como parte de un sistema totalmente automatizado. Una ventaja de este enfoque reside en que gradualmente puede ir informatizando su empresa en lugar de someterla a un brusco cambio de sistema

Libro mayor simplificado
El software de gestión en cassette es limitado en cuanto a la magnitud y número de programas contenidos en un paquete. En consecuencia, algunos programas como éste suelen concentrarse en una tarea específica. El paquete contiene programas de contabilidad especiales para cuentas corrientes, resúmenes anuales, etc. Pero sólo se propone ser una ayuda administrativa para controlar el movimiento del dinero y no manejará por sí mismo todas sus cuentas

tran gran cantidad de detalles en los libros de ventas y compras. Así, cada venta o compra se registra en la cuenta de cada cliente o de cada proveedor.

Incluso el más grande sistema para microordenador tiene un límite máximo en cuanto a la cantidad de datos que puede almacenar, dado por descontado que los paquetes de cassette no pueden hacer frente a los detalles de tantas transacciones como los paquetes de disco. En consecuencia, una de las preguntas clave que el usuario potencial de este tipo de software en cassette (y también de software en disco) tiene que hacerse es: ¿podrá este sistema hacer frente al volumen de trabajo relacionado con mi negocio?

Cuantos más datos se almacenan en un archivo, más tarda el ordenador en seleccionarlos, por lo que la solución más común consiste en que los programas sólo retengan información detallada de las transacciones pendientes de pago. Los sistemas basados en este principio se denominan libros mayores *abiertos*.

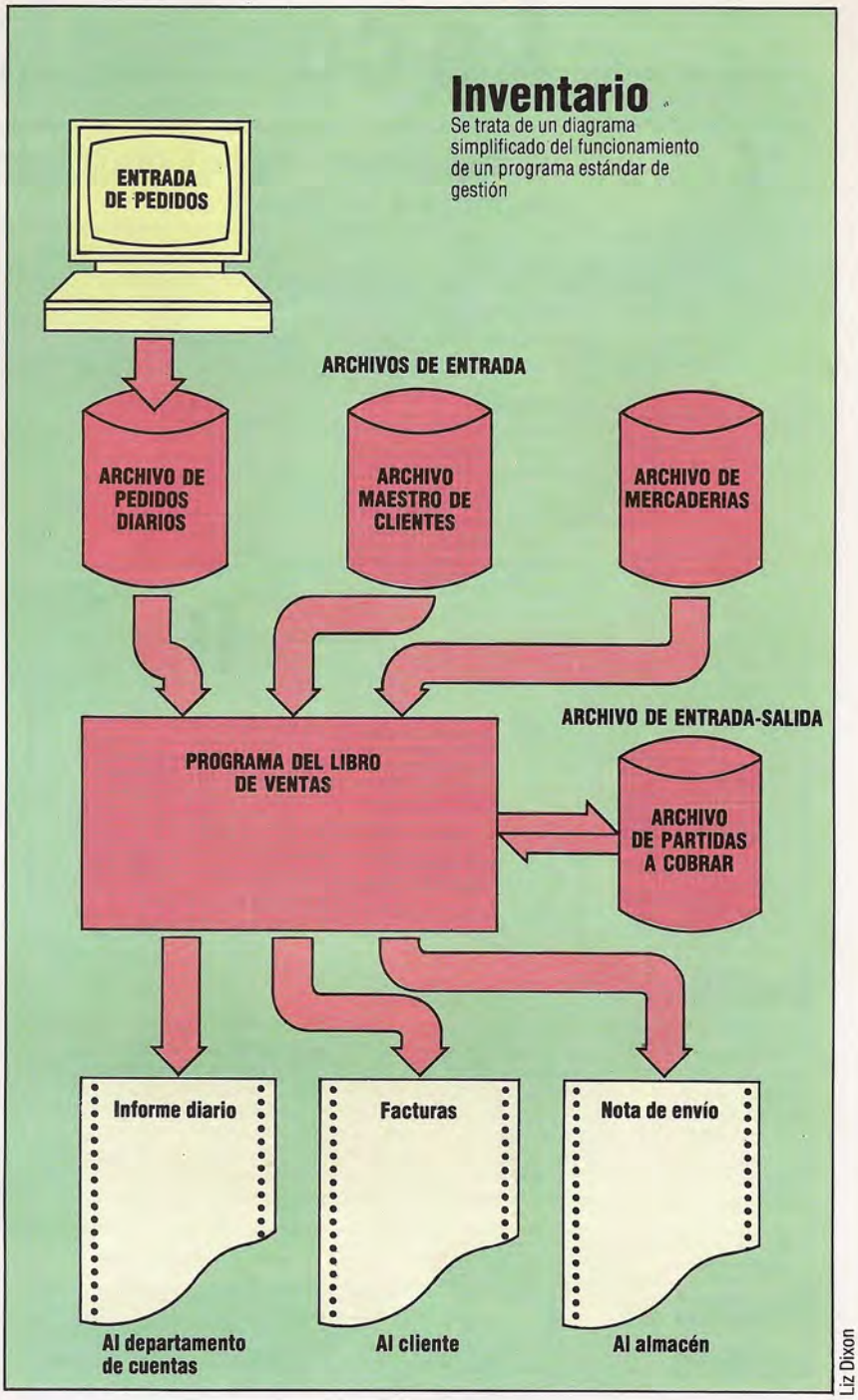
Otra solución suele ser que el ordenador sólo retenga el valor acumulativo de las transacciones entre la empresa y sus clientes o proveedores. Se denomina contabilidad por balances de saldos. Aunque transmite menos información, necesita menos memoria y es más fácil de operar. En resumen, existe un compromiso entre eficiencia (respecto al nivel de detalle que puede registrarse) y lo que se desearía en cuanto a capacidad de procesamiento y memoria de la máquina.

Como veremos a lo largo de las próximas páginas de esta serie, los programas de disco para libros de ventas y compras poseen rasgos específicos que no es posible incorporar en los paquetes de cassette. Por ejemplo, los programas del libro de ventas pueden incluir hasta una opción de facturación que permite a la empresa generar facturas y estados de cuenta para enviarlos a los clientes. Los programas del libro de compras pueden incluso imprimir cheques y acuses de recibo (haciendo una lista de lo que ha entrado) para remitir a los proveedores.

El software de gestión cumple una serie de funciones. El modo más sencillo de comprenderlas consiste en considerar la forma en que el dinero entra y sale de una empresa. El requisito básico de todo el que haga un negocio consiste en averiguar cuántos ingresos tiene la empresa y cuántos gastos ocasiona. El método más sencillo para saberlo es el libro de caja. Un libro de caja no informatizado sino manual lleva las páginas divididas en tantas columnas como necesita la empresa para identificar los conceptos por los que paga o cobra dinero. El libro de caja también tendrá que consignar el valor del IGTE, o del IVA (impuesto sobre el valor añadido) a efectos fiscales.

El libro de caja suele llevarse diaria o semanalmente. Se pueden apuntar las transacciones una a una o, como es más corriente, hacer el apunte del valor total de la recaudación diaria. La diferencia entre el simple libro de caja y un sistema de cuentas completo centrado en el libro mayor y ayudado por libros de ventas, de compras y gastos generales, radica en la falta de detalles del primero.

Los programas de cassette especiales para teneduría de libros disponen de una cantidad muy reducida de memoria. No tienen un disco donde escribir los datos para despejar a continuación una zona de espacio de trabajo de la memoria interna que acepte nuevos datos. Por esto los programas de cassette suelen adoptar el sistema resumido del libro de caja.

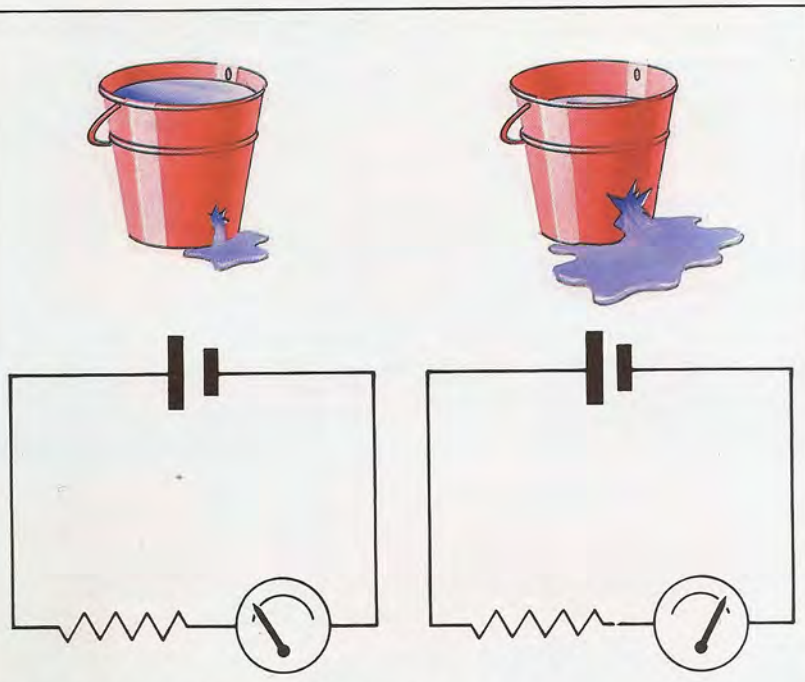


En lugar de contar con tres programas de cassette por separado correspondientes a los libros de ventas. de compras y general, dichos programas suelen unir los tres. Un ejemplo típico sería el programa que lleva cuenta de los ingresos y gastos de un pequeño comerciante en apuntes semanales. Tal sistema tendría como objetivo final unificar y sintetizar los datos incorporados semana tras semana. Para ello tiene que sumar todo el dinero cobrado y pagado durante la semana, restar el segundo del primero y presentar un informe sobre el resultado, favorable o desfavorable, a caja.

En el próximo capítulo analizaremos los requisitos de diseño de dicho sistema.

La corriente y sus usos

Repasemos la ley de Ohm y veamos cómo se puede conectar un monitor a un Spectrum

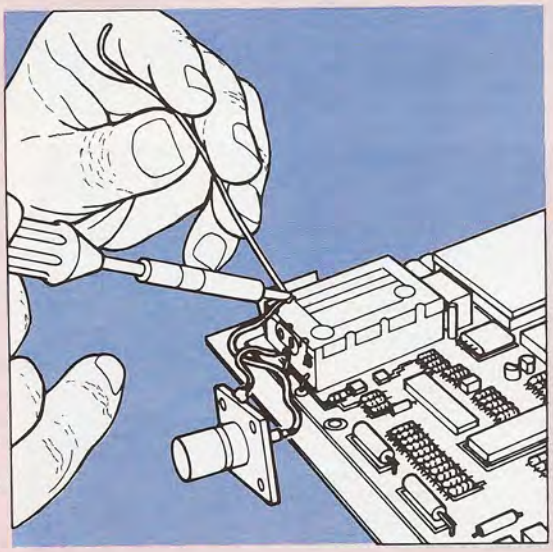


Resistencia y corriente

El componente de un circuito eléctrico ofrece resistencia al flujo de electricidad, del mismo modo que las tuberías y los depósitos de un sistema de calefacción central resisten el flujo de agua: cuanto mayor sea la resistencia ofrecida, menor será el flujo. Si un cubo tiene un pequeño agujero, escapará poca agua debido a que los bordes del agujero resisten el flujo; si el agujero es más grande, la resistencia se reduce y fluye más agua. Así, un circuito eléctrico que porta una alta resistencia tiene un flujo de corriente menor que el que tendría con una baja resistencia. La resistencia eléctrica de un componente depende en parte del material con que está fabricado: los cables son de cobre porque ofrece poca resistencia; los filamentos de las bombillas se hacen de un material altamente resistente, de modo que al pasar la corriente por la resistencia se genere calor y, en consecuencia, luz

Control de calidad

Mediante esta sencilla modificación, el Spectrum de Sinclair proporciona una señal compuesta de video, que permite usar monitor de alta resolución. Abra la caja e identifique el modulador RF, un voluminoso componente plateado situado en el extremo superior izquierdo. El modulador presenta dos conexiones. Busque la más cercana a la parte posterior. Suelde un trozo de hilo conductor desde ésta hasta la parte exterior de un enchufe de soporte de superficie BNC. Suelde otro conductor desde la caja del modulador hasta el conector central del enchufe. Monte éste a un lado de la caja



ATENCIÓN: La garantía de su ordenador quedará anulada si abre la caja

En el último capítulo de la serie *Bricolaje* repasamos algunos de los fenómenos que hacen funcionar un sistema eléctrico. A partir de este momento profundizaremos en la relación existente entre los tres componentes teóricos de la electricidad: potencial, corriente y resistencia. Para ello estudiaremos el modo en que se relacionan y la sencillísima operación aritmética que rige tal relación.

El físico alemán Georg Simon Ohm (1789-1854), que en 1827 enunció la ley fundamental de las corrientes eléctricas que lleva su nombre, fue el primero en deducir la relación entre potencial, corriente y resistencia, y en comprender que cualquiera de los tres valores fundamentales puede deducirse a partir de los otros dos. Comprendió que el potencial era directamente proporcional a la resistencia y a la intensidad de corriente, y una simple operación aritmética le permitió modificar la ecuación, de modo que era posible determinar la resistencia dividiendo el potencial por la intensidad de corriente, y calcular la intensidad de corriente dividiendo el potencial por la resistencia.

Mediante el empleo de los símbolos convencionales de una sola letra V por potencial, I por intensidad de corriente y R por resistencia, las ecuaciones propiamente dichas resultan muy sencillas, y se expresan como sigue:

$$V = R \times I$$
$$R = V/I$$
$$I = V/R$$

Una extensión de la ley de Ohm es la ley de potencia:

$$W = V \times I$$

que relaciona W, la potencia consumida en un circuito o dispositivo, con el potencial y la intensidad de la corriente que lo recorre. Si el potencial en esta fórmula se expresa en voltios y la corriente en amperios, W (la potencia) se mide en vatios.

Esta ecuación es muy útil en el hogar pues nos permite determinar el valor del fusible que debemos aplicar a un electrodoméstico. Tomemos como ejemplo un conector de tres kilovatios, que funciona a 220 voltios. Si dividimos 3 000 (la potencia consumida por el electrodoméstico, expresada en vatios) por 220 (la tensión de la red eléctrica), tenemos que la cantidad de corriente consumida será de 13,6 amperios. Por lo tanto, cualquier fusible de valor inferior a 13,64 amperios saltará. Puesto que el valor máximo de cualquier enchufe doméstico es de 13 amperios, también podemos deducir que cualquier otro dispositivo, como un calefactor de aire de tres kilovatios, que funcione con un adaptador conectado al mismo enchufe no debe consumir más de 120 vatios o estaremos sobrecargando el circuito. En el próximo capítulo llevaremos a cabo algunos experimentos a fin de demostrar prácticamente la ley de Ohm.

Piezas de un circuito

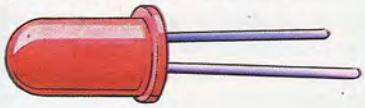
El diagrama de un circuito electrónico parece complicado para quien nunca lo ha visto. A fin de ilustrarlo, hemos elegido el circuito simple de un intercomunicador y descrito uno por uno sus componentes electrónicos. Cada uno de ellos está representado por un símbolo determinado. Cada componente del diagrama también se identifica mediante un código, como por ejemplo "R1" o "TR2". Es un modo práctico de referirse a los componentes, por ejemplo, en una lista de piezas de recambio. Las líneas que conectan los componentes representan alambres. Se dibujan en línea recta por razones de claridad y podrían ser verdaderos alambres o líneas de estaño de un circuito impreso

1 Interruptores

Existen múltiples variedades de interruptores, cada uno con su propia función. Los dos tipos principales son los de retención, donde el estado del interruptor se mantiene al ser presionado, y los de no retención, donde el contacto sólo tiene lugar mientras se mantenga la presión

2 Diodos emisores de luz

Los diodos constituyen la forma más simple del semiconductor. Son el equivalente electrónico de una válvula sin retorno y sólo permiten que la corriente fluya en una dirección. Algunos diodos encapsulados en una resina translúcida despiden una pequeña cantidad de luz y por ello son útiles como indicadores

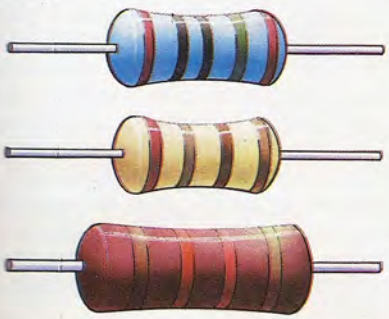


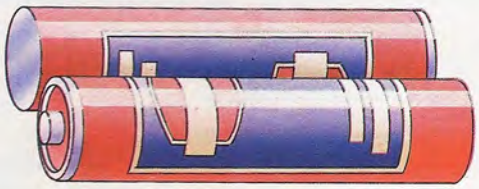
3 Transistores

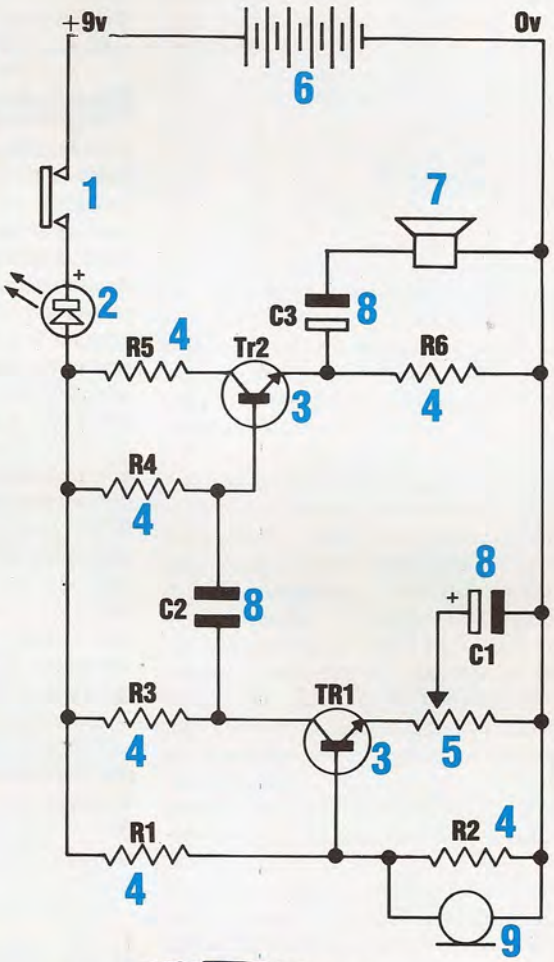
Según su especificación y la forma en que se emplea, el transistor puede hacer las veces de interruptor o de amplificador. Su invención, en 1947, allanó el terreno para posteriores desarrollos en el campo de la microelectrónica

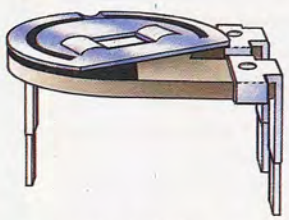
4 Resistencias

Si introducimos en el circuito materiales menos conductores, podemos utilizarlos para controlar el flujo de electricidad. Las resistencias se presentan en muchos tamaños distintos. Su valor se expresa mediante listas de color alrededor de sus cuerpos









5 Resistencias variables

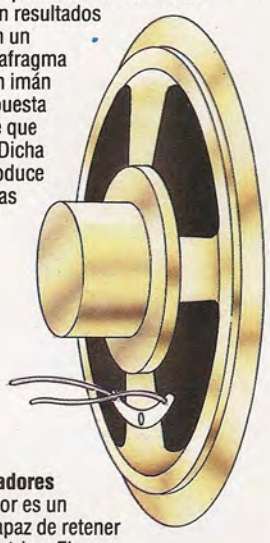
No todas las resistencias son constantes. Las resistencias variables, en ocasiones denominadas potenciómetros, utilizan una faja de carbono como conductor. La distancia que la corriente ha de recorrer a través del carbono antes de llegar al terminal de despegue determina la resistencia del componente

6 Pilas

La mayoría de los circuitos pequeños como éste pueden alimentarse con pilas corrientes pues suministran una corriente continua estable

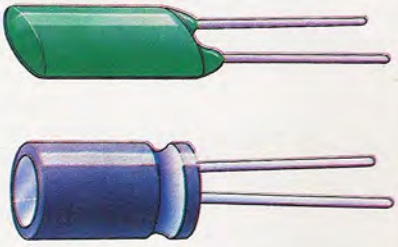
7 Altavoces

Los altavoces son parientes directos de los micrófonos: operan del mismo modo pero obtienen resultados opuestos. En un altavoz, el diafragma adosado a un imán vibra en respuesta a la corriente que se le aplica. Dicha vibración produce ondas sonoras en el aire



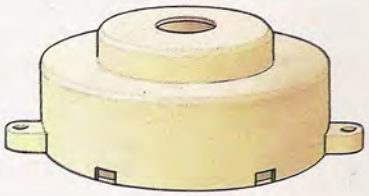
8 Condensadores

El condensador es un dispositivo capaz de retener una carga eléctrica. El condensador se carga cuando sus dos terminales están conectadas a una fuente de alimentación. Una vez plenamente cargado, no ocurre nada, ni siquiera cuando cesa la alimentación, hasta que ambos terminales del condensador se conectan, momento en que se descarga



9 Micrófonos

Los micrófonos funcionan de manera inversa a los altavoces. Las ondas sonoras hacen que el diafragma vibre y esta vibración, a través del imán, produce distintas tensiones en el circuito



Kevin Jones

De byte en byte

Analizaremos el desarrollo de un programa, desde la definición de la tarea inicial hasta el lenguaje máquina propiamente dicho

En los capítulos precedentes hemos visto de qué manera a la entrada de las líneas de un programa BASIC éstas se reducen a distintivos (*tokens*) seguidos de datos ASCII. A partir de ello comprendimos que el BASIC, aunque ciertamente es un lenguaje de alto nivel, no es tan elevado: se compone, básicamente, de secuencias de instrucciones y cada instrucción consiste en una orden (inmediatamente reemplazada por un distintivo, que en sí mismo sólo está un escalón por encima de un código operativo máquina) seguida de los datos para dicha orden. El hecho de que las órdenes y sus datos (variables, números o cadenas) estén más próximos al lenguaje natural y de que las instrucciones se encuentren visiblemente separadas por números de línea o por dos puntos, hace que a nosotros un programa en BASIC nos parezca de mucho más alto nivel que el intérprete de BASIC. De ello se deduce que el lenguaje máquina sólo necesitará unos pequeños "retoques" para que resulte razonablemente comprensible a nuestros ojos.

Estos "retoques" del lenguaje máquina son los creadores del lenguaje assembler, en el que un alfabeto mnemotécnico como LDA y ADC representan los códigos operativos (opcodes) de byte único que el microprocesador comprende realmente, y en el que pueden emplearse símbolos alfanuméricos como LABEL 1 y TFLAG en lugar de direcciones de memoria y datos numéricos. Puesto que el microprocesador no comprende el lenguaje ensamblador, antes de poder ejecutar un programa hay que traducirlo a lenguaje máquina, ya sea mediante un programa denominado ensamblador o manualmente por parte del programador. La utilidad del lenguaje ensamblador consiste en que es lenguaje máquina traducido. Mediante la simple sustitución de códigos operativos por mnemotecnia y números por símbolos, lo podemos convertir directamente en un código ejecutable. Pero a nuestros ojos resulta mucho más comprensible que el lenguaje máquina y, por ende, resulta muy útil en la elaboración de programas. Siempre escribiremos los programas en lenguaje ensamblador y prácticamente no nos ocuparemos del equivalente en el lenguaje máquina hasta las últimas etapas del desarrollo de un programa. Pero en este momento vale la pena hacer ambos por su interés y por razones de total claridad, recordando que, en general, el lenguaje ensamblador hará todo lo que queramos.

El microprocesador puede llevar a cabo muchas operaciones distintas pero, fundamentalmente, lo único que hace es manipular el contenido de la memoria. Lo hace actuando directamente en la memoria del ordenador —los chips de RAM y ROM de que consta el sistema del ordenador— u operando a través de su propia memoria interna, que se compone de *registros*. Estos últimos son varios bytes de memoria físicamente localizados dentro del chip del microprocesador y que tienen determinadas funciones específicas pero que, por lo demás, no presentan ningún carácter que los distinga de los restantes bytes de memoria.

Registro acumulador

El más importante de los registros del microprocesador se denomina *acumulador*. Está directamente conectado a la ALU (*Arithmetic and Logic Unit*: unidad aritmético lógica) y se utiliza con más frecuencia que cualquiera de los restantes registros. A fin de poder emplearlo, debemos estar en condiciones de introducirle información, proceso que se denomina "cargar el acumulador" (*Loading the Accumulator*). Mediante el uso de lenguaje ensamblador, decimos que el 6502 lo hace llevando a cabo la operación LDA, y en el Z80 mediante la operación de LD A. Tomar información *desde* el acumulador es tan decisivo como cargarlo y en el lenguaje ensamblador 6502 ello se logra mediante la operación STA (*STore the Accumulator contents*: almacenar el contenido del acumulador). Empero, el Z80 considera la carga y el almacenamiento como casos distintos de lo mismo, es decir, transferencia de datos. Por lo tanto, tomar información desde el registro acumulador es algo que también se hace mediante la operación LD A, pero en un formato distinto, como veremos más adelante.

Supongamos que queremos escribir un programa de lenguaje ensamblador que copie el contenido de un byte de memoria en el siguiente. Empecemos por copiar el byte $09FF en el byte $0A00. Inmediatamente podemos expresarlo del siguiente modo:

6502	**Z80**
LDA $09FF	LD A,($09FF)
STA $0A00	LD ($0A00),A

Nótese que estamos copiando el contenido del byte $09FF en el byte $0A00 sin saber cuál es dicho contenido: es vital tener clara esta cuestión desde el principio. El byte $09FF puede contener cualquier número desde el $00 al $FF, y todo lo que nuestro programa hace es cargar en el acumulador dicho número y a continuación transferirlo desde el acumulador al byte $0A00. La versión 6502 del lenguaje ensamblador no pone de relieve que LDA se refiera al *contenido* de $09FF, pero distingue inequívocamente entre cargar (LDA) y almacenar (STA). La versión Z80 no hace esta última distinción en sus códigos operativos, pero su formato de instrucción siempre es:

OPCODE DESTINO (FUENTE)

Esta versión pone entre paréntesis direcciones de memoria cuando significa "el contenido de", lo cual refuerza la distinción esencial entre la dirección de un byte y lo que contiene.

El programa que hemos dado es lógicamente completo, pero hemos de ejecutarlo como una subrutina, de modo que para completarlo se necesita el equivalente de la orden RETURN (volver al programa desde la subrutina). Los códigos operativos (opcodes) son RTS en el 6502 y RET en el Z80.

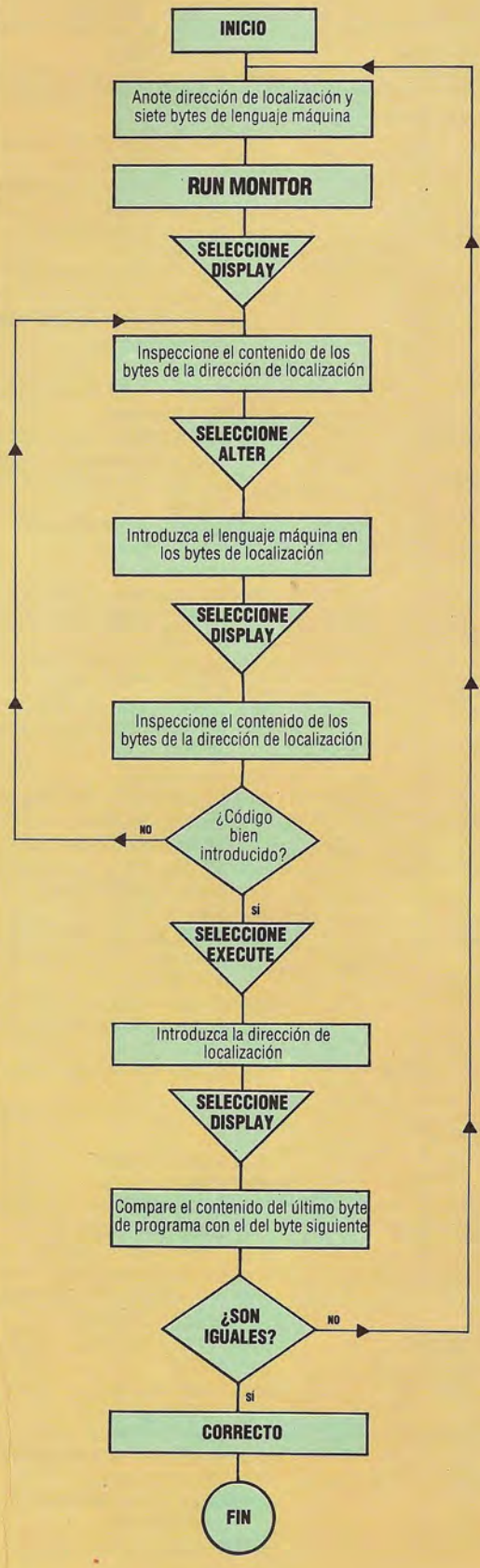
Para usar esta subrutina, primero tenemos que traducirla a lenguaje máquina, luego almacenar el código en algún lugar de la memoria y hacer que la ejecute el microprocesador. Podemos utilizar el programa monitor (véase p. 598) para las dos últimas tareas, pero primero tenemos que realizar la traducción y decidir a dónde irá el código. Esto último es nuevo para los programadores de BASIC, que nunca han de pensar dónde se acumulará un programa de BASIC, ya que se limitan a escribirlo y a teclear RUN. Las decisiones de almacenamiento son tomadas por los diseñadores del sistema en nombre del programador y ejecutadas luego por el sistema operativo.

Un programa en lenguaje máquina puede almacenarse y ejecutarse en cualquier lugar de la memoria, si bien algunos sitios son más idóneos que otros. Los lugares seguros varían de máquina en máquina, de ahí las versiones distintas del programa siguiente:

6502		
Dirección de localización	**Lenguaje máquina**	**Lenguaje ensamblador**
COMMODORE 64		
$0350	AD 56 03	LDA $0356
$0353	8D 57 03	STA $0357
$0356	60	RTS
MICRO BBC		
$0070	AD 76 00	LDA $0076
$0073	8D 77 00	STA $0077
$0076	60	RTS
Z80		
SPECTRUM DE 16 K		
$7FA0	3A A6 7F	LD A,($7FA6)
$7FA3	32 A7 7F	LD ($7FA7),A
$7FA6	C9	RET
SPECTRUM DE 48 K		
$FFA0	3A A6 FF	LD A,($FFA6)
$FFA3	32 A7 FF	LD ($FFA7),A
$FFA6	C9	RET

Nótese que cada versión del programa copia su último byte en el byte siguiente. Por ejemplo, el programa del Spectrum de 48 K copia el contenido de $FFA6 en $FFA7. Nótese asimismo que en lenguaje ensamblador las direcciones aparecen en *hi-lo* (alto-bajo), para nuestro beneficio, pero que en la traducción a lenguaje máquina aparecen en *lo-hi* (bajo-alto). Convendría reparar especialmente en que para el Z80 la mnemotecnia es LD tanto en la primera como en la segunda instrucción, aunque los opcodes difieran: 3A para la transferencia de datos al acumulador y 32 para la transferencia desde el acumulador.

Uso del programa monitor



El programa (véase página siguiente) le permite **ALTER**, (alterar) **DISPLAY** (visualizar) y **EXECUTE** (ejecutar) memoria. Cada vez que seleccione una de estas opciones, se le pedirá una dirección hexadecimal. Es la dirección de la memoria donde:

1) Se comienza a alterar el contenido de la memoria, o

2) Se comienza a visualizar el contenido de la memoria, o

3) Se comienza a hacer que el microprocesador ejecute un programa en lenguaje máquina.

Al introducir "X" en lugar de un número o dirección siempre se le remitirá al nivel de mando y en éste "Q" pondrá fin a la ejecución del programa.

Si selecciona el modo **ALTER** y ha dado la dirección cuyo contenido desea alterar, dicha dirección se visualizará seguida de un interrogante; escriba el nuevo contenido hexadecimal y teclee **RETURN**. La dirección del byte siguiente se visualizará de modo semejante para ser alterada. Mientras quiera alterar bytes, siga escribiendo el nuevo contenido y tecleando **RETURN**. Si incorpora la letra X en lugar de una dirección, será remitido el nivel de mando.

Cargar en la memoria

Observe el programa para su máquina, anote la primera dirección de localización y los siete bytes en lenguaje máquina (p. ej., $0350 y AD, 56, 03, 8D, 57, 03, 60 para el Commodore 64): utilizará el programa monitor para cargar estos siete números hexadecimales en los siete bytes de memoria a partir de la última dirección de localización.

1) **RUN** el programa monitor, **DISPLAY** el contenido de la memoria desde la primera dirección de localización en adelante (p. ej., del $0350 al $0357 en el caso del Commodore 64).

2) Seleccione **ALTER**, dé entrada a la dirección de localización y los siete bytes de que consta el programa en lenguaje máquina.

3) Vuelva a seleccionar **DISPLAY** y cerciórese de que ha ingresado correctamente los bytes en lenguaje máquina en los bytes de dirección de localización.

4) Seleccione **EXECUTE** e introduzca la dirección de localización: parecerá que no ocurre nada.

5) Seleccione **DISPLAY**, inspeccione las direcciones de localización y verá que el contenido del último byte del programa ha sido copiado en el byte siguiente

Programa monitor para el Spectrum

```
48 REM ************************************
49 REM*                                   *
50 REM*          HCAC MONITOR 1           *
51 REM*          ——SPECTRUM———           *
52 REM*       GUARDE ESTE PROGRAMA        *
53 REM*         ANTES DE EJECUTARLO       *
54 REM*                                   *
55 REM ************************************
100 GOSUB 1000 :REM *INIC*
200 CLS
300 PRINT "**HCAC MONITOR 1 CO
    MMANDS **"
400 FOR P = 1 TO LT:PRINT O$(P):NE
    XT P
500 FOR Z = 0 TO 1 STEP 0
550 GOSUB 2000 :REM *INPUT*
600 GOSUB (4500 + CM*500)
650 NEXT Z
700 STOP
750 REM****FIN PROG PRINC****
1000 REM*****S/R INIC******
1050 LET LT = 4:DIM C$(LT):DIM O$(
    LT,24):DIM X$(16)
1100 LET X$ = "0123456789ABCDEF"
1150 LET C$ = "ADGQ":LET C1 = -48:LE
    T C2 = 10-CODE (C$(1))
1200 LET O$(1) = " A - ALTER
    MEMORY"
1220 LET O$(2) = " D - DISPLA
    Y MEMORY"
1240 LET O$(3) = " G - EXECUT
    E M/CODE"
1260 LET O$(4) = "                Q - EXIT
    PROGRAM"
1300 RETURN
2000 REM*****S/R INPUT******
2100 FOR P = 0 TO 1 STEP 0
2150 PRINT:PRINT"COMMAND ??"
2190 IF INKEY$<>" " THEN GO TO 21
    90
2200 LET A$ = INKEY$:IF A$ = " " THEN
    GO TO 2200
2250 FOR J = 1 TO LT
2300 IF A$ = C$(J) THEN LET CM = J:L
    ET J = LT:LET P = 2
2350 NEXT J:NEXT P:IF A$ = "Q" THE
    N RETURN
2400 PRINT O$(CM)
2450 FOR P = 0 TO 1 STEP 0
2500 INPUT "DIRECCION HEXA (X = FUERA)"
    ;A$
2550 GOSUB 5200 :REM CHK&ADJ
2600 NEXT P:IF A$ = "X" THEN LET C
    M = 0
2650 RETURN
3000 REM****S/R BYTE HEXA****
3010 LET HB = INT (N/16):LET LB = N-H
    B*16
3020 LET B$ = X$(HB + 1) + X$(LB + 1)
3030 RETURN
3100 REM*****S/R D-H******
3110 IF NM<256 THEN LET N = NM GOS
    UB 3000:LET H$ = "00" + B$:RETURN
3120 LET HI = INT (NM/256):LET LO = N
    M-256*HI
3130 LET N = HI:GOSUB 3000:LET H$ =
    B$
3140 LT N = LO:GOSUB 3000:LET H$ =
    H$ + B$
3150 RETURN
4000 REM*****S/R H-D******
4050 LET RX = 1:LET DN = 0:LET HL = LE
    N(H$):IF (HL<1) OR (HL>4) THEN L
    ET DN = -1:RETURN
4100 FOR H = HL TO 1 STEP -1
4150 LET D$ = H$(H)
4200 LET V = CODE(D$) + C1*(D$> = "0"
    AND D$< = "9") + C2*(D$> = "A" AND D
    $< = "F")
4250 IF V>15 THEN LET DN = -1:LET
    H = 1:NEXT H:RETURN
4300 LET DN = DN + V*RX:LET RX = RX*16
4350 NEXT H:RETURN
4500 REM*****S/R FICTICIA*****
4550 RETURN
5000 REM*****S/R ALTER*****
5020 FOR P = 0 TO 1 STEP 0
5040 PRINT A$;:INPUT "NUEVO VALOR HEXA
    (X = SALIR)?";V$
5050 PRINT V$
5060 GOSUB 5340 :REM CHK&OBS
5080 NEXT P:RETURN
5200 REM*S/R PRUEBA-AJUSTE**
5220 IF A$ = "X" THEN LET P = 2:RETURN
5240 LET LL = LEN(A$):IF LL>4 THEN
    RETURN
5260 LET H$ = A$:GOSUB 4000
5280 IF DN> = 0 THEN LET P = 2:LET N
    M = DN
5300 LET A$ = A$ + "  ":IF LL<4 THEN
    LET A$ = "0000" (TO 4-LL) + A$
5320 RETURN
5340 REM**S/R PRUEBA-OBSERV*
5360 IF V$ = "X" THEN LET P = 2:RETURN
5380 LET H$ = V$:GOSUB 4000
5400 IF (DN<0) OR (DN>255) THEN
    RETURN
5420 POKE NM,DN
5440 LET NM = NM + 1:IF NM>65535 THEN
    LET P = 2:RETURN
5460 GOSUB 3100 :REM D-H S/R
5480 LET A$ = H$ + "  ":RETURN
5500 REM****S/R DISPLAY*******
5520 FOR P = 0 TO 1 STEP 0
5540 INPUT "NO. DEC. DE BYTES (X = FUERA)"
    ;N$:IF N$ = "X" THEN LET P = 2:NE
    XT P:RETURN
5560 LET BN = VAL(N$):IF (BN>0) AN
    D (BN + NM < (65536) THEN LET P = 2
5580 NEXT P
5600 FOR B = NM TO (NM + BN-1) STEP
    4
5620 LET L$ = "  ":LET NM = B:GOSUB 31
    00
5640 PRINT H$;TAB(6);
5660 FOR C = 0 TO 3
5680 LET N = PEEK(B + C):LET K$ = "."
5700 GOSUB 3000:REM S/R D-H
5720 PRINT TAB(6 + 4*C);B$;
5740 IF N = 0 THEN LET K$ = "■"
5760 IF (N>31) AND (N<128) THEN
    LET K$ = CHR$(N)
5780 LET L$ = L$ + K$
5800 NEXT C
5820 PRINT TAB(26);L$
5840 NEXT B:RETURN
6000 REM*****S/R EXECUTE****
6050 RANDOMIZE USR(NM):RETURN
6500 REM*****S/R EXIT******
6550 PRINT TAB(5);"■■■■FIN DE
    PROGRAM■■■■"
6600 LET Z = 2:RETURN
```

BBC Micro

```
39 REM*************************************
40 REM*          HCAC MONITOR 1           *
41 REM*          ---------BBC----------   *
42 REM*       CAMBIE LA VERSION           *
43 REM*     SPECTRUM DE ESTE MODO:        *
44 REM*                                   *
45 REM*     REEMPLACE CODE(POR ASC(       *
47 REM*                                   *
50 REM*       AÑADA, CAMBIE O BORRE       *
51 REM*          SEGUN INDICACION:        *
52 REM*                                   *
53 REM*************************************
60 *TV 255
70 MODE 7
200 PRINT CHP$(147);CHR$(142)
600 ON CM GOSUB 5000,5500,6000,6500
1050 LT = 4:DIM C$(LT),O$(LT)
1150 C$(1) = "A":C$(2) = "D":C$(3) = "G":C$(3)
     = "Q":C1 = 48:C2 = ASC(C$(1)) - 10
2190 ---------- BORRE ----------
3020 B$ = MID$(X$,HB + 1,1) + MID$(X$,LB + 1,1)
4150 D$ = MID$ (H$,H,1)
4500 ---------- BORRE ----------
4550 ---------- BORRE ----------
5050 ---------- BORRE ----------
5300 A$ = A$ + "  ":IF LL<4 THEN A$ = LEFT$("0
     000",4-LL) + A$
5420 ?(INM) = DN
5680 N = ? (B + C):K$ = " "
5740 IF N = 13 THEN K$ = CHR$ (255)
6050 CALL NM:RETURN
6600 Z = 1:RETURN
```

Commodore 64

```
49 REM *************************************
50 REM*          HCAC MONITOR 1           *
51 REM*          ---------CBM----------   *
52 REM*       CAMBIE LA VERSION           *
53 REM*     SPECTRUM DE ESTE MODO:        *
54 REM*                                   *
55 REM*       CAMBIE SIEMPRE              *
56 REM*     "LET P = 2" POR "P = 1"       *
57 REM*                                   *
58 REEMPLACE CODE (POR ASC)               *
59 REM*                                   *
60 REM*       Y CAMBIE O BORRE            *
61 REM*       SEGUN INDICACION            *
62 REM*                                   *
63 REM *************************************
200 PRINT CHR$(147);CHR$(142)
600 ON CM GOSUB 5000,5500,6000,6500
1050 LT = 4:DIM C$(LT),O$(LT)
1150 C$(1) = "A":C$(2) = "D":C$(3) = "G":C$(3)
     = "Q":C1 = 48:C2 = ASC(C$(1))-10
2190 ---------- BORRE ----------
3020 B$ = MID$(X$,HB + 1,1) + MID$(X$,LB + 1,1)
4150 D$ = MID$(H$,H,1)
4500 ---------- BORRE ----------
4550 ---------- BORRE ----------
5050 ---------- BORRE ----------
5300 A$ = A$ + "  ":IF LL<4 THEN A$ = LEFT$("0
     000",4-LL) + A$
5740 IF N = 0 THEN K$ = CHR$(122)
6050 SYS(NM):RETURN
6600 Z = 1:RETURN
```

Este programa le permitirá visualizar el contenido de la memoria, alterar el contenido de la misma y ejecutar un programa en lenguaje máquina almacenado

Historia de un éxito

Tras recorrer un azaroso camino, las cifras de ventas de Commodore son la envidia de los fabricantes de ordenadores de todo el mundo

El secreto del éxito de Commodore radica en una operación internacional de fabricación perfectamente organizada, representada en Gran Bretaña por una fábrica moderna —construida con subvenciones del gobierno— en la ciudad siderúrgica de Corby. Dicha fábrica se encuentra en proceso de expansión. En el presente emplea alrededor de 250 personas y la producción media asciende a 5 000 ordenadores diarios. Como todas las fábricas de Commodore, se beneficia del suministro asegurado de componentes semiconductores por parte de las propias fábricas de la casa matriz. Commodore es una enorme corporación y, dado su volumen de producción, puede llegar a un trato ajustado con los proveedores del exterior: en algunos casos, sólo paga la mitad de lo que desembolsan sus competidores por chips insustituibles.

La firme posición de Commodore se debe en gran medida al éxito del CBM PET (*Personal Electronic Transactor*: gestor electrónico personal). Su diseño fue esencialmente de Chuck Peddle, quien tomó tres decisiones importantes. La primera consistió en construir la máquina con el procesador 6502; la segunda fue equiparla con BASIC Microsoft, y la tercera, proporcionar un editor de pantalla completo, que permitió que la máquina fuera mucho más fácil de utilizar que los tableros únicos con que los entusiastas de la microelectrónica habían estado jugando desde la aparición del microprocesador, a mediados de la década de los setenta. Hasta hoy, el IBM PC, considerado de una generación más reciente que el PET, carece de un editor de pantalla completo como pieza estándar.

Commodore estaba en condiciones de construir dicha máquina pues ya era propietario de MOS Technology, que tenía los derechos para producir el microprocesador 6502. Este elemento resultó valiosísimo: los dos competidores del PET (el Apple II y el Tandy TRS-80) utilizaban la CPU 6502, por lo que Commodore estaba en condiciones de controlar su producción. De todos modos, el nacimiento del PET planteó problemas. En contra de los deseos de Peddle, Jack Tramiel insistió en que los componentes de memoria del PET también provinieran de MOS Technology. Ello condujo a una discusión muy aireada por la prensa entre ambos hombres y al brusco abandono de la empresa por parte de Peddle.

Ahora el PET es una máquina prestigiosa. Originalmente albergado en una caja de acero prensado (tal como solía construirse el mobiliario de oficina), en la actualidad se presenta recubierto de plástico a fin de modernizar su aspecto. Con esta última apariencia y con un potencial para manejar un almacén de discos de 22 megabytes, se le conoce como la serie 8000. Pese a su antigüedad, sigue vendiéndose extraordinariamente bien a los clientes más conservadores, que se sienten cómodos con él y no encuentran motivos para cambiar su software por la nueva generación de ordenadores de oficina basados en la CPU 8088. Sus proveedores incluso sostienen que, para su propio desconcierto, aún compite con el IBM PC y el Apricot de ACT.

Ello podría ser el resultado de un accidente histórico. Junto con Apple y Tandy, Commodore fue de las primeras empresas en introducir un ordenador personal asequible y fácilmente utilizable. Y en cuanto conquistó clientes, Commodore hizo todo lo posible por no perderlos. Tanto por conservadurismo como por deseo de limitar los costes de producción, Commodore jamás se preocupó por revisar las especificaciones de su máquina para principiantes. La mayoría del software original del PET aún se utiliza en las máquinas de hoy y el BASIC Microsoft versión 2.0 aún es prácticamente igual al adoptado por Peddle.

Lamentablemente para el aficionado más avanzado, el Microsoft 2.0 se desarrolló en una época en que los símbolos gráficos y el sonido constituían un lujo para un ordenador barato. Si bien los recientes ordenadores para aficionados de Commodore están perfectamente provistos de esas configuraciones, el BASIC carece de las instrucciones necesarias y requiere un amplio y laborioso uso de POKE para direccionar posiciones de memoria específicas. Pero esto no es un problema con el software de cartucho escrito de antemano, del que Commodore ofrece una amplia variedad.

Debido, al menos parcialmente, a su provisión constante de componentes, Commodore logró superar la tormenta que hundió a sus competidores en el mercado de juegos. Texas y Mattel se retiraron totalmente del mercado de ordenadores personales y Atari prácticamente ha estado ausente en estos dos últimos años. Gracias al suministro asegurado de piezas baratas, Commodore logró deslizar los precios del software de ordenadores y de cartuchos muy por debajo del precio con el que otros fabricantes podían competir, sin por ello dejar de obtener beneficios. En cierto momento, los cartuchos de Commodore costaban un tercio de lo que costaban los de sus competidores. Una planta moderna y automatizada y el acceso a recambios baratos fueron resultado de la experiencia de fabricación durante dos décadas. Se afirma que el precio de fábrica de un Commodore 64 es sólo la décima parte de su precio de venta.

Commodore no ha alcanzado esta posición de predominio por pura casualidad. Ha tenido sus altibajos y, al menos dos veces, estuvo al borde de la

Cortesía de Microscope

Jack Tramiel
La fuerza impulsora de Commodore ha sido su presidente, Jack Tramiel. Su sagaz capacidad comercial se echará de menos ahora que ha dejado de dedicarse plenamente a la empresa

Cortesía de VNU

Chuck Peddle
Chuck Peddle es el hombre que se encuentra tras el diseño del PET y el chip 6502 que contiene. Creó su propia empresa para producir la máquina de oficina Sirius, de 16 bits

bancarrota. Jack Tramiel, que llegó a Estados Unidos después de la segunda guerra mundial, siendo adolescente, en calidad de refugiado polaco de Auschwitz, fundó la CBM (*Commodore Business Machines*) en 1955. Estableció su sede en Toronto (Canadá), donde la nueva sociedad inició modestamente su producción montando máquinas de escribir con licencia checoslovaca. Se dice que Tramiel eligió el nombre de la empresa precisamente por su parecido con IBM.

En 1975, luego de dos décadas de comerciar con productos para oficinas, la empresa fue abatida por la feroz guerra de las calculadoras, que por el momento ganaron los japoneses. Pero Tramiel, tan temido como respetado por sus métodos comerciales, era un superviviente. Reparó en la existencia de un mercado potencial para un ordenador personal, en 1976 contrató a Chuck Peddle y en menos de una década vio cómo se multiplicaba por cincuenta el valor de la empresa.

Tramiel ha hecho de Commodore un monstruo del mercado y de la fabricación pero, si alguna debilidad tiene, ésta radica en la creación de nuevos productos. La filosofía de la empresa es "vendemos a las masas, no a las clases", y Tramiel está convencido de que el cliente siempre comprará lo que haga más rentable su dinero. Es posible que los requisitos de la fabricación barata a gran escala vayan contra la incorporación de la tecnología más avanzada. A fines de 1982 un alto porcentaje del pequeño equipo de investigación y desarrollo de Commodore abandonó la empresa en una retirada en masa y desde entonces ésta se ha dedicado a comprar los frutos de las investigaciones de otras empresas. Ha firmado acuerdos de fabricación con empresas del Lejano Oriente para producir unidades de disco y ha celebrado conversaciones con firmas como Sony a fin de adquirir costosas tecnologías de la "quinta generación" como reconocimiento de voz, robots domésticos y sofisticados dispositivos de almacenamiento. Incluso ha abordado a Paul Johnson, diseñador creativo de Oric, para tratar de que le diseñe un chip ULA destinado a su nueva serie de ordenadores personales.

En 1984 Commodore se muestra tan confiada como siempre y sigue haciendo una virtud del bajo coste y la sencillez. Ha presentado dos nuevos ordenadores de uso doméstico basados en los nuevos procesadores 7501 y conocidos como el 264 y el V364. Este último tiene un sintetizador de voz y un vocabulario incorporado de 250 palabras. En línea con las tendencias actuales, como opción podrá disponerse de software para tratamiento de textos, hoja electrónica y símbolos gráficos.

Hitos de Commodore

1982
El **CBM 700** es el reemplazo de las máquinas 8032 que promete poner las máquinas de oficina de Commodore a la altura de los micros más modernos

1984
El **SX64** es una versión actualizada del 64, con caja portátil, pantalla en color y unidad de disco

1977
El **Commodore PET** original fue el primer ordenador personal para el mercado de masas. Después de múltiples modernizaciones, sigue vendiéndose bien

1979
El **Vic-20** fue el primer ordenador de uso doméstico barato de Commodore y, pese a sus limitaciones y a una fuerte competencia, sigue siendo muy popular

1983
El **Commodore 64** ha mejorado las limitaciones del Vic al contar con pantalla de 40 columnas y memoria de 64 K

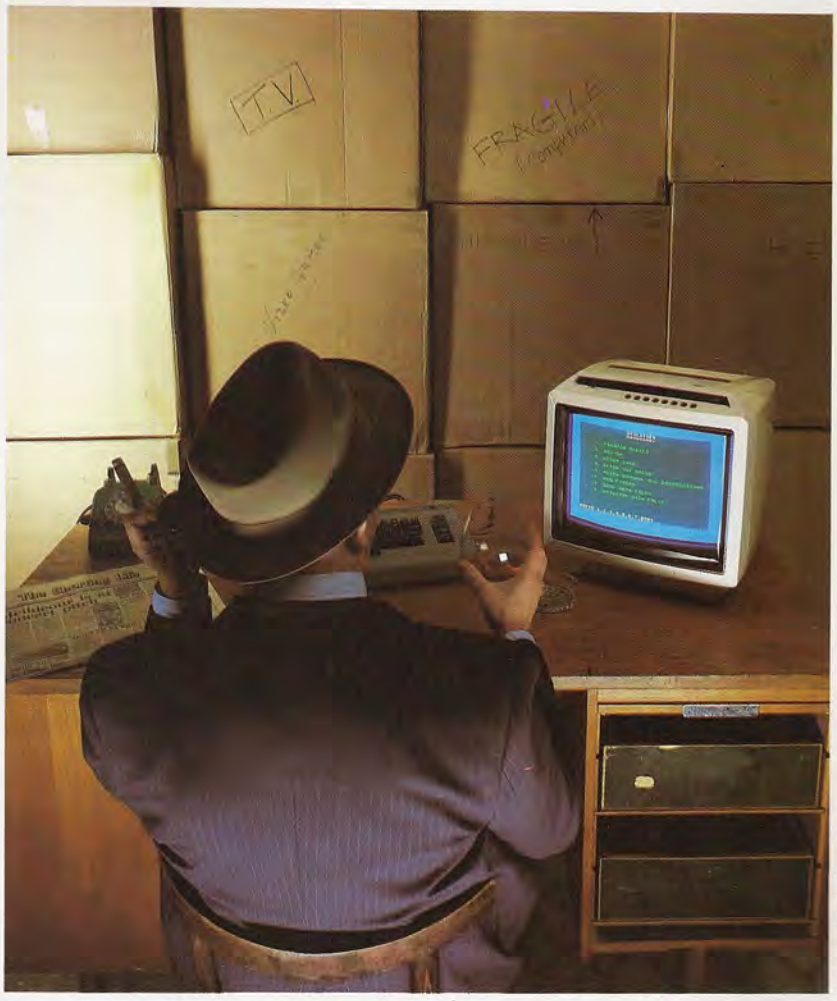
1981
El **CBM 8032** añadió una capacidad de 80 columnas al PET. Ello permite usar avanzados programas de oficina

1980
El **SuperPET** (o CBM 9000) fue un intento de producir una versión de oficina del PET espectacularmente modificada

Ian McKinnell, Chris Stevens y cortesía de Commodore

Sistemas de gestión comercial

Uno de los sistemas de gestión más sencillos consiste en reflejar tan sólo el movimiento de caja del negocio



Marcus Wilson-Smith

Son tres los programas que hemos elegido para ilustrar las exigencias de diseño de un paquete de gestión: *Book-keeping system for the cash trader* (sistema de teneduría de libros para la compra-venta al contado) de Quick-Count, un programa grabado en cinta para el Commodore 64; *Accountant* (el contable), de Compact Accounting Services, pensado para el BBC Modelo B Micro equipado con unidades de disco y el segundo procesador Z80 de Acorn; y *Microledger* (microlibro mayor), de Lewis Ashley Computers, un sistema en disco que funciona en el Apple II.

Si es que ha de satisfacer las exigencias de teneduría de libros, un programa ha de ser capaz de distinguir las diversas fuentes de ingresos y gastos del negocio. El negociante no se conforma con tener un resumen global de cuánto se ha hecho de caja o cuánto se ha gastado en un período dado. Necesita analizar los totales disponiendo de cada uno de los conceptos que los integran. Necesita saber cuánto se le fue en alquiler, en viajes, en pequeños gastos, en papelería y otras partidas similares. Si así no fuera, la única herramienta contable que necesitaría el comerciante sería una máquina de sumar las entradas y las salidas.

Para satisfacer esta necesidad de análisis del movimiento del dinero, el programa ha de dar al usuario capacidad para apuntar los ingresos y los gastos según un "plan de cuentas". El *libro mayor* es un instrumento contable donde se anotan tales cuentas. Constituye el corazón de todo sistema de gestión comercial.

Cash trader nos servirá de excelente ejemplo para mostrar tanto lo que se puede conseguir con un programa en cassette como las limitaciones de esta clase de programas. Consta de un libro mayor que contiene 79 cuentas distintas. Son muy pocas, en comparación con los sistemas basados en disco, el más pequeño de los cuales ofrece centenares de cuentas individuales, pero suelen ser más que suficientes para un gran número de comerciantes que sólo desean identificar los conceptos de mayor interés en su negocio.

El libro mayor de *Cash trader* viene con "estructura previa" (a diferencia tanto del *Accountant* como del *Microledger*, que son libros mayores de formato libre). Esto significa que la codificción numérica de las cuentas está ya establecida en dicho libro. Esto hace que el programa sea más sencillo a la hora de decidir qué operaciones ha de llevar a cabo con los datos de cada cuenta al objeto de elaborar los distintos informes. Facilita además su puesta en marcha por el usuario. Aunque, por otra parte, impone restricciones para una libre y más apropiada reestructuración del libro mayor.

Los números del 01 al 19, por ejemplo, son para las cuentas de ingresos. Por mejor decirlo, se trata de los apuntes provenientes de ventas y otros ingresos (incluidas las mercancías en stock) a favor de la empresa. Sin embargo, puede que el comerciante sólo desee utilizar cuatro de estas 19 posibles cuentas. Una cuenta individual, pongamos por caso la número 01, se podría denominar "Ingresos", y dar entrada (el término técnico es "pasar") a todos los ingresos generados por el comercio. La cuenta 01 llevaría entonces un total acumulativo de todos los ingresos.

Por otra parte, puede que al comerciante le interese obtener saldos separados para, supongamos, cinco categorías distintas de mercancías vendidas. Esto significa la apertura de cinco cuentas de ingresos (del número 01 al 05). Entonces sería necesario preparar procedimientos para asegurar que cada venta se pasa a su cuenta. *Cash trader* no tendría ninguna dificultad con este detalle.

Minoristas
Un micro personal y una cassette pueden bastar para el funcionamiento de un pequeño negocio. La mayoría de los paquetes de este nivel tienen un modesto objetivo, como por ejemplo tratar sólo con el flujo de caja y no con sistemas completos de contabilidad

Los números del 20 al 49 se reservan para las cuentas de compras y otros gastos. Esta sección está diseñada para cubrir los distintos tipos de gastos a que ha de hacer frente la empresa. El comerciante empleará nombres de cuentas como: alquiler, sueldos, impuestos, comisiones bancarias, intereses de préstamos, material de oficina, publicidad, teléfono.

Los números del 50 al 79 corresponden a cuentas de balance. Éstas reflejan el estado financiero global de la empresa en cualquier momento dado. Nombres de cuentas de balance son, por ejemplo, la cuenta de inmovilizado (que muestra el valor de todos los bienes fijos que posee la empresa), la cuenta bancos (o las cuentas, si hubiera más de una), la cuenta del IVA (impuesto al valor añadido), la cuenta acreedores (que muestra lo que adeuda la empresa en concepto de compras y gastos a crédito).

En un sistema de contabilidad manual, el comerciante habría de anotar sus operaciones en los libros de ventas y de compras, y las sumas totales reflejarían los gastos e ingresos del día o de la semana. El menú *Cash trader* para pasar los saldos al libro mayor casi se explica por sí solo:

1. Ingresos diarios
2. Pagos en efectivo
3. Pagos por banco
4. Sueldos

La opción 1, Ingresos diarios, está diseñada para registrar los ingresos de una semana. Le solicita al usuario que seleccione un día (numerados del 1 al 7) y luego que dé entrada al total de ingresos de ese día. La única distinción posible con las ventas es la que permite al usuario señalar un total, mediante una bandera, como una "suma especial".

Así se atiende al hecho de que no todos los ingresos provienen de ventas ordinarias. Si una tienda de jardinería vende su furgoneta, obtiene un ingreso extraordinario, y englobar ese efectivo en los ingresos por ventas del día produciría una cifra de ventas distorsionada. El *Cash trader*, por sorprendente que parezca, tratándose de un programa tan sencillo, posee las facilidades necesarias para identificar las ventas "especiales". Pero éste es el único análisis o detalle del total de ventas diarias que permite el programa. El usuario puede optar por pasar la cantidad a una de tres cuentas nominales: la cuenta de bancos, la de caja o la cuenta transitoria de tarjeta de crédito.

En cuanto a los pagos (el equivalente al libro de compras), el sistema permite que el usuario pague ya sea mediante la cuenta de caja o la cuenta de bancos. Un código de referencia de tres dígitos identifica cada una de las operaciones de pago. Y permite una descripción narrativa de 16 caracteres del motivo de cada desembolso (por lo general, el nombre del proveedor). Asimismo, calcula el IVA repercutido sobre los pagos y pasa la cantidad a la cuenta de control del IVA.

Todo aquel que utilice el *Microledger* o el *Accountant* habrá de organizar un plan contable similar. No obstante, entre ambos existe una gran diferencia dado que estos dos sistemas carecen de un orden predefinido para las cuentas. Con el *Accountant*, por ejemplo, se le otorga al usuario una estructura de códigos de ocho dígitos y éste puede hacer corresponder cualquier cuenta con los números que guste, cualesquiera sean. El *Mocroledger* posee una estructura de códigos de tres dígitos, pero se aplica el principio de formato libre.

El programa *Accountant* también se ha elaborado en torno a un libro mayor general, aunque mucho más sofisticado, capaz de tratar con muchas más cuentas. También dispone de una facilidad que posibilita el análisis. Pero obliga al usuario a resumir hasta cierto punto las entradas de datos, en comparación con un sistema completo de libro de ventas y de compras.

Con un programa completo de libro de ventas y libro de compras, el usuario tiene los medios para llevar un archivo maestro de los clientes y de los proveedores, respectivamente. Este archivo incluirá detalles completos de la cuenta del cliente o del proveedor y también llevará un registro completo de todas las transacciones más sobresalientes llevadas a cabo con ese cliente o ese proveedor. El usuario tendrá la posibilidad de llamar la cuenta a la pantalla y disponer de un listado completo de todas las facturas pasadas a esa cuenta y de todos los recibos de pagos efectuados en dicha cuenta.

El *Accountant* no llega a proporcionar estos medios. En cambio, al igual que el *Cash trader*, asume el enfoque del apunte diario. Pero como es un sistema en disco que dispone de un espacio de memoria considerablemente mayor, no exige que la información tenga una entrada de forma tan condensada.

En lugar de dar entrada a una sola cifra por día de la semana, el usuario puede realizar todas las entradas que sean necesarias. El sistema reconoce cinco tipos distintos de operaciones: facturas, créditos concedidos, recibos de caja, ventas al contado y recibos varios. Cada una de las entradas se puede describir con un máximo de 16 caracteres, se les puede otorgar un número de referencia exclusivo y se las puede analizar en cualquier número de cuentas del libro mayor general (en vez de la simple opción de tres posibilidades que ofrece *Cash trader*).

El libro de compras puede ocuparse de las compras a crédito, compras al contado y otras compras. También posee una facilidad para informar detalladamente sobre el IVA.

El *Microledger*, a diferencia de estos dos sistemas, lleva un libro mayor completo de compras y ventas. El usuario puede crear hasta 999 cuentas individuales del libro mayor de compras y ventas. Cada cuenta registrará el nombre del cliente y hasta cinco líneas de dirección. La cuenta mantiene de forma automática un balance acumulativo y una lista de todas las operaciones más relevantes.

La principal diferencia existente entre el *Microledger* y los otros dos paquetes que hemos descrito previamente, es que éste contiene una gran cantidad de información sobre los negocios efectuados con cada cliente y con cada proveedor. El *Cash trader* condensa esa información en una o dos sumas totales. El *Accountant* puede procesar transacciones individuales con clientes o proveedores, pero el usuario necesitará llevar un libro de ventas o compras manual con el objeto de ver con facilidad cuál es el saldo de cada cliente o proveedor.

En el próximo capítulo de esta serie continuaremos con la comparación entre estos tres paquetes, analizando las formas en que manipulan la entrada de valores en los títulos de cuentas nominales.

Justificantes
Arthur está haciendo sus cuentas de la semana. Ha vendido tres cajas de whisky Glen Kyushu al contado, y 10 cajas mediante tarjeta de crédito; ha recibido un talón de 500 libras esterlinas por servicios prestados y ha vendido el Rolls Royce en efectivo. Sus pagos de la semana han sido la compra al contado de más whisky, pagarle en efectivo a Terry, su único empleado, y un talón para comprarse un coche nuevo

Entrada de cobros
A todos los ingresos se les da entrada de esta forma, y se marcan como ARTÍCULOS o ESPECIAL a tenor de la naturaleza de la venta, y EFECTIVO, BANCO o TARJETA DE CRÉDITO según la forma en que haya pagado el cliente

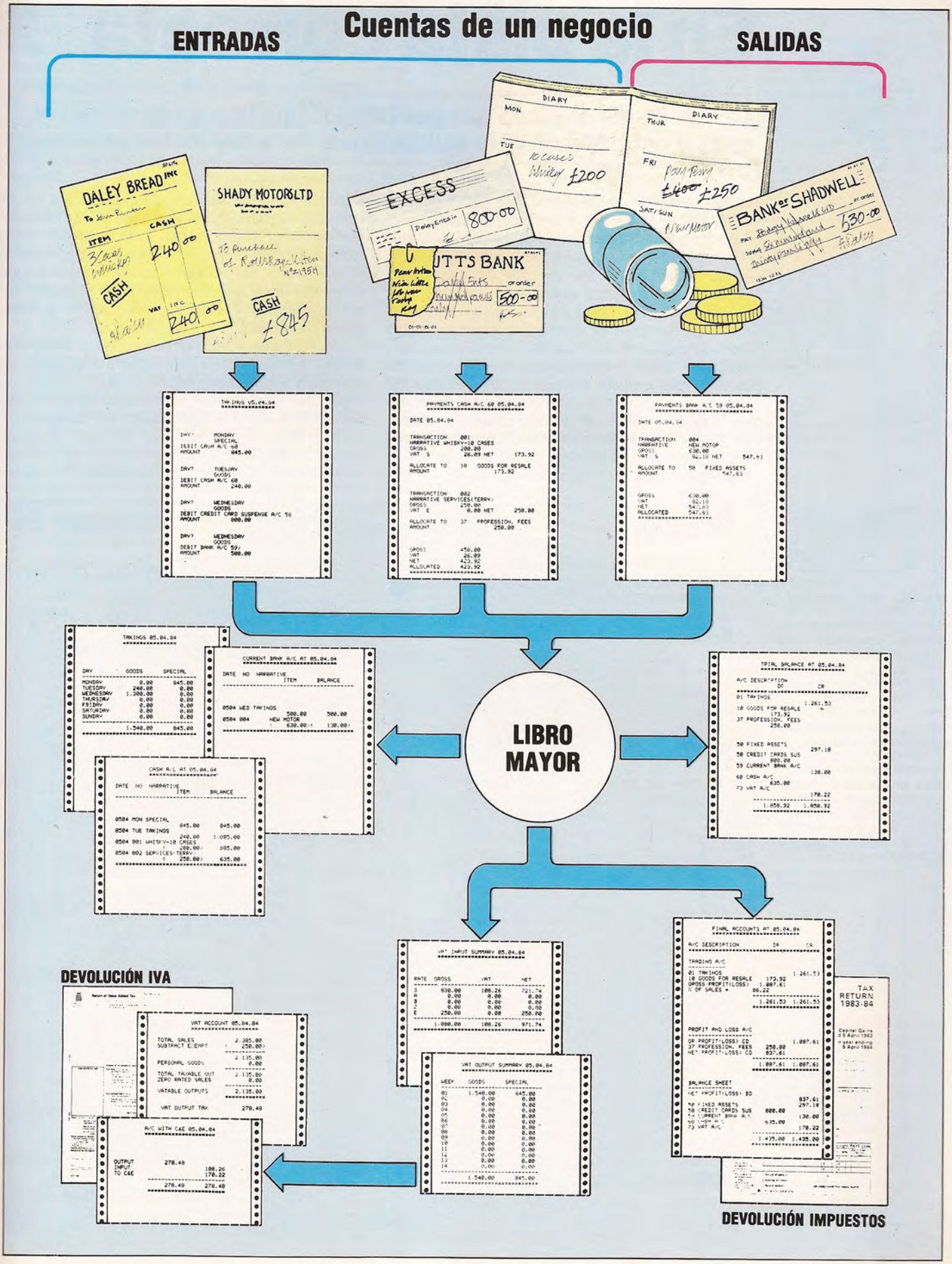
Entrada de pagos
A todos los pagos se les da entrada de la misma forma, indicando EFECTIVO o BANCO, la parte del IVA y el número de código de la cuenta del libro mayor que recogerá la cantidad

Informes
Estos resúmenes se generan automáticamente una vez finalizada la entrada COBROS o PAGOS

Libro mayor
Consta de hasta 79 cuentas nominales tituladas: COBROS, STOCK, SALARIOS, ALQUILERES, CARGOS BANCARIOS, ACTIVOS FIJOS, CAJA, etc. Éstas, al objeto del balance, están agrupadas en la cuenta de COMPRA Y VENTA, la cuenta de PÉRDIDAS Y GANANCIAS y la HOJA DE BALANCE

Informes
Se puede disponer de diversos informes: el estado de la cuenta de CAJA y la cuenta BANCARIA y un BALANCE DE COMPROBACIÓN son lo que la mayoría de los comerciantes desearía ver después de dar entrada a las operaciones de una semana o de un día. La cuenta FINAL y el informe del IVA probablemente sólo se requerirán una vez por trimestre

Cuentas de un negocio



Tony Duncan Smith

En la pista correcta

Las unidades de disco han posibilitado muchos adelantos en el campo del software para aplicaciones. Pero hay inconvenientes...

Cuando usted adquiere una caja de discos nuevos, no los puede utilizar de inmediato. Antes de usarlos debe formatearlos para su micro particular y, a pesar de que los principios básicos del procedimiento para dar formato son los mismos, los detalles varían de una máquina a otra.

Formatear es en cierta manera algo así como trazar los márgenes sobre una hoja de papel en blanco antes de escribir en ella. Un disco nuevo es un disco flexible de plástico cubierto con una caja magnética. Para usar el disco, el micro escribe información que lo divide en un conjunto de pistas concéntricas y que subdivide éstas en sectores como si se tratara de "cortar un pastel en raciones". Algunas unidades de disco dan un formato de 40 pistas de datos, otras de 80. Algunas poseen dos cabezas de lectura-escritura y, por consiguiente, dan formato a las dos caras del disco, otras sólo escriben y leen en la superficie superior del mismo. Esto se conoce como *sectorización soft*, porque es el proceso de formateo el que marca los sectores. La *sectorización hard* empleaba unos agujeros perforados alrededor del borde interior del disco para marcar los sectores, pero esta técnica casi ha desaparecido.

Una vez que se han efectuado estas divisiones, apenas la tercera parte de la superficie del disco es destinada a almacenar información. La última variable que incide sobre la cantidad de datos que caben en un disco es la *densidad*. En cualquier superficie determinada, un disco de doble densidad guarda el doble de lo que guardaría un disco de densidad simple, y así sucesivamente. Por consi-

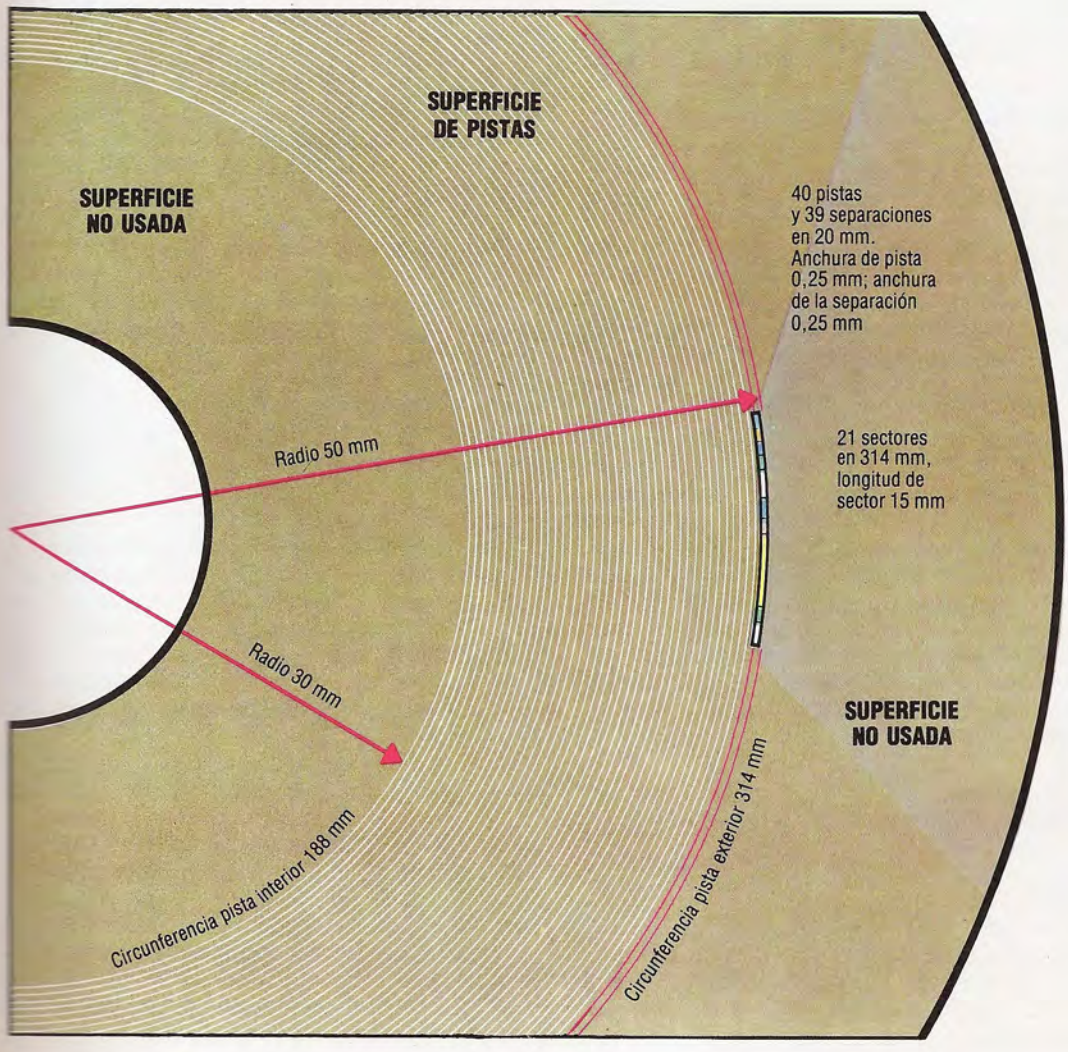
Cuatro formatos
Aun cuando utilizan el mismo tipo de disco, el BBC Micro, el Dragon, el Commodore 64 y el Atari, todos formatean sus discos de diferente manera. El número de archivos separados que se pueden almacenar en el disco depende del sistema operativo. Observe que el posicionamiento en el disco de la lista de archivos del DOS (la pista del catálogo), en un disco BBC se halla en el borde exterior, mientras que en las otras máquinas está en una pista central. Se suele preferir la posición central, ya que la cabeza del disco, por término medio, tendrá menos distancia que recorrer cuando se desplace entre la pista del catálogo y las pistas de datos reales, haciendo que todo el proceso de acceso al disco resulte más rápido

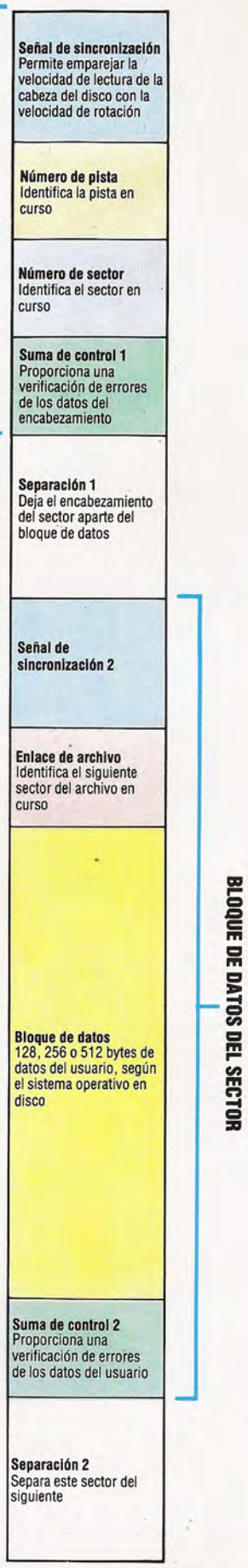
guiente, las capacidades de los discos flexibles varían, desde unos 90 Kbytes en el caso de un disco de densidad simple de una sola cara, hasta 1,2 Mbytes en los de densidad doble y dos caras.

Una vez formateado, usted puede copiar en el disco programas e información. Esto no se realiza directamente. Todo se conserva en archivos y se controla mediante el sistema operativo en disco (DOS: *Disk Operating System*). Éste lleva en un directorio la lista de los archivos que contiene el disco y utiliza entonces los sectores libres que resulten convenientes para almacenar la información. Físicamente, un archivo puede estar esparcido a través de todo el disco, y el DOS también llevará un índice de cuáles son los sectores que ocupó un archivo, dejando señaladores en cada sector indicando dónde se halla la siguiente parte del archivo.

La capacidad de tomar un disco de una máquina y utilizarlo con otra depende de dos cosas: de cómo sea el formato del disco y de cuántos archivos estén acomodados en el mismo. Por suerte, en la actualidad muchos micros emplean el mismo formato de disco o, al menos, poseen programas de utilidades especiales que pueden leer y escribir formatos extraños. Muchas máquinas utilizan sistemas operativos estándar como el CP/M (*Control Program Microprocessors*: programa de control para microprocesadores) y MS-DOS (*Microsoft Disk Operating System*: sistema operativo en disco Microsoft), y los archivos se acomodan de la misma manera.

Pistas y sectores
En el disco los datos se escriben en pistas concéntricas. Cada pista está dividida en sectores. Un sector contiene un bloque de los datos del usuario, y a este bloque el DOS le adjunta automáticamente los datos de verificación de errores y de identificación necesarios para la administración de archivos y registros





La respuesta lógica

Esta vez haremos alto en nuestro curso y resumiremos las reglas de lógica y álgebra booleanas estudiadas hasta ahora

Vamos a efectuar una breve revisión del trabajo que hemos realizado en las lecciones anteriores. Los principios de la lógica se aplican a los ordenadores en el diseño del hardware que debe llevar a cabo ciertas operaciones especiales. Ya hemos diseñado un circuito sumador, el cual al combinarse con otros circuitos sumadores permite la adición de números binarios (véase p. 528). Este circuito reproduce el método tradicional de la suma, permitiendo llevar dígitos de una columna a la siguiente.

En el diseño de este circuito utilizamos tres elementos básicos, denominados puertas lógicas. Las funciones que podían realizar estas puertas se indicaban mediante los nombres que les habíamos asignado (AND, OR y NOT), y definimos cada una de ellas mediante una tabla de verdad. La tabla de verdad es una forma sencilla de reflejar por escrito la(s) salida(s) de un circuito para cualquier posible combinación de entradas. Con dos entradas son posibles cuatro (2^2) combinaciones de entradas, con tres entradas se hacen ocho (2^3) combinaciones, y así sucesivamente. Asimismo, vimos cómo las puertas lógicas se pueden enlazar entre sí para producir ciertas salidas deseadas. Estos circuitos más complicados también se podían describir mediante sus tablas de verdad. En particular, diseñamos un circuito OR, exclusivo de cinco puertas, que nos dio una salida verdadera sólo cuando *una* de sus entradas era verdadera (véase p. 512).

Álgebra booleana

La combinación de los elementos lógicos se puede describir sobre el papel mediante un conjunto de símbolos muy parecidos a los del álgebra normal. La rama de las matemáticas que se ocupa de la representación de la lógica se denomina álgebra booleana, en honor del matemático inglés George Boole (1815-1864), que fue quien definió por primera vez sus principios. Cada uno de los tres elementos básicos de la lógica posee su propio símbolo especial:

A AND B	A.B
A OR B	A+B
NOT A	$\overline{A}$

Así como existen leyes que gobiernan el manejo de cifras en aritmética y de letras en álgebra, del mismo modo existen leyes especiales que gobiernan la simplificación de las expresiones lógicas. Las leyes del álgebra booleana están resumidas en la tabla siguiente.

RELACIONES ESPECIALES	
Relación	**Dual**
$A.A=A$	$A+A=A$
$A.\overline{A}=0$	$A+\overline{A}=1$
$A.0=0$	$A+1=1$
$A.1=A$	$A+0=A$
$A.(A+B)=A$	$A+A.B=A$
$A.(\overline{A}+B)=A.B$	$A+\overline{A}.B=A+B$
LEYES DE MORGAN	
1) $\overline{A+B}=\overline{A}.\overline{B}$	
2) $\overline{A.B}=\overline{A}+\overline{B}$	
PROPIEDAD ASOCIATIVA	
$A.(B.C)=(A.B).C=A.B.C$	
$A+(B+C)=(A+B)+C=A+B+C$	
PROPIEDAD CONMUTATIVA	
$A.B=B.A$	
$A+B=B+A$	
PROPIEDAD DISTRIBUTIVA	
$A.(B+C)=A.B+A.C$	

Utilizando estas reglas se puede simplificar expresiones lógicas y reducir el número de puertas que requiera el circuito final. Además del método algebraico, también hemos estudiado el uso de los diagramas de Karnaugh en la simplificación de circuitos lógicos (véase p. 572). Los diagramas de Karnaugh representan un avance significativo. Aunque no nos dispensan de las simplificaciones algebraicas, reducen la cantidad de esfuerzo inherente al tratar con álgebra booleana. Estos diagramas, que en realidad son una versión de los diagramas de Venn (véase p. 526), permiten el agrupamiento de expresiones sacadas de una tabla de verdad con dos, tres y hasta cuatro variables (o sea, cuatro, ocho y dieciséis recuadros). Estos grupos representan expresiones booleanas más sencillas, de modo que es posible la simplificación. En la práctica, con frecuencia se recurrirá a una mezcla de diagramas y de álgebra para producir el circuito más eficaz.

También hemos analizado el empleo de las operaciones lógicas AND y OR en la programación en BASIC. Vimos cómo se pueden utilizar estas órdenes para combinar condiciones en sentencias IF...THEN en BASIC, y cómo le permiten al programador activar y desactivar bits aislados dentro de un registro (véase p. 546).

Como punto final de esta primera etapa del curso, hemos empleado todos nuestros conocimientos relativos a tablas de verdad, diagramas de Karnaugh, álgebra booleana y notación de puertas lógicas para diseñar circuitos que proporcionen las salidas deseadas para ciertas tareas definidas (véase p. 586). El siguiente ejercicio de revisión cubre todos los aspectos que hemos abarcado hasta el momento. Una vez que usted adquiera seguridad en el manejo de estas reglas, podremos seguir adelante para abordar la teoría lógica más avanzada.

Ejercicio de repaso

1) Un grupo pop está seguro de que su disco sencillo será un éxito si produce un buen video para acompañar a la canción y (*and*) si su casa discográfica incluye un obsequio gratis con el disco o (*or*) si aparecen en el programa de música pop más famoso de la televisión. Dibuje una tabla de verdad que refleje todos los resultados posibles. Si cada contingencia es igualmente probable, ¿qué posibilidad tiene el grupo de tener un sencillo de éxito?

2) Una escuela está organizando un club de ajedrez y se han de distribuir las tarjetas de socio. El comité decide que el número de socios se debe restringir de modo que los miembros sean:

- **a)** Un miembro del personal;
- **b)** Un alumno de cuarto o quinto año que estudie matemáticas y ciencias;
- **c)** Un alumno de sexto curso que estudie matemáticas o ciencias.

Diseñe un distribuidor automático de tarjetas de socio que funcione pulsando los botones que describan mejor al posible socio. Estos botones son:

- **A** = Alumno de cuarto curso
- **B** = Alumno de quinto curso
- **C** = Alumno de sexto curso
- **D** = Miembro del personal
- **E** = Alumno que estudia matemáticas
- **F** = Alumno que estudia ciencias

Dibuje un diagrama del circuito para la máquina.

3) Un determinado microordenador posee un registro con la dirección 23148 (en decimal) que se utiliza para controlar la visualización en video. El bit 0 es el bit menos significativo del registro y el bit 7 es el más significativo. La pantalla se puede colocar en modalidad de alta resolución estableciendo los bits 4 y 5 en uno. Ya que los otros bits se utilizan para otras funciones, es importante no alterar sus valores. Escriba órdenes en BASIC que:

- **a)** Enciendan la pantalla en alta resolución
- **b)** La vuelvan a apagar

4) La cerradura de la caja fuerte de un banco se basa en interruptores operados mediante teclas. Tanto el gerente como su delegado y el cajero jefe poseen llaves. La puerta de acceso a la caja fuerte se puede abrir con dos de las tres llaves, pero:

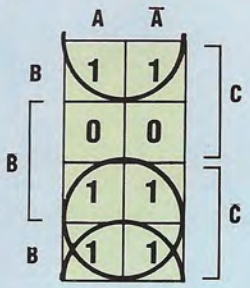
- **a)** Los días laborables, entre las 9 y las 17 horas, la caja fuerte sólo se puede abrir en presencia del cajero jefe.
- **b)** En todo otro momento la caja fuerte sólo la puede abrir el gerente conjuntamente con una de las otras dos personas.

Diseñe un circuito para controlar el sistema de conmutación.

5) En un sistema digital binario de transmisión de datos, cada bit de datos alimenta simultáneamente a tres canales en paralelo que, en el extremo de recepción, alimentan a su vez un "controlador". De no haberse producido ningún error de transmisión, entonces los tres bits deberían ser iguales. En condiciones defectuosas, el controlador corregirá un error en un canal, dando una salida simple que se corresponda con la mayoría de las entradas. Diseñe un circuito para este controlador.

Respuestas al ejercicio 5 de la página 587

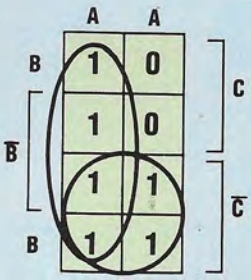
1a)



$= \overline{C} + \underline{B}$
(o bien, $\overline{C.\overline{B}}$)

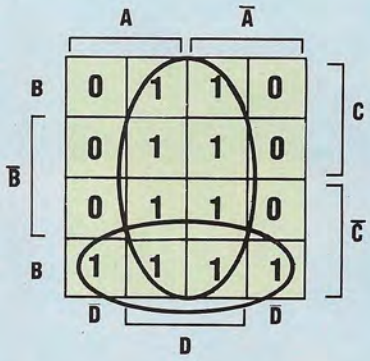
b)

$\overline{\underline{B} + \overline{C}} + B.\overline{C} + A.C$
$= \overline{B}.\overline{C} + B.\overline{C} + A.C$ (ley de Morgan)



$= A + \overline{C}$
(o bien, $\overline{A}.C$)

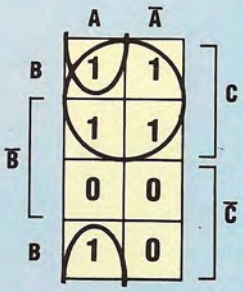
c)



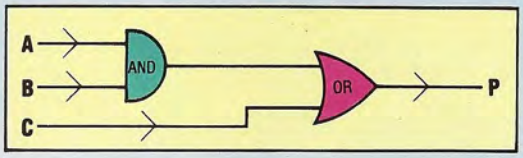
$= D + B.\overline{C}$

2) La tabla de verdad es:

Decimal	A	B	C	P
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



A partir del diagrama obtenemos la expresión $P = C + A.B$. El circuito resultante es:



Liz Dixon

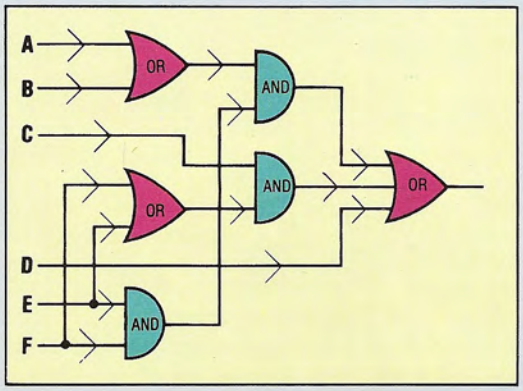
Respuestas al ejercicio de repaso

1)

Buen video	Obsequio	Aparecer TV	Éxito disco
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Probabilidad de éxito del disco = **5/8 = 62,5 %**

2)



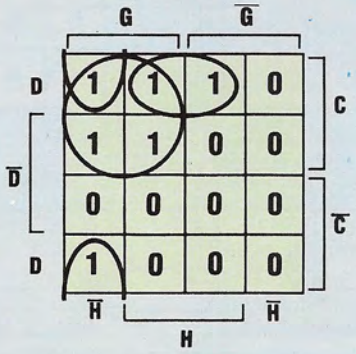
3a) Para encender alta resolución:
POKE23148,PEEK(23148)OR48

b) Para apagar alta resolución:
POKE23148,PEEK(23148)AND207

4) La tabla de verdad es:

Gerente	Delegado	Cajero jefe	9-17 horas	Abrir puerta
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

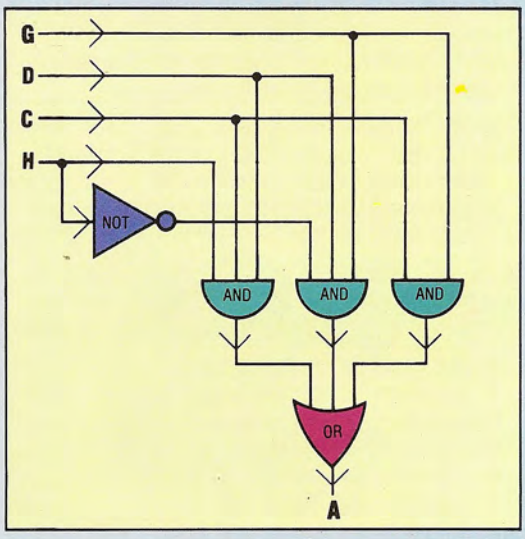
A partir del diagrama de Karnaugh:



Obtenemos la expresión: A=G.C+G.D.$\overline{H}$+ +D.C.H. Es fácil verificar si hemos resuelto correctamente esta etapa volviendo a convertir esta expresión al idioma corriente. La expresión dice: "La puerta la pueden abrir:

a) el gerente y el cajero en cualquier momento;
b) el gerente y su delegado fuera de horario;
c) el delegado y el cajero entre las 9 y las 17 h».

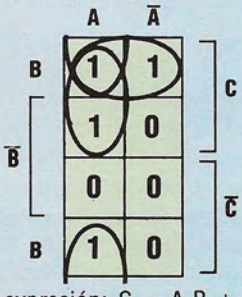
El correspondiente circuito asume la siguiente forma:



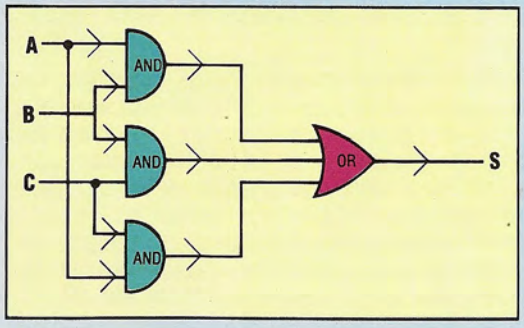
5) La tabla de verdad es:

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Con el diagrama de Karnaugh:



Obtenemos la expresión: S = A.B + A.C + B.C.
El circuito correspondiente es:



Liz Dixon

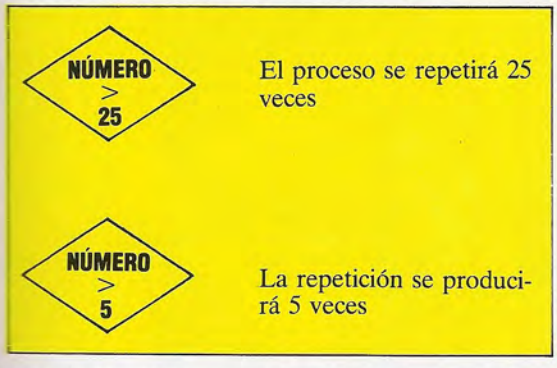
Bucles condicionados

Una noria sería un acertado símil para un bucle, cuyo movimiento tendremos que detener alguna vez

El bucle incondicionado, que vimos en el capítulo anterior, tiene unas aplicaciones limitadas. Supongamos que lo que se desea es mostrar en la pantalla la tabla de multiplicar de un número cualquiera, pero sin límite. El organigrama quedaría como en la figura 1. Como BASE se dará entrada al número cuya tabla de multiplicar se desea visualizar. El campo NÚMERO corresponderá a los diferentes valores por los cuales, en incrementos de uno, la BASE irá multiplicándose, dando así un resultado que se visualiza, y vuelta a empezar. No existirá un final hasta llegar al límite de capacidad numérica del ordenador o bien hasta que la persona que observa el desarrollo del programa adopte una decisión.

Los *bucles condicionados* se caracterizan porque su final depende siempre de una decisión, una pregunta que sirve de test para enviar la secuencia de nuevo a ejecutar la repetición o bien salir del ciclo. Retomando el ejemplo de la tabla de multiplicar, supongamos que se quieren visualizar los diez primeros elementos de la tabla. El programa sería el que indica la figura 2. La novedad estriba en la inclusión del rombo que marca una decisión. La pregunta sobre si el número, convenientemente incrementado tras el cálculo e impresión del resultado, ha superado el valor 10 tendrá dos posibles salidas. Si no se cumple dicha condición, es decir, si el número todavía está dentro de los 10 primeros números, el flujo retorna al principio del ciclo, mientras que en el caso contrario la secuencia derivará hacia el final del organigrama.

Es posible en cualquier momento aumentar o disminuir el intervalo, o sea el número de veces que el ciclo va a ser repetido, solamente con variar el valor incluido en el símbolo decisorio.



Un mismo programa puede contener un número no determinado de bucles, inconexos o no entre sí. En el próximo capítulo estudiaremos los denominados bucles anidados.

Tabla de multiplicar hasta el infinito

Figura 1

Tabla de multiplicar hasta 10

Figura 2

La evolución del Dragon

El Dragon 64 es una versión mejorada del 32, que ofrece amplias posibilidades para un serio software de gestión

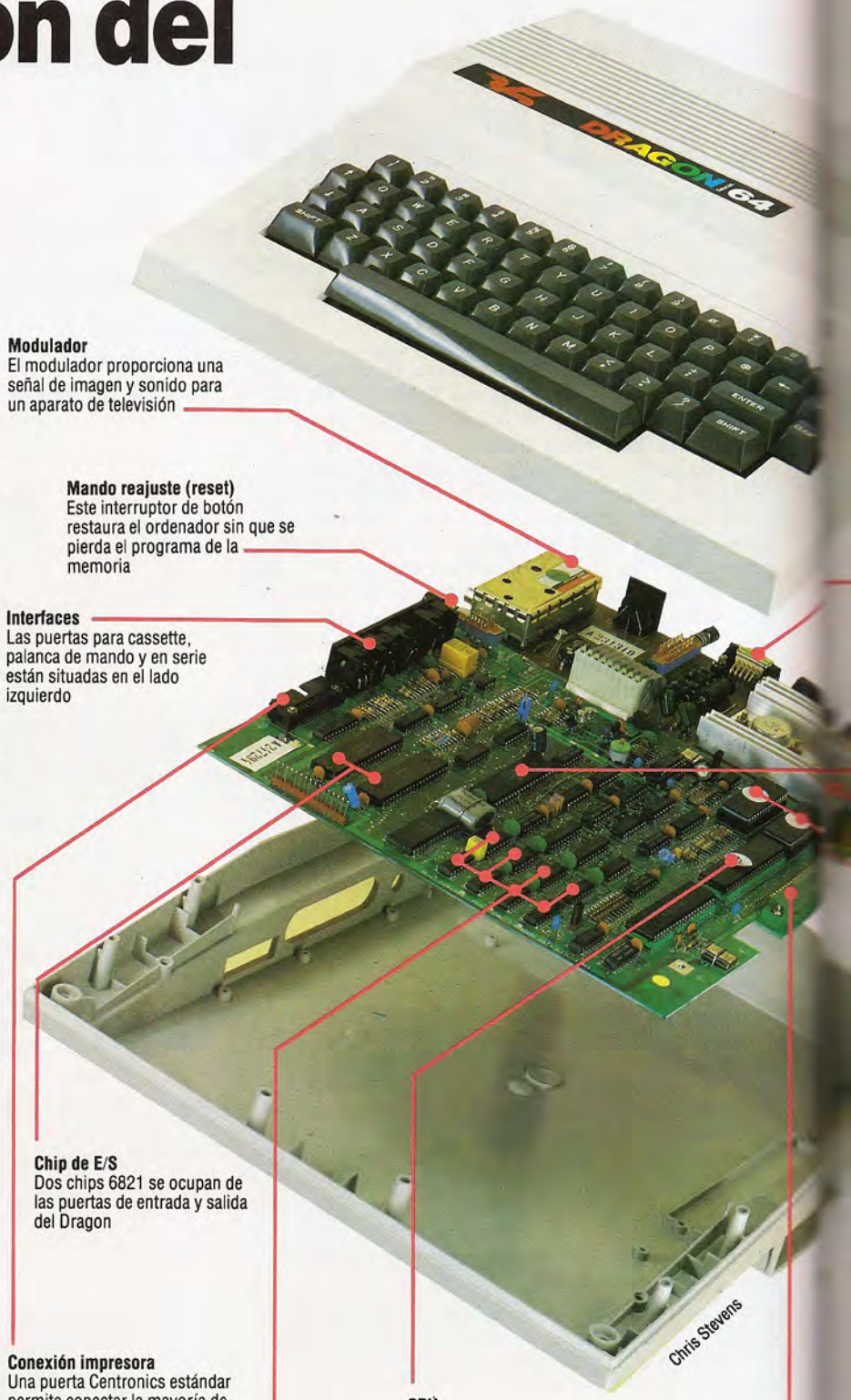
El Dragon 64 posee el mismo teclado que los últimos Dragon 32, si bien es de mejor calidad. Pero carece de muchas teclas útiles, como Escape, Tab y Control.

Al encenderlo por primera vez, el Dragon 64 (al igual que el 32) estará automáticamente en la "modalidad 32", en la que puede ejecutar todo el software de Dragon. Los usuarios de BASIC disponen de hasta 30 Kbytes de memoria libre. La orden EXEC 48000 apaga la ROM de BASIC y enciende en la parte superior 32 Kbytes de memoria libre. El BASIC se copia entonces en la memoria de arriba, dejando hasta 45 Kbytes libres. Los 64 Kbytes completos sólo están disponibles bajo los otros sistemas operativos o programas en código de lenguaje máquina. El DOS de Dragon restaurará el sistema a la modalidad 32, dejando sólo alrededor de 23 Kbytes libres para los programas en BASIC.

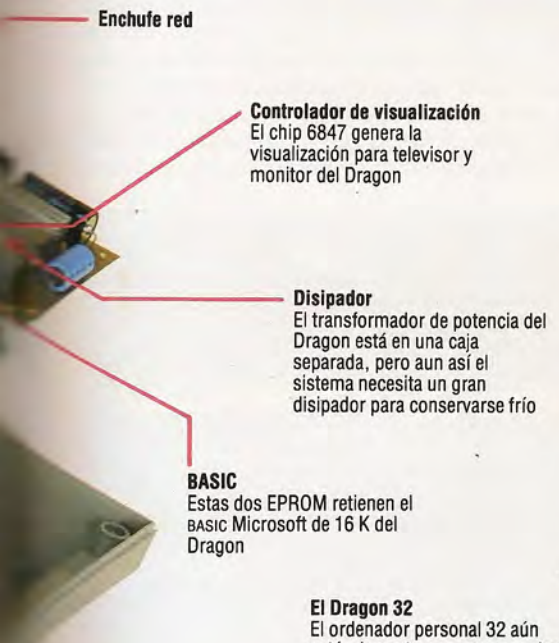
Sorprendentemente, el tablero de circuitos impresos del 64 es muy diferente del tablero del 32. Se trata del mismo ordenador, pero se han desplazado los chips con el objeto de dejar sitio a la nueva puerta en serie y la memoria extra. A consecuencia de ello, los usuarios del 32 que deseen mejorar sus máquinas habrán de reemplazarlas por el modelo 64, ya que no podrán ampliar sus modelos 32 enchufando componentes extras. La CPU del 64 es la Motorola 6809, un diseño de ocho bits que ha llegado demasiado tarde para alcanzar la popularidad de la 6502 y la Z80. Un chip compañero, el 6847, genera la visualización ya sea en un aparato de televisión o en un monitor de video compuesto. Es este chip el que le confiere al Dragon sus más bien curiosas modalidades de visualización.

El Dragon cuenta con una pantalla para textos de 32 × 16 caracteres sobre un fondo verde o naranja. Esta visualización limitada, si bien no constituye un problema para el usuario del Dragon 32, tiene serias implicaciones para quienes deseen utilizar el Dragon 64 como máquina de oficina. El nuevo software basado en disco, en particular el sistema operativo OS9 profesional, se ve muy limitado por la reducida superficie de visualización.

Para gráficos detallados el Dragon posee modalidades en alta resolución, que van desde una resolución de 128 × 96 en cuatro colores hasta un máximo de 256 × 192 en dos colores. En comparación con muchas otras máquinas, esto es muy limitado, especialmente porque la calidad de la visualización puede ser pobre. Sin embargo, le permite al Dra-



La unidad de disco Dragon
La unidad de disco Dragon posee una o dos unidades de 175 K cada una. Viene con el DOS de Dragon, pequeño pero adecuado, incorporado en su cartucho de interface



Enchufe red

Controlador de visualización
El chip 6847 genera la visualización para televisor y monitor del Dragon

Disipador
El transformador de potencia del Dragon está en una caja separada, pero aun así el sistema necesita un gran disipador para conservarse frío

BASIC
Estas dos EPROM retienen el BASIC Microsoft de 16 K del Dragon

El Dragon 32
El ordenador personal 32 aún está a la venta y es muy popular. Pero muchos propietarios están mejorando sus máquinas, para equipararlas con el poder y la flexibilidad del 64

Ian McKinnell

gon ejecutar juegos de estilo recreativo y se puede utilizar con el software escrito especialmente para visualizar mayúsculas y minúsculas en 51 columnas de 24 filas, útil para programas de hoja electrónica y de tratamiento de textos.

Almacenar y ampliar

El Dragon posee una buena versión de BASIC Microsoft. Es una versión ampliada de 16 Kbytes y entre sus facilidades figura un conjunto de eficaces órdenes para gráficos y sonido. El Dragon puede retener simultáneamente en la memoria un gran número de imágenes de gráficos en pantalla y puede conmutar entre ellas de forma instantánea, proporcionando una forma sencilla de realizar animación de gráficos en BASIC. El usuario también puede optar por emplear el espacio de memoria normalmente reservado para gráficos para los programas y los datos en BASIC.

La máquina posee un conjunto completo de interfaces, incluyendo dos conversores de analógico a digital y una puerta Centronics estándar para conectar la mayor parte de impresoras en paralelo. La nueva puerta en serie puede conectar a impresoras (incluyendo impresoras margarita) o a otros ordenadores y equipos. Una forma interesante de desarrollar el sistema sería conectarlo a un terminal profesional, lo que permitiría la utilización de un software más avanzado.

El Dragon se conecta en interface a grabadoras de cassette estándar y puede hacer arrancar y detener la cinta desde dentro del programa, y reproducir sonido desde la cinta a través del altavoz del televisor. El BASIC también admite una gama de órdenes para manipulación de archivos en cassette. Ya hemos analizado las unidades de disco del Dragon (véase p. 584). Vienen con un efectivo sistema operativo en ROM en el cartucho de la interface, y con una serie de ampliaciones al BASIC.

El Dragon 64 no se podrá ampliar para convertirlo en un serio sistema de gestión hasta que no posea una adecuada visualización en pantalla y un teclado profesional. Ahora se trata de una máquina muy capaz e interesante para aficionados.

El OS9 multitarea-multiusuario
Su microprocesador 6809, sus 64 Kbytes de memoria y sus unidades de disco le permiten al Dragon 64 ejecutar el sistema operativo OS9 profesional. Es éste el principal sistema operativo para ordenadores basados en el 6809, y proporciona algunas facilidades excelentes así como el acceso a una gama de sofisticados paquetes de gestión. Dragon ha dedicado considerables esfuerzos a lograr que el 64 dispusiera del OS9. Pero el elevado costo del OS9 y de su software podría limitar su popularidad entre los usuarios de ordenadores personales.
El OS9 proporciona las facilidades estándar asociadas con el UNIX, un sistema operativo para miniordenadores de que disponen los micros de gestión más grandes. Entre sus configuraciones se incluye la capacidad para que un usuario ejecute más de un programa a la vez (multitarea) y para que más de una persona utilice el ordenador al mismo tiempo (multiusuario). Para conseguir estas capacidades, el OS9 organiza los archivos de programas y de información en una estructura formal, en vez de limitarse a hacerlo en un directorio simple de archivos en cada disco. Cada archivo posee también contraseñas y códigos de acceso, de modo tal que a ciertos programas y a cierta información sólo pueden acceder determinadas personas.
La mayoría de los usuarios de ordenadores personales no explotan estas facilidades y el Dragon debería soportar una tensión considerable si se utilizara el OS9 al completo. No obstante, proporciona una forma económica de experimentar con un sistema sofisticado, además de dar acceso a programas avanzados y a lenguajes como el c y el PASCAL

DRAGON 64

DIMENSIONES
380 x 330 x 90 mm

CPU
6809

MEMORIA
64 K de RAM, de los cuales hasta 45 K están disponibles para programas en BASIC.
16 K de ROM

PANTALLA
En la modalidad para textos, 16 filas de 32 columnas, mayúsculas sólo con un juego de formas para gráficos en ocho colores. Modalidades para gráficos desde 128 x 96 en cuatro colores, hasta 256 x 192 en dos colores

INTERFACES
Palancas de mando (2), puerta en serie, puerta para impresora en paralelo, puerta para cassette, monitor compuesto con sonido, puerta para TV y ampliación con cartucho

UNIDADES DE DISCO
Hasta dos de 175 K, con el DOS de Dragon o bien con el sistema operativo OS9

LENGUAJES DISPONIBLES
BASIC, FORTH, lenguaje ensamblador 6809. El OS9 proporciona C, PASCAL, BASIC estructurado, así como otros lenguajes

TECLADO
Tipo máquina de escribir con 53 teclas

DOCUMENTACION
Lamentablemente, los manuales del Dragon contienen sólo la información más elemental y, por otra parte, adolecen de omisiones y errores

VENTAJAS
El 64 posee las configuraciones de un ordenador sofisticado (una gran memoria y una gama completa de interfaces). El sistema de disco constituye un elemento valioso, y la capacidad de ejecutar software OS9 una propiedad de primer orden, aunque su precio es relativamente alto

DESVENTAJAS
Sufre de limitaciones en cuanto al teclado y las modalidades de visualización en pantalla, especialmente cuando se utiliza para tareas serias. La CPU 6809 de 1 MHz podría resultar demasiado lenta para algunas aplicaciones determinadas, basadas en el OS9

Escaparate de color

Con un adecuado software para gráficos es posible crear páginas de videotex y hacerlas desfilar cíclicamente por la pantalla

Además del Prestel, en Gran Bretaña existen otros dos sistemas públicos de videotex: Ceefax y Oracle, aunque dado que se transmiten a través de las ondas y que no se envían mediante un cable, se los denomina estrictamente sistemas de *teletexto*. Muchas grandes empresas han adoptado sus propios sistemas de videotex "de uso doméstico", que mantienen al cuadro directivo informado con índices y sucesos de la empresa y les permite acceder a información especializada (como las últimas cotizaciones bursátiles, p. ej.). Siguiendo la misma tendencia, algunos municipios británicos han instalado un sistema de videotex local y gratuito, que contiene la información allí denominada *What's on* (espectáculos y efemérides), detalles relativos a los servicios municipales, etc.

En los centros comerciales urbanos el videotex también se está convirtiendo en algo común. Tal vez un agente de viajes haya hecho las reservas de vacaciones a un cliente mediante el Prestel, o quizá en la tienda de alquiler de películas de video haya un sistema de videotex en el escaparate, que va mostrando una serie de páginas de publicidad, según el sistema que se conoce como carrusel.

En todas estas áreas se puede instalar una fuente informativa. Esto significa que uno puede crear páginas de información en videotex y después utilizarlas en un sistema privado de videotexto propio o venderlas a los clientes. Por ejemplo, en Gran Bretaña colocar una página individual en el sistema Prestel vale entre 15 y 45 libras, pero después de colocar la información se suele cargar a los usuarios del Prestel cada vez que leen dicha página (normalmente entre uno y 10 peniques por cada vez que se accede a ella). A juzgar por la cantidad de fuentes informativas posibles en el Prestel, resulta evidente

Creación de páginas
Vamos a seguir uno a uno los pasos necesarios para crear una página de videotex.
Supongamos que una agencia de viajes local le ha encargado que cree un llamativo anuncio ofreciendo unas vacaciones en Grecia, el cual, junto a otras páginas, se colocará en un sistema carrusel en el escaparate de la agencia.
El primer paso, antes de que toquemos siquiera el teclado, consiste en dibujar en un papel un boceto esquemático de la página. Para ser más exactos, podríamos trazar primero una cuadrícula de 40 por 24 y luego trasladar cada cuadrado del papel a un carácter de pantalla. Dibujaríamos una escena típica de vacaciones (es decir, playa, mar, sol...) y a continuación superpondríamos los detalles de precio, lugar, etc.

Página 100
Aquí hemos dibujado el mar y la playa. El mar se produce pulsando la tecla de función azul seguida por la tecla de fondo al comienzo de cada línea. Varias líneas de fondo amarillo componen la playa

Página 101
Ahora empezamos a "construir" la colina olímpica con caracteres mosaico verdes superpuestos sobre el mar. De forma similar, se han realizado los primeros indicios de una sombrilla con mosaicos color magenta

Página 102
La imagen toma forma: se ha terminado la colina, a la sombrilla se le han agregado franjas rojas y se ha comenzado a esbozar el sol. Cada una de las letras grandes se compone de seis caracteres mosaico

Página 103
La imagen ya está casi acabada. Las letras grandes, el sol y la sombrilla ya están completos y se ha agregado un barco de vela. Para dar un toque de autenticidad, ¡no falta ni siquiera el templo griego mirando al mar!

Página 104
La página al completo. El toque final consistió en la adición del texto digitado sobre la imagen. La leyenda "TWO WEEKS IN GREECE" (dos semanas en Grecia) se hizo con caracteres de doble altura, y si no se tratara de una foto fija podríamos ver que en realidad el precio es intermitente. Es de advertir que las "olas" que hemos agregado al mar son signos menos (−) de color cyan

Ian McKinnell

que se puede ganar mucho dinero, ya sea de manera directa a través de ese cargo por acceder a la página o bien indirectamente a través de la publicidad conseguida a partir de la misma.

Un emprendedor grupo de estudiantes de Gales del Sur se pusieron en contacto con unos grandes almacenes durante la época de Navidad y se ofrecieron para instalar un sistema de videotex local que visualizara para los clientes páginas de anuncios. Crearon las páginas utilizando un sencillo paquete editor de videotex con su micro y después instalaron el ordenador en el escaparate principal de la tienda, donde automáticamente iban pasando una y otra vez las diversas páginas. ¡Cobrando de la tienda las páginas y el alquiler del micro se ganaron una buena paga extra de Navidad!

Éstos son apenas algunos ejemplos de las aplicaciones que puede tener el videotex, pero a cualquiera que tenga un poco de imaginación seguramente se le ocurrirán otros muchos. Todo cuanto se necesita es un microordenador, un poco de talento artístico y, por supuesto, el software.

El paquete de videotex específico que vamos a analizar aquí se denomina *Viewtext* y funciona en el BBC Micro, ordenador que se destaca sobre la mayoría de los otros micros en virtud de su juego de caracteres de videotex incorporado. Aun siendo este paquete uno de los más baratos, incorpora tanto un editor de videotex para crear las páginas como un programa para establecer un sistema carrusel de 22 páginas.

El editor trabaja de forma muy similar a cualquier otro sistema de edición basado en pantalla: el cursor se posiciona mediante las teclas de control del cursor y después se digita el texto. Para seleccionar las diversas características del videotex (colores para el texto y para el fondo, caracteres intermitentes, caracteres de altura doble, etc.), tan sólo debe pulsar las teclas de función rojas de la fila superior del teclado del BBC. Por ejemplo, para conseguir que la palabra "HOLA" aparezca intermitente y en letras amarillas, se pulsan las teclas f3 (para amarillo) y f8 (para intermitente), y después se digita la palabra.

Habiendo llegado a este punto, es importante mencionar dos detalles respecto al funcionamiento de un editor de videotex. En primer lugar, las propiedades como "intermitente" o "texto rojo" se pierden cuando se pasa a la línea siguiente: el texto vuelve a las letras blancas y pequeñas sobre fondo negro. En segundo lugar, cada característica ocupa un espacio de carácter invisible en la pantalla. Por lo tanto, en el ejemplo anterior habría dos espacios aparentemente en blanco delante de la palabra "HOLA". Si usted borra estos espacios o si escribe sobre ellos, entonces las propiedades correspondientes se pierden.

Gráficos de videotex

Lo que hemos dicho hasta ahora nos permite crear páginas que sólo contengan texto, lo cual resulta adecuado para muchas aplicaciones de videotex. Sin embargo, los diagramas, el material publicitario y cosas de este tipo en realidad exigen no sólo la utilización de texto sino también de gráficos. Los gráficos de videotex se crean a partir de un juego de caracteres para gráficos, constando cada uno de los caracteres hasta el seis pixels en una cuadrícula

El editor Viewtext
El paquete Viewtext incorpora un sencillo editor para que el usuario pueda diseñar sus propias páginas de teletexto. En este ejemplo, el anuncio de las vacaciones en Grecia se ha adaptado modificando el texto y la imagen

de dos columnas por tres filas. Las imágenes se construyen de forma similar a como se construye un mosaico mediante ínfimas teselas y por esta razón a los caracteres se los denomina *mosaicos alfa*. La calidad de la imagen resultante deja bastante que desear si se compara con las capacidades para gráficos de la mayoría de los ordenadores personales, pero con un poco de imaginación cualquier imagen se puede realizar con mosaicos alfa.

A este propósito, añadiremos que la razón por la cual los gráficos son tan elementales es que a cada página de videotex sólo se le concede un Kbyte de memoria, mientras que una pantalla para gráficos de alta resolución de un ordenador personal puede utilizar alrededor de 20 Kbytes. Dada la velocidad baudio a la que funciona el Prestel, el tiempo que se tardaría en recibir una página completa en alta resolución sería excesivo.

La utilización de los caracteres mosaicos alfa es un área en la que el editor *Viewtext* no se desenvuelve bien. Los caracteres se obtienen pulsando primero la tecla de función roja (f9) para encender los gráficos, y pulsando después cualquiera de las teclas de minúsculas o de números del teclado. Cada tecla produce un carácter para gráficos, pero lamentablemente no existe relación alguna entre lo que está grabado en cada tecla y el carácter que aparece en la pantalla: uno debe remitirse continuamente a una tabla del manual para encontrar la tecla adecuada. Este proceso es muy pesado: la creación de los gráficos mosaico en las fotografías que acompañan este artículo requirió un par de horas. Los editores más sofisticados permiten que uno construya cada carácter encendiendo o apagando cada uno de los seis pixels de la matriz del carácter, método éste mucho más rápido.

Hay dos configuraciones especiales que sólo se utilizan con los caracteres para gráficos de mosaico: *Separate graphics* (separar gráficos) hace saltar cada carácter de modo que los pixels queden ligeramente separados los unos de los otros; *Hold graphics* (retener gráficos) se utiliza para cubrir los espacios en blanco que quedan cuando se establece una propiedad nueva. Por ejemplo, para obtener un carácter de mosaico rojo junto a otro verde sin que quede un espacio en blanco entre ellos, se debe pulsar la tecla de función *Hold graphics* antes de pulsar la tecla *Green graphics* (gráficos verdes).

Con esto cubrimos la mayoría de las configuraciones de que disponen todos los editores de videotex; las versiones más caras poseen facilidades para acelerar el proceso de creación de páginas, que re-

Videotex de alta resolución

Ian McKinnell

La mayoría de los sistemas de videotex utilizan gráficos de mosaicos alfa, que son económicos en cuanto a memoria pero menos satisfactorios en cuanto a la calidad de las imágenes obtenidas. El estándar alternativo para gráficos con videotex se denomina alfageometría y es capaz de producir gráficos mucho más realistas.

Además de las letras, los números y los caracteres mosaico usuales, los gráficos alfageométricos permiten dibujar líneas, círculos y formas similares utilizando órdenes simples, de forma muy parecida a las órdenes en BASIC para gráficos de alta resolución de que disponen la mayoría de los micros personales. Para acomodar la resolución más alta, la pantalla se divide en 320 × 240 pixels, frente a los 40 × 24 cuadrados que se utilizan en los gráficos de mosaicos alfa. Esto tiene el inconveniente de que transmitir a través de las líneas telefónicas una imagen muy detallada (p. ej., la del rostro de una persona) ocupa un par de minutos.

Uno de los sistemas comerciales disponibles en el mercado británico es el terminal MUPID. Se trata de un microordenador exclusivo y es capaz de visualizar tanto gráficos de mosaicos alfa como alfageométricos. Entre sus configuraciones más novedosas está la capacidad de producir animación en una página individual, así como colores de tonalidad intermedia y sombreado.

El sistema alfageométrico es excelente para presentar imágenes detalladas, como mapas o facsímiles.

sultan útiles en el campo comercial pero que no son necesarias para el usuario personal. Por ejemplo, un paquete más caro le permitiría desplazar o copiar partes de la página de videotex y contaría con un método más rápido para crear los gráficos de mosaico que los descritos anteriormente.

Después de que haya creado su página de videotex, el paso siguiente consiste en guardarla en disco o en cinta, otorgarle un número de página y colocarla en su propio sistema de videotex. El paquete *Viewtext* puede retener hasta 22 páginas en la memoria del BBC en cualquier momento, y a cada página se le da un número entre 100 y 121. Luego que haya creado y guardado suficientes páginas, puede ejecutar el programa carrusel. Éste carga todas las páginas y las hace desfilar en un ciclo repetitivo, siendo el usuario quien decide la duración del intervalo entre página y página. En la línea superior de todas las páginas de videotex aparece información como la fecha y la hora, el número de la página que se está contemplando y el nombre del servicio de videotex (Prestel, etc.). El programa carrusel pide al usuario esta información antes de ir pasando las páginas cíclicamente.

Hay paquetes de videotex más refinados que permiten transmitir y recibir páginas de otros sistemas, utilizando un modem y una línea telefónica. Estos paquetes son caros y exigen gran cantidad de almacenamiento en disco para las muchas páginas de información, de modo que podrían no estar al alcance de la mayoría de los usuarios de ordenadores personales. Ejecutando un sencillo programa para edición en un micro personal, se encuentra usted en disposición de crear páginas de videotex capaces de competir con las que se obtienen de cualquier agencia comercial.

Espacio de memoria

Debemos analizar dónde van a parar los programas dentro de la memoria, y ver cómo se lleva a cabo una operación aritmética simple

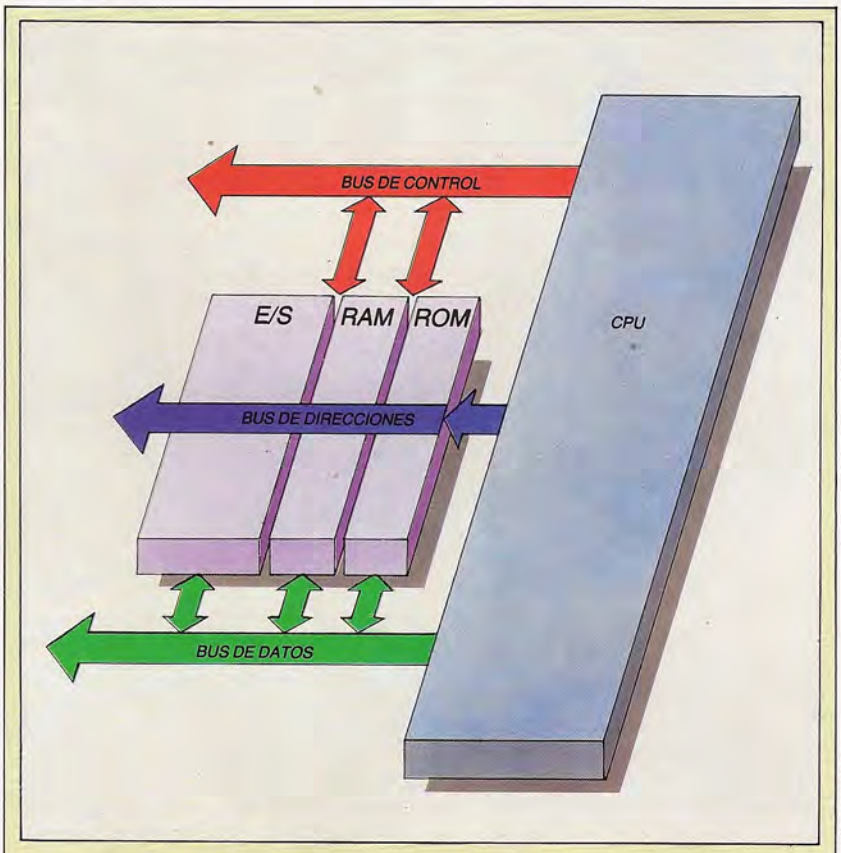
En la última lección de este curso de lenguaje máquina elaboramos un programa muy sencillo en lenguaje ensamblador, lo tradujimos (ensamblamos) a lenguaje máquina, lo cargamos en la memoria y lo ejecutamos. Para las dos últimas tareas utilizamos el programa monitor de la página 598. Si el programa fuera un paquete más sofisticado que contuviera un ensamblador, podríamos haberlo empleado también para ensamblar nuestro programa en lenguaje máquina. En esta etapa del curso, hacer el ensamblaje a mano no supone ningún esfuerzo excesivo; en realidad, constituye un ejercicio muy educativo. Pero una vez que usted haya asimilado los principios del proceso, y a medida que sus programas en lenguaje ensamblador se vayan haciendo más largos, no tendrá ningún sentido que se preocupe por la verdadera traducción a lenguaje máquina. Por consiguiente, cuando llegue a esta etapa del aprendizaje del lenguaje máquina le interesará adquirir un programa ensamblador adecuado para su ordenador.

Son muchos los puntos a resaltar dentro del corto programa en lenguaje máquina que le proporcionamos (véase p. 597). Nosotros utilizamos uno de los registros de la CPU para manipular la memoria, tuvimos que decidir en qué lugar de la memoria íbamos a almacenar el lenguaje máquina e hicimos que el microprocesador lo ejecutara. Todos éstos son aspectos de la programación en lenguaje ensamblador que confunden al principiante y deben analizarse con detención. Empecemos por la cuestión de dónde se almacena el lenguaje máquina.

Para la CPU la única diferencia entre un byte de memoria y el siguiente es el que sea de memoria de acceso directo (RAM) o de memoria de lectura solamente (ROM). Los chips de ROM contienen programas y datos del sistema que se deben proteger contra una sobreescritura accidental o deliberada y, por consiguiente, sólo se pueden leer. En ellos no se puede escribir, de modo que no es posible cargar un programa de lenguaje máquina en la ROM. A excepción de estas áreas de la memoria, teóricamente nada hay que nos impida cargar un programa en cualquier otra parte de la misma, pero existen ciertas consideraciones prácticas que nos prohíben utilizar determinadas áreas.

La CPU emplea ciertas secciones de la RAM para almacenamiento temporal en el curso de sus operaciones, y, si cargamos un programa allí, simplemente se destruirá porque la CPU escribe sobre ella o bien (y esto es lo más probable) la CPU leerá nuestro lenguaje máquina como si se tratara de algunos de sus propios datos. El sistema operativo también utiliza grandes porciones de la RAM para almacenar sus datos de trabajo y para ejecutar el sistema del ordenador. Cargar programas en código de lenguaje máquina en cualquiera de estas zonas sería imprudente o imposible, por las mismas razones que nos prohíben emplear el espacio de trabajo de la CPU. Además, los programas en BASIC pueden ocupar toda la RAM restante (en parte como texto de programas y en parte como zonas de almacenamiento de variables). Una vez más, sería un error tocar estas zonas y por ello el programador se enfrenta con un complejo dilema.

Una respuesta es no utilizar el BASIC para nada, liberando zonas como el área para textos de programas en BASIC. Pero por lo general es conveniente



Kevin Jones

La arquitectura de un pequeño sistema

Un sistema típico de ordenador, en su forma más esquemática, comprende la memoria y una CPU. La primera se compone de chips de ROM (que contienen programas del sistema), chips de RAM y chips especializados exclusivos para operaciones de entrada-salida. Los datos y las señales de control fluyen hacia y desde la CPU y alrededor del sistema mediante *buses*. Los buses son vías (muy similares a cables planos) que pueden transportar un byte o más de datos por vez. Pueden ser unidireccionales, como el *bus de datos*, que puede transmitir en cualquier sentido. El *bus de control* transporta información de conmutación alrededor del sistema, abriendo y cerrando puertas lógicas para dirigir el flujo de datos. El *bus de direcciones* transporta una dirección de 16 bits desde la CPU para seleccionar un byte de memoria, permitiendo que los datos fluyan a lo largo del *bus de datos*, de ocho bits, hacia o desde el byte

te escribir al menos parte de un programa en ese lenguaje, utilizando subrutinas en lenguaje máquina sólo allí donde el BASIC sea demasiado lento, como por ejemplo, en las visualizaciones animadas sobre pantalla. Si los programas en BASIC y en lenguaje máquina han de coexistir en la RAM, entonces debemos reservar algo de espacio y destinarlo para lenguaje máquina. Podemos desplazar las fronteras del área para textos de programas en BASIC, o podemos encontrar secciones de la misma o de la RAM del sistema operativo que estén temporalmente sin utilizar.

Desplazar las fronteras del BASIC es muy fácil en el BBC Micro, dado que estas direcciones están retenidas en las variables del sistema PAGE, TOP, LOMEM y HIMEM. PAGE, por ejemplo, señala el comienzo del área para textos de programas en BASIC, que suele estar en la dirección $1200. Si ejecutáramos la instrucción:

```
PAGE = PAGE + 500
```

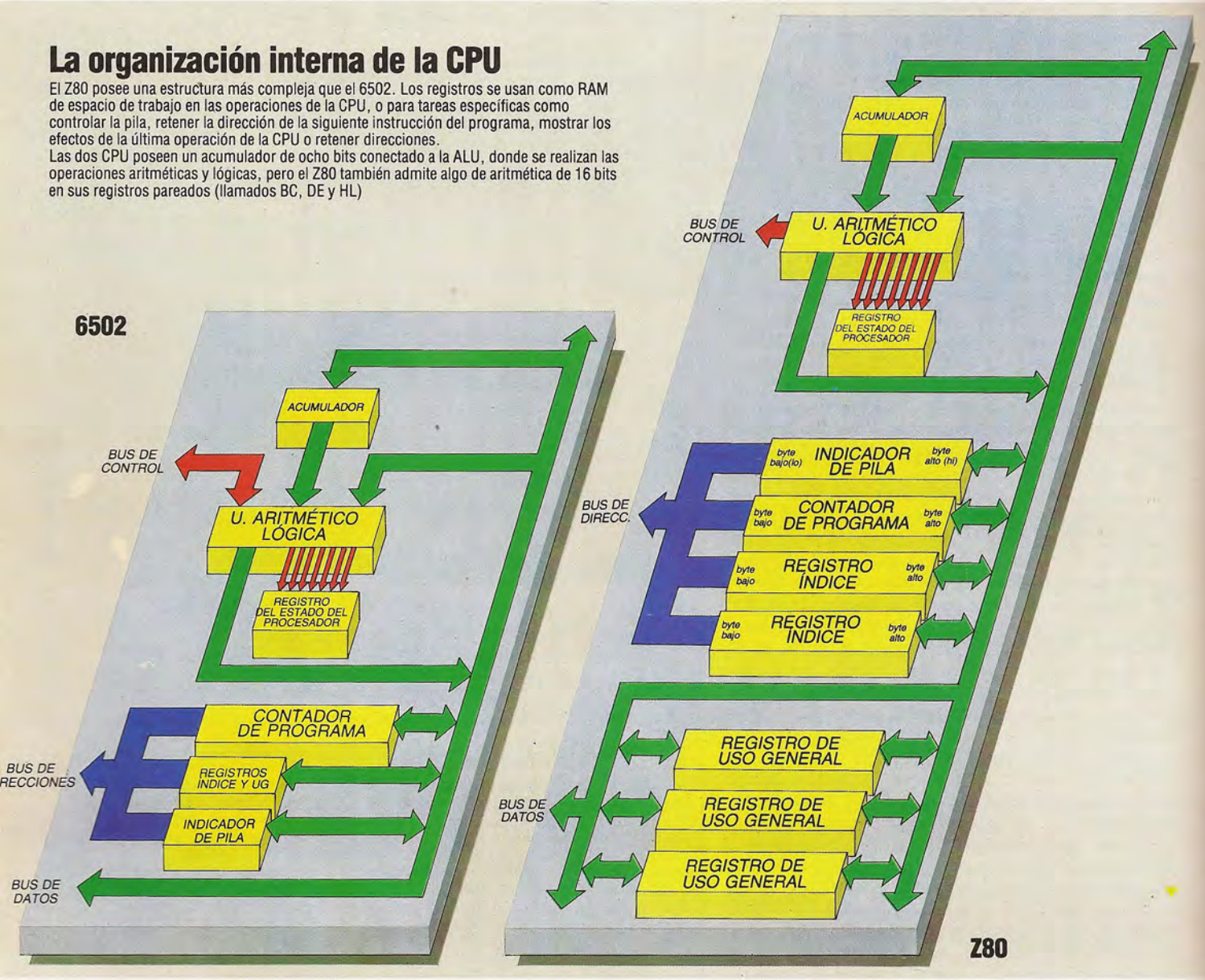
entonces el sistema operativo almacenará los programas en BASIC 500 bytes más arriba de la memoria, dejando libre una zona de 500 bytes para nuestros programas en lenguaje máquina. En otros ordenadores se consigue el mismo efecto colocando (POKE) direcciones más altas en los señaladores del sistema (véanse los mapas de memoria de p. 538). Se puede también reservar espacio bajando la dirección del extremo superior del área para textos de programas en BASIC. En el Spectrum, la orden CLEAR seguida de una dirección realiza exactamente eso. La única restricción en cuanto a la cantidad de memoria que se puede reservar mediante este método es que quede espacio suficiente para nuestro programa en BASIC.

En la RAM del sistema operativo se pueden encontrar pequeños bloques de espacio libre; un ejemplo típico de ello es el buffer (almacenamiento intermedio) para cassette del Commodore 64. Éste se compone de 192 bytes de RAM (desde $033C hasta $03FB) y el sistema operativo sólo lo utiliza cuando se emplea la unidad de cassette. A muchos programadores este espacio les basta para satisfacer todas sus necesidades de lenguaje máquina.

Dentro de programas en BASIC se pueden encon-

La organización interna de la CPU

El Z80 posee una estructura más compleja que el 6502. Los registros se usan como RAM de espacio de trabajo en las operaciones de la CPU, o para tareas específicas como controlar la pila, retener la dirección de la siguiente instrucción del programa, mostrar los efectos de la última operación de la CPU o retener direcciones.
Las dos CPU poseen un acumulador de ocho bits conectado a la ALU, donde se realizan las operaciones aritméticas y lógicas, pero el Z80 también admite algo de aritmética de 16 bits en sus registros pareados (llamados BC, DE y HL)



trar bloques de espacio aún más pequeños: las líneas REM, por ejemplo. En el área para textos de programas en BASIC, esta línea:

```
10 REM**************************
```

tiene 25 bytes consecutivos con el mismo contenido $2A, el código ASCII para '*' (decimal, 42). Ni el sistema operativo ni el intérprete de BASIC inspeccionan jamás estos bytes, porque para ellos la orden REM significa "prescinda del resto de esta línea". Una vez que se le ha dado entrada a la línea en el programa de esta forma, se puede cargar una subrutina en lenguaje máquina en los bytes de asteriscos, donde residirá sin que el intérprete la moleste en lo más mínimo. La gran ventaja que ofrece este método rebuscado en apariencia (pero que con frecuencia se utiliza en programas para el ZX81 ampliado) es que cuando el usuario guarda (SAVE) y subsiguientemente carga (LOAD) el programa en BASIC, la subrutina en lenguaje máquina lo acompaña. La utilización de los otros métodos descritos por lo general significa tener que guardar el lenguaje máquina por separado respecto del programa en BASIC. El problema de este método es que, al LISTar las líneas, el sistema operativo interpreta el byte del lenguaje máquina como datos de caracteres ASCII, lo que produce extrañas visualizaciones en la pantalla. Esto explica por qué había advertencias incluidas en los programas de demostración de la página 499 para el BBC Micro y el Spectrum. La versión de aquel programa para el Commodore 64 carga la subrutina en lenguaje máquina en el buffer de la cassette, mientras que las versiones para el BBC y el Spectrum la cargan en su línea REM de comienzo, y de allí las advertencias de no LISTar estas versiones del programa.

Después de haber escrito un programa en lenguaje ensamblador, de haberlo traducido a lenguaje máquina y de cargarlo (LOAD) en la RAM de su elección, puede proceder a ejecutarlo. Esto se efectúa en BASIC mediante las órdenes CALL (sólo el BBC), SYS (sólo el Commodore 64) o USR (las tres máquinas). Cada una de estas órdenes va seguida de la dirección del primer byte del programa en lenguaje máquina, donde sea que se almacene. Estas tres órdenes significan lo mismo para el intérprete: "Ejecute el programa en lenguaje máquina que comienza en la dirección dada, y retorne a ejecutar la siguiente instrucción en BASIC cuando se ejecute el opcode RET o RTS". Es como GOSUB en BASIC.

En la última lección escribimos un programa para copiar el contenido de un byte en otro byte, cargando en el acumulador el contenido de una dirección y almacenando luego el contenido del acumulador en la otra dirección. Lo que demuestra el papel central que desempeña la CPU en todo el sistema: los datos y el control deben fluir desde la memoria, siempre a través de la CPU, y regresar por este medio a la memoria. Mientras que en BASIC podemos escribir LET X = Y (que significa "copiar el contenido de Y directamente en X"), en lenguaje ensamblador tenemos que copiar de la memoria a la CPU, y de ésta devolver la copia a la memoria. Los registros de la CPU (véase el cuadro adjunto) son los bytes de RAM, dentro de la propia CPU, donde se almacenan o se manipulan los datos de la memoria. Tanto el Z80 como el 6502 poseen un registro llamado *acumulador*, al que se orientan muchas de las instrucciones en lenguaje ensamblador, y que es el registro en el que se efectúan principalmente las operaciones aritméticas.

Supongamos que deseamos sumar $42 y $07 (el símbolo $ significa hexadecimal). Colocamos uno de ellos en el acumulador y agregamos el otro encima del primero: su suma se "acumulará" en el registro. He aquí las instrucciones:

Z80		6502	
LD	A,$42	LDA	#$42
ADC	A,$07	ADC	#$07

Aquí las dos instrucciones del Z80 proveen para que los números se carguen y sumen, mientras que en la versión para el 6502 los números van precedidos por #, que indica que se trata de números verdaderos y no de direcciones. Así, por ejemplo, LDA #$65 significa "cargar el número $65 en el acumulador", mientras que LDA $65 significa "cargar en el acumulador el contenido del byte cuya dirección es $65". Del mismo modo, la instrucción de suma, ADC (que resulta ser una expresión mnemotécnica válida tanto para el Z80 como para el 6502) significa en este caso: "agregar en el acumulador un número verdadero". Los números $42 y $07 son "datos inmediatos", y LDA #$42 se lee como: "cargar en el acumulador los datos inmediatos #$42".

Después de que se hayan ejecutado estas dos instrucciones, el acumulador contendrá la suma de los números. Para nosotros la suma queda allí "invisible", lo que significa que debemos almacenar el contenido del acumulador en un byte de RAM donde se pueda inspeccionar. El programa debe terminar con una instrucción RETURN y comenzar con una instrucción para colocar un registro relacionado de la CPU en el estado correcto, de modo que el programa completo se lee así:

Z80	
Leng. máquina	**Leng. ensamblador**
A7	AND A
3E 42	LD A,$42
CE 07	ADC A,$07
32 ?? ??	LD BYTE 1,A
C9	RET
6502	
18	CLC
A9 42	LDA # $42
69 07	ADC # $07
8D ?? ??	STA BYTE 1
60	RTS

De momento no nos vamos a ocupar del significado de la primera instrucción en ambos programas, pero observe que la cuarta instrucción contiene el símbolo BYTE 1en lugar de la dirección verdadera. El valor de BYTE 1 es distinto de una máquina a otra, de modo que aquí nos limitaremos a utilizar tan sólo el símbolo, y más adelante, cuando lleguemos a ensamblar el código, lo sustituiremos por un número hexa real.

Ahora lo que debemos hacer es decidir dónde colocar el lenguaje máquina y qué dirección representa BYTE 1. Elija un lugar para almacenar el lenguaje máquina y después haga que BYTE 1 sea igual a la dirección del byte después del final del programa, y coloque esa dirección en el lenguaje máquina en forma *lo-hi* (bajo-alto). Después de ello, utilice el programa monitor de la página 598 para cargar y ejecutar el lenguaje máquina y para inspeccionar el byte donde se ha de almacenar el resultado: $49.

Acumulador
Éste es realmente el registro central de la CPU. Las operaciones de aritmética y lógica, así como las transferencias de datos, se conducen fundamentalmente a través de este registro, con mayor intensidad en el 6502 que en el Z80

Unidad aritmético lógica (ALU)
Consta de un sumador binario y puertas lógicas, que permiten acceder a bits individuales de los registros y al bus de datos. Controlada adecuadamente, posibilita la suma, la resta y las operaciones booleanas

Registro del estado del procesador
Siempre que se realiza una operación de la CPU, los bits individuales del registro del estado del procesador muestran algunos de los efectos de la operación: ¿produce, por ejemplo, un resultado cero?, ¿o hay allí un bit acumulado de la operación suma?

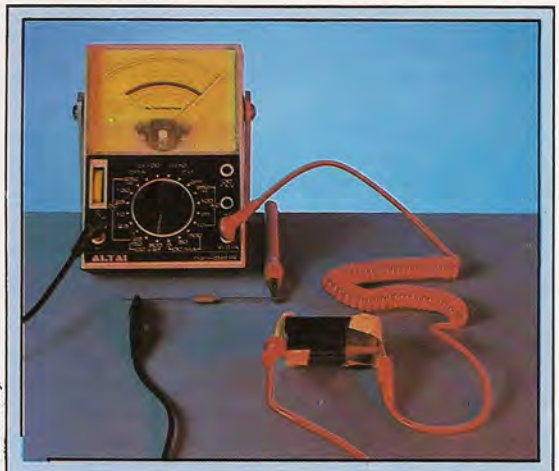
Contador de programa
Señala la dirección de la memoria donde está almacenado el siguiente opcode que va a ejecutar la CPU. La función en BASIC USR(dirección) hace que la dirección especificada se cargue directamente en el contador de programa, de modo que la ejecución de la CPU prosigue a partir de este punto

Indicador de pila
Lleva la dirección del siguiente byte de RAM de espacio de trabajo libre para el empleo de la CPU. Cada vez que la CPU escribe algún dato en él, el indicador de pila cambia para señalar el siguiente byte libre

Registros de índice y uso general
La CPU utiliza los registros de índice para direccionar la memoria de muchas maneras, mientras que los registros de uso general se utilizan como espacio de trabajo general y para fines específicos de la CPU

Partes de una pieza

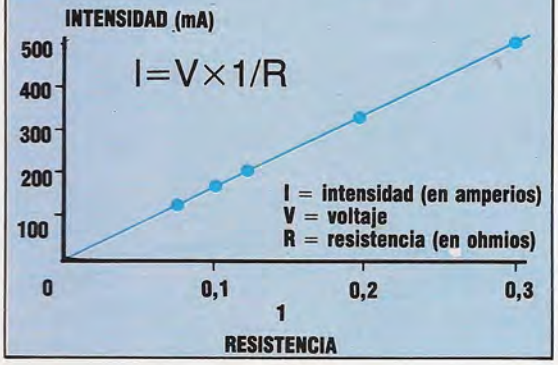
Examinaremos los componentes fundamentales de una pieza y veremos dos modos de calcular los valores de las resistencias



Liz Heaney

Demostrar la ley de Ohm

Se demuestra la ley de Ohm con unas resistencias, una pila, un poco de cable y un multímetro. Las resistencias se valoran a un vatio o más, con valores desde 3,3 ohmios hasta 15 ohmios. Prepare el multímetro para medir la corriente, a la escala de 400 mA (miliamperios) para el indicador. Conéctelo al circuito con los otros componentes, pruebe con cada resistencia y verifique la lectura del multímetro. Si representa gráficamente sus lecturas de corriente en función de uno dividido por los valores de las resistencias tendrá un gráfico de línea recta en el cual I es directamente proporcional a 1/R con un voltaje constante. Luego $I = V \times 1/R$, que es $V = I \times R$, fórmula de la ley de Ohm



Diodos

Los diodos son el equivalente eléctrico de las válvulas unidireccionales. Un diodo posee una resistencia muy baja a la corriente eléctrica en una dirección (por lo general, unos pocos ohmios) y una resistencia muy alta a la corriente en la otra dirección. Esta irregularidad respecto a la ley de Ohm se debe a que el diodo es un semiconductor construido con materiales como el silicio y el germanio. Cada diodo tiene una franja en el extremo hacia el cual puede fluir la corriente

Los microprocesadores y chips de control de video existentes en el corazón de todo microordenador son tan complejos que el amplio alcance de sus acciones y sus diminutas dimensiones hacen que parezcan ajenos al mundo de los humanos.

Aun el más complicado de los circuitos integrados necesita el apoyo de piezas eléctricas mucho más sencillas. Estos componentes simples se pueden clasificar en dos tipos: activos y pasivos.

Los componentes pasivos son los más sencillos. Se limitan a obstaculizar la señal eléctrica que fluye a través de los mismos. Las diferentes señales se obstaculizan de diferentes formas. Por esto resultan tan útiles. Las resistencias y los condensadores constituyen ejemplos de este tipo de componentes.

Los componentes activos tienen más ajetreo. Pueden incrementar la señal que fluye a través de ellos. Un transistor es un buen ejemplo.

Los componentes pasivos se fabrican con materiales simples. Aparte de los materiales de recubrimiento (los diversos plásticos y resinas) que uno ve cuando compra, por ejemplo, una resistencia, el componente está hecho de cobre, acero y carbón. Todos son conductores. Los componentes activos, en cambio, son de silicio o germanio principalmente. Estos dos elementos poseen propiedades especiales que hacen que no se comporten ni como conductores, como el acero y el cobre, ni como aislantes, como las resinas y plásticos. Bajo determinadas condiciones conducen la electricidad, y bajo otras no. Por ello se dice que son semiconductores.

Los componentes activos, debido a que se basan en semiconductores, funcionan de muchas formas interesantes. Uno no puede afirmar que la aplicación de un determinado voltaje a través de un semiconductor dará por resultado que a través del mismo fluya una corriente eléctrica de cierto valor, ni siquiera que fluya alguna corriente en absoluto. Pero, todos los conductores están sometidos a una ley muy sencilla. La ley de Ohm (véase p. 594) predice la forma en que reaccionará cualquier conductor simple ante la señal eléctrica que fluye a través del mismo. Para la informática, el más importante de estos componentes es el transistor, ya que es la base sobre la cual los ordenadores almacenan y procesan la información. En el próximo capítulo utilizaremos transistores para construir algunas de las puertas lógicas simples que hemos analizado en el apartado de *Ciencia informática*.

Spectrum

```
20 DIM B$(3,7)
25 DIM C$(4,7)
30 INPUT "DE ENTRADA AL COLOR DE LAS
   BANDAS"
40 INPUT "BANDA NUMERO 1",B$(1)
50 IF B$(1)="DORADO" OR B$(1)="PLATEADO"
   THEN GO TO 30
60 INPUT "BANDA NUMERO 2",B$(2)
70 INPUT "BANDA NUMERO 3",B$(3)
80 PRINT
90 FOR I=L TO 3
100 RESTORE 180
110 FOR J=0 TO 9
120 READ C$(1),C$(2),C$(3),C$(4)
130 IF C$(1)=B$(I) THEN GO SUB 1000
140 NEXT J
150 NEXT I
160 PRINT "                    OHMIOS"
170 STOP
180 DATA "NEGRO","0","0","0","MARRON",
    "1","1","0","ROJO","2","2","00",
    "NARANJA","3","3","000"
190 DATA "AMARILLO","4","4","0000",
    "VERDE","5","5","00000","AZUL","6",
    "6","000000"
200 DATA "VIOLETA","7","7","0000000",
    "GRIS","8","8",".0","BLANCO","9",
    "9",".0"
1000 FOR K=1 TO 7
1020 IF C$((I+1),K)<>" " THEN PRINT C$
     ((I+1),K);
1025 NEXT K
1030 RETURN
```

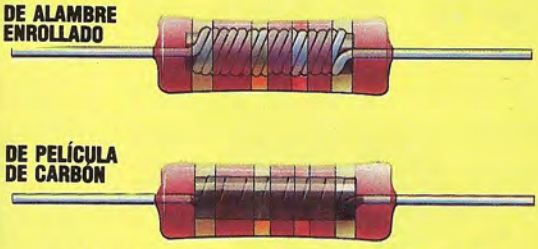
BBC

```
10 REM CODIGOS DE COLOR DE RESISTENCIAS
20 DIM B$(3),C$(3)
30 PRINT "DE ENTRADA A COLOR DE BANDAS:"
40 INPUT "BANDA NUMERO 1",B$(1)
50 IF B$(1)="DORADO" OR B$(1)="PLATEADO"
   THEN GOTO 30
60 INPUT "BANDA NUMERO 2",B$(2)
70 INPUT "BANDA NUMERO 3",B$(3)
80 PRINT
90 FOR I=1 TO 3
100 RESTORE
110 FOR J= 0 TO 9
120         READ C$(0),C$(1),C$(2),C$(3)
130         IF C$(0)=B$(I)
   THEN PRINT C$(I);
140         NEXT J
150 NEXT I
160 PRINT "              OHMIOS"
170 END
180 DATA NEGRO,0,0,, MARRON,1,1,0,
    ROJO,2,2,00, NARANJA,3,3,000
190 DATA AMARILLO,4,4,0000,
    VERDE,5,5,00000, AZUL,6,6,000000
200 DATA VIOLETA,7,7,0000000,
    GRIS,8,8,, BLANCO,9,9,.
```

Para el Commodore 64 reemplace las líneas 40, 60 y 70 por:

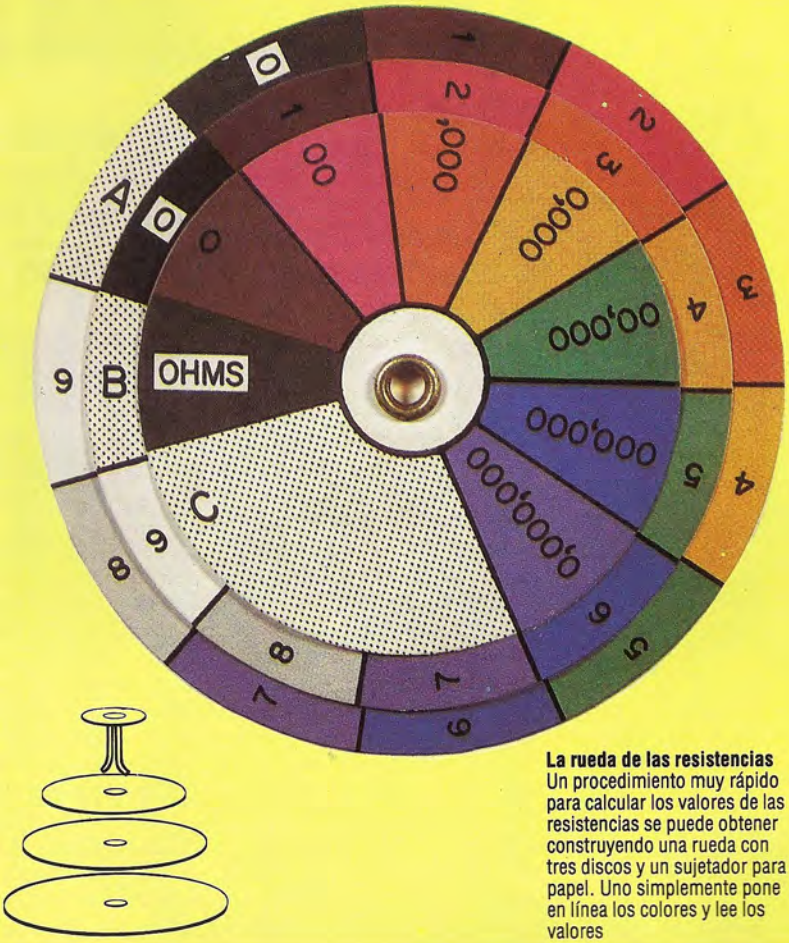
```
40 INPUT "BANDA NUMERO 1";B$(1)
60 INPUT "BANDA NUMERO 2";B$(2)
70 INPUT "BANDA NUMERO 3";B$(3)
```

Resistencias



Las resistencias son una de las piezas electrónicas más sencillas: las más comunes son las de alambre enrollado y las de película de carbón. Las primeras se componen de largos trozos de cable enrollado bien ajustado alrededor de un cilindro aislante y colocado dentro de una cubierta aislante. La corriente fluye a lo largo del cable, lo que causa una reducción de la misma. En las resistencias de película de carbón la corriente pasa a través de un corte en espiral de una película de carbón que hay alrededor del cilindro aislante.

Ambos tipos de resistencia están marcados con bandas de colores para identificar sus valores. Aprender a leer estas bandas es muy sencillo. Las dos primeras representan una cifra para la resistencia expresada en ohmios y la tercera un valor por el que se multiplica esta cifra. La banda final, dorada o plateada, indica la tolerancia del componente y se puede utilizar para señalar en qué sentido se deben leer las otras bandas. Siempre se lee en dirección a la banda dorada o plateada. Para no aprenderse de memoria los colores, use un corto programa como el listado al margen para calcular los valores de las resistencias



La rueda de las resistencias
Un procedimiento muy rápido para calcular los valores de las resistencias se puede obtener construyendo una rueda con tres discos y un sujetador para papel. Uno simplemente pone en línea los colores y lee los valores

Condensadores

Los condensadores resisten la corriente alterna. Las corrientes CA de alta frecuencia fluyen a través de los condensadores más fácilmente que las de baja frecuencia, lo que resulta útil para limpiar o filtrar una señal eléctrica. Muchos tableros de microprocesadores están llenos de condensadores con este único objetivo. Esencialmente se componen de dos placas conductoras separadas por una capa aislante de varias sales especiales (para voltajes altos) o de cerámica (para voltajes bajos). Las placas pueden ser bastante grandes y las dimensiones reales del componente se reducen enrollando las placas de tal manera que formen una espiral compacta

Transistores

El transistor es la más compleja de estas piezas: es un dispositivo semiconductor y su invención, en 1947, marcó el comienzo de la electrónica moderna. Un transistor posee dos aplicaciones básicas. Como amplificador, puede tomar una pequeña corriente de entrada y producir una gran corriente de salida. Como interruptor, se puede utilizar una corriente para encender o apagar otra corriente. Esta capacidad para actuar como un interruptor electrónico es la base de toda la electrónica digital y es un factor esencial para el funcionamiento de los ordenadores.

Al igual que los diodos, los transistores se componen de materiales semiconductores, pero en su interior poseen dos junturas en lugar de sólo una. Hay tres cables que van hacia los semiconductores, que suelen denominarse base, colector y emisor. Es la forma en que estos tres cables se acoplan al circuito lo que determina si el transistor actúa como un amplificador o como un interruptor. A diferencia de las resistencias y de los diodos, no existen formas estandarizadas para identificar los cables de un transistor ni sus especificaciones. El método usual consiste en consultar el libro de referencias del fabricante para saber el número de ese componente en particular

Kevin Jones

La lozanía de Oric

Esta novel compañía se encuentra en pleno ascenso en el mercado de la informática

Oric Products International se creó para competir con Sinclair Research. Financiada por la British Car Auctions, sus directivos provinieron de Tangerine Computer System y Tansoft.

Cuando se fundó, en 1982, Oric amalgamó la ciencia del marketing (representada por el director gerente Barry Muncaster y el director de ventas Peter Harding) con la del diseño (avalada por Paul Johnson, en hardware, y Paul Kaufman, en software). La máquina creada, el Oric-1, es un equivalente del Spectrum, pero con un 6502. Viene en versiones de 49 Kbytes y 16 Kbytes y, en comparación con el Spectrum, su teclado es mejor, viene en una carcasa más sólida y posee gráficos y sonido superiores. Su BASIC residente es la versión Microsoft estándar de que disponen ordenadores como el Vic-20 y el Apple. Asimismo, el Oric posee una interface para impresora Centronics estándar, que le permite conectar directamente con impresoras de tamaño normal.

Sin embargo, la empresa hubo de hacer frente a una serie de obstáculos importantes, sufriendo demoras en las entregas y fallos de diseño. Las primeras máquinas tenían problemas con la carga de cintas y sobre visualizaciones inestables en pantalla. La ROM de BASIC, con un buen número de errores pequeños pero molestos, hacían que la programación resultara delicada. Además, las técnicas para gráficos del Oric, que economizaban memoria, dificultaban enormemente la programación. Estos problemas se acentuaron cuando la empresa se vio obligada a reajustar el precio de la máquina, situándolo por encima del precio del Spectrum.

La gama Oric
La apariencia futurista del Oric-1 duró quince meses, hasta que se decidió remodelarlo y crear así el elegante Atmos, rojo y negro, con un teclado completo y una ROM de BASIC revisada. La impresora-plotter Oric, a cuatro colores, también se ha vuelto a diseñar para armonizar con la nueva línea

A consecuencia de estas dificultades, al principio el ordenador no se vendió bien y era difícil conseguir software para él. Tansoft trabajó mucho para apoyar a la máquina, proporcionando lenguajes como assembler y FORTH, y es así como actualmente el Oric posee un respetable surtido de software.

Oric también prometió una gama completa de accesorios, incluyendo unidades de disco, un modem y una impresora. De éstos sólo apareció la impresora-plotter a cuatro colores. Después de este tormentoso comienzo, lentamente el Oric ha ido ganando terreno. Las ventas al extranjero arrojan cifras impresionantes. De las 170 000 máquinas que se fabricaron en 1983, más del 50 % se destinaron a la exportación.

Quince meses después de la fecha de su lanzamiento, Oric ya había superado de una forma muy novedosa muchos de los problemas que afloraron con la aparición del Oric 1. Lanzó el modelo Oric Atmos, presentando un teclado completo y unas ROM de BASIC mejoradas; pero en todos los otros aspectos, se trata de la misma máquina remodelada. Con el anuncio de la unidad de disco Hitachi de tres pulgadas para complementar el sistema, el Oric Atmos empieza a parecerse a un micro basado en disco que ofrece una buena relación calidad-precio y en competencia directa con el Acorn Electron y el Commodore 64 más que con el Spectrum.

No obstante, otra vez el optimismo inicial se ha visto rebajado por nuevos problemas. Hubo demoras en los envíos de las unidades de disco y el Atmos tiene ciertas dificultades para cargar y ejecutar las cintas ya existentes. Pero se ha creado un nuevo centro de estudios y proyectos en Cambridge que, junto con un acuerdo comercial con Inmos, el fabricante británico de semiconductores, habrá de proporcionar una sólida base sobre la cual trabajar en el futuro.

Estrella del futuro

Este ordenador norteamericano incorpora muchas de las configuraciones estándar del avanzado MSX, lo que, unido a su capacidad de ampliación, le augura un brillante porvenir

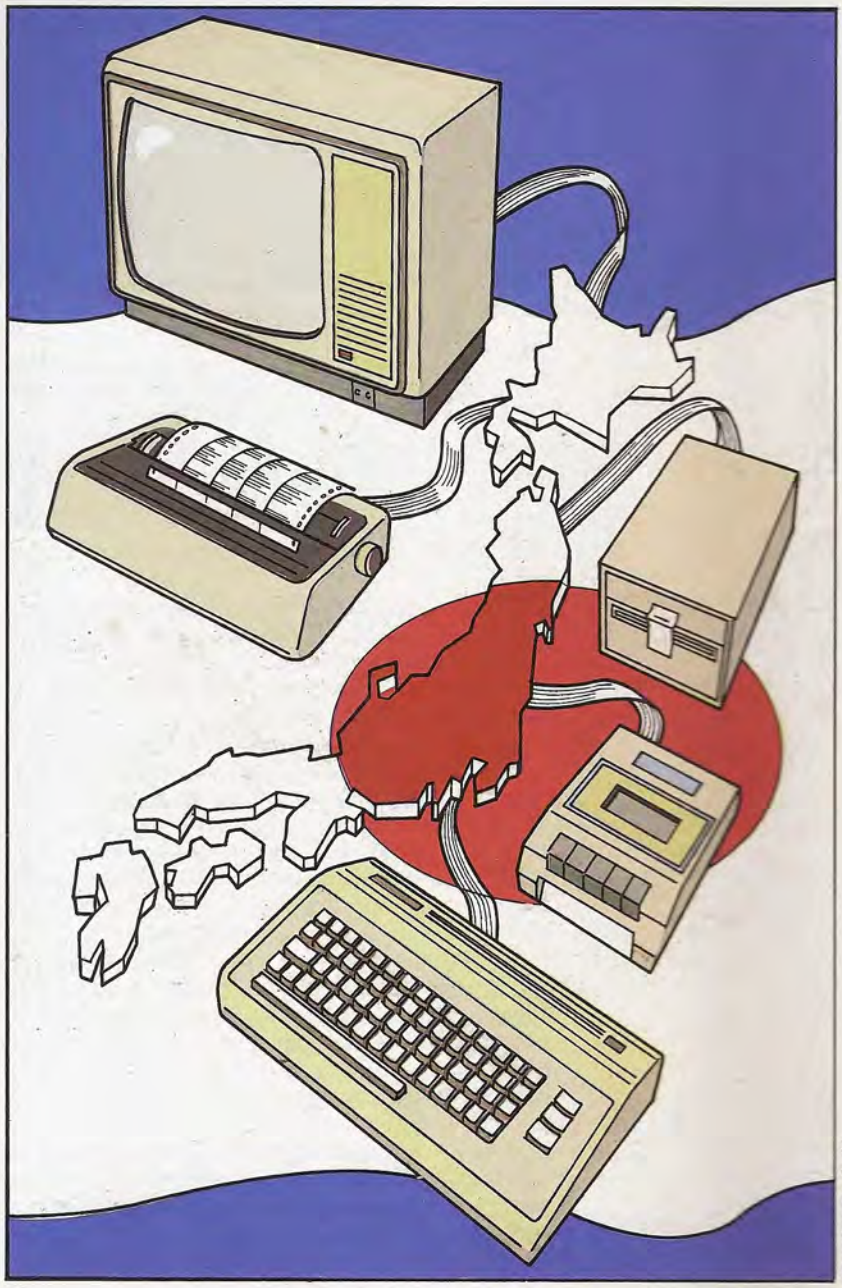
Para entender por qué los distintos ordenadores personales son tan singulares en cuanto a diseño necesitamos conocer cómo se desarrolló la industria del microordenador. Los primeros microprocesadores que causaron un impacto en el mercado del ordenador personal fueron el 8080 de Intel y, presumiblemente, el 6800 de Motorola. Los conjuntos de instrucciones de estos procesadores se fijaron desde el comienzo. El problema de la compatibilidad surgió por la versatilidad de los procesadores. Por ejemplo, enviar un byte de datos a una puerta de salida implicaba exactamente la misma instrucción, cualquiera que fuera el ordenador, siempre y cuando se utilizara el mismo procesador. Pero la salida podía tener cualquier dirección entre centenares o millares de direcciones. Muchos de los creadores de los primeros micros personales los construyeron en los garajes o en los patios de sus casas, por lo cual, como es natural, no tenían ningún tipo de acuerdo entre ellos acerca de en qué lugar del mapa de memoria deberían localizarse las puertas de entrada o salida. El fabricante A podía escoger que la puerta para salida a impresora en paralelo estuviera localizada en la dirección 255, mientras que el fabricante B podía escoger para lo mismo la dirección 254.

Esta situación ya era negativa de por sí, pero con el advenimiento de los gráficos en pantalla controlados por chips controladores CRT especiales, y los efectos de sonido controlados por chips exclusivos para sonido, la situación se hizo todavía más caótica. Y así se garantizó que todo programa que aprovechara al máximo las capacidades especiales de cualquier ordenador no se pudiera ejecutar en otro ordenador como no fuese a costa de un considerable trabajo de reescritura.

De haber habido alguna empresa que estuviera en una posición de suficiente poder como para forzar un estándar desde el principio, la situación habría sido totalmente distinta. Pero no fue ése el caso. En los primeros tiempos de los ordenadores personales había numerosos pequeños fabricantes, cada uno de ellos con un estilo y un estándar propios y diferentes. No se trataba tan sólo de que las máquinas eran distintas las unas de las otras física y electrónicamente, sino que hasta los lenguajes de programación que se proporcionaban con ellas eran incompatibles con otras máquinas. Desde el principio, el BASIC jamás respondió a un estándar. Hasta la década de los setenta, como mínimo, la comunidad informática profesional no se tomaba muy en serio el BASIC, al que consideraban un lenguaje estrictamente para principiantes.

Desde finales de los setenta a principios de los ochenta los microordenadores se desarrollaron a ritmo acelerado. Diseños pioneros, como el Apple, comenzaron a incorporar refinamientos tales como gráficos y color sofisticados. Para que estas innovaciones se pudieran utilizar, los fabricantes tuvieron que desarrollar sus propias versiones de BASIC, y así empezó la diversificación del lenguaje.

El precio de esta proliferación de versiones no es tanto la disposición de mayores opciones por parte del consumidor, sino más bien la frustración que



Tony Duncan-Smith

El camino hacia el éxito
El estándar MSX es la ruta japonesa hacia los mercados mundiales de ordenadores. De conseguir el éxito, Japón podría llegar a dominar el mercado del ordenador, tal como consiguiera hacerlo en los campos de la alta fidelidad y las cámaras fotográficas

experimentan por igual propietarios, fabricantes y escritores de software.

Si usted fuera un fabricante de ordenadores con un nuevo producto que lanzar al mercado, sabría que prácticamente no habrá software para su máquina hasta que se haya establecido un sustancial número de usuarios. Con poco o nada de software compatible con su ordenador, será muy consciente de que las previsiones de ventas de su máquina son limitadas y que esto puede poner en peligro la inversión que supone el desarrollo del producto.

Si se ganara la vida escribiendo software comercial, sus ventas estarían limitadas (en el mejor de los casos) a la cantidad de personas que posean el modelo de ordenador para el cual ha escrito el programa. Supongamos que ha creado un juego de aventuras que se llama *Los calabozos de Rathbone*, todo completo, con una Presencia Sin Rostro, un sádico jefe de calabozos e innumerables escondrijos siniestros para atrapar a los incautos. Los investigadores de mercado podrían afirmar que el potencial de ventas del producto sería inmenso si pudiera sacarlo al mercado al precio de 1 200 pesetas y vender al menos 65 000 ejemplares. Lamentablemente, las previsiones del producto son tan altas que sería improbable que el número de usuarios de Spectrum por sí solo generara la cifra de ventas requerida, y se necesitaría al menos una versión más. El costo que supondría la producción de nuevas versiones para, pongamos por caso, el Oric y el BBC Micro elevaría el precio de la unidad a 1 500 pesetas y, de este modo, las ventas caerían por debajo de un nivel económicamente viable. Éste es el dilema que frustra a muchos posibles escritores de software.

La industria informática no ha ignorado el problema de la incompatibilidad de software. El individualista mundo occidental no se muestra muy dispuesto a la estandarización, pero quienes desarrollan ordenadores en el Lejano Oriente prefieren que las cosas sean sistemáticas y estandarizadas.

ASCII/Microsoft está intentando reemplazar el caos por el orden: La empresa es el resultado de una fusión en la que están comprometidas la American Microsoft Corporation y ASCII, una exitosa editorial japonesa que tiene en su haber una serie de populares publicaciones. Como actividad complementaria, ASCII también edita software comercial, y cuando Microsoft quiso introducirse en el impenetrable mercado japonés, ASCII pareció la firma idónea a la cual dirigirse para una empresa conjunta. El razonamiento era muy válido, ya que Microsoft poseía la necesaria experiencia técnica y ASCII la experiencia en comercialización.

American Microsoft Corporation ha conseguido su prestigio en virtud de lo que más se parece a un estándar del BASIC, el MBASIC Microsoft, que han adoptado muchos fabricantes de ordenadores en todo el mundo. Pero, así y todo, no se podía garantizar que cualquier programa en MBASIC se pudiera ejecutar en todos los ordenadores que utilizaran ese lenguaje siempre que hubiera de por medio alguna configuración especial, simplemente debido a la carencia de una compatibilidad de hardware.

El BASIC Microsoft se vendió con gran éxito a numerosos fabricantes japoneses de ordenadores, a través de las oficinas de ASCII/Microsoft. Pero aun así esto no solucionó el problema de la compatibilidad de hardware y software. La solución de ASCII/Microsoft consistió en crear un estándar, en colaboración con los principales fabricantes japoneses, con el deseo de que se lo reconociera internacionalmente como tal. Al resultado final que obtuvieron se lo denomina Estándar MSX. La especificación incluye exigencias básicas de hardware (basadas alrededor del microprocesador Z80 y otros chips fijos), así como un lenguaje estandarizado.

Las especificaciones MSX

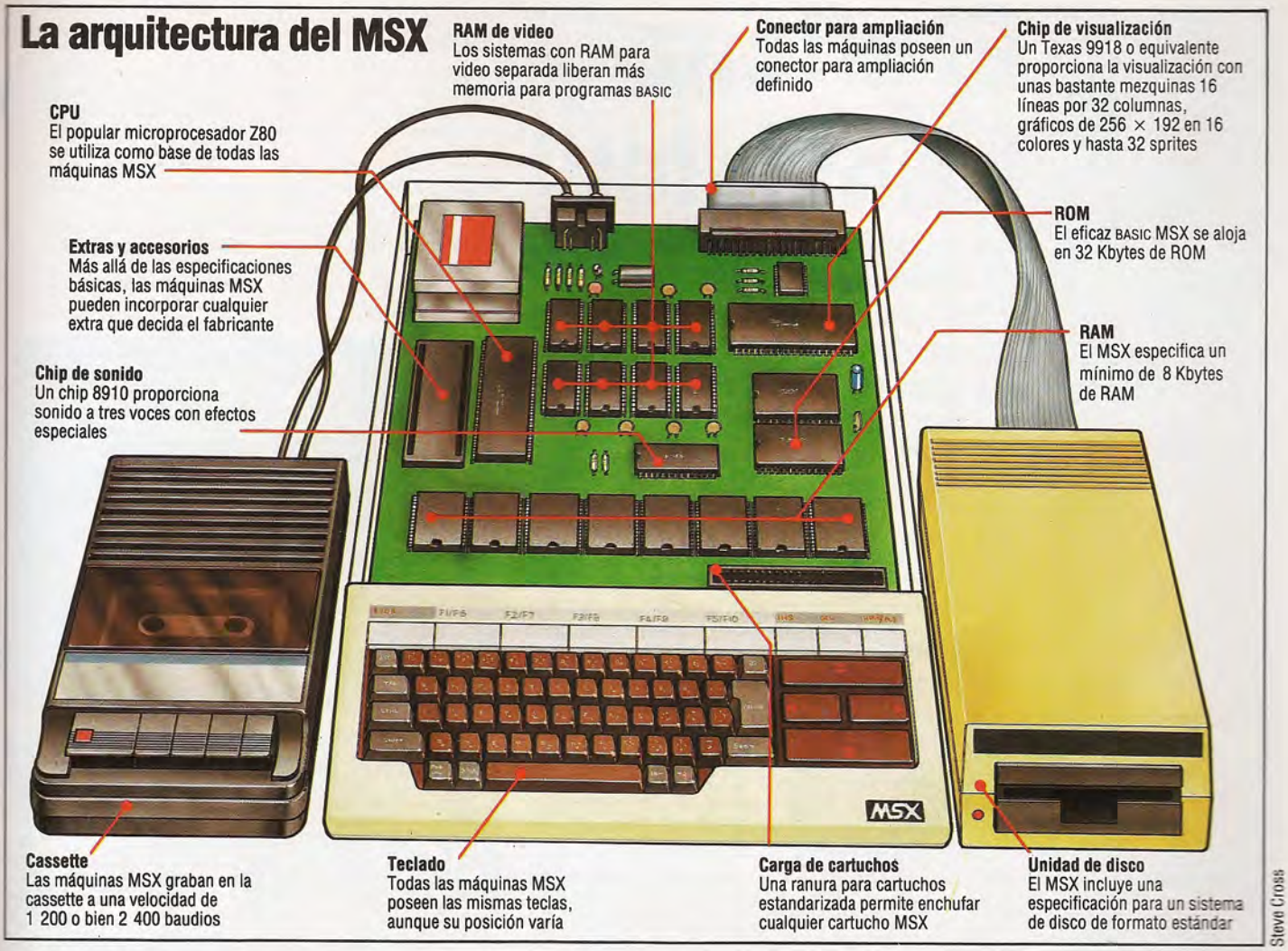
El BASIC MSX es muy similar al MBASIC de Microsoft, pero incluye varias significativas mejoras para sacar partido de las capacidades para gráficos y sonido actuales. Entre las sentencias incorporadas se incluyen SCREEN, para especificar la modalidad de pantalla, el tamaño de los sprites, el "tecleo" de teclas, la velocidad baudio de la cassette y las opciones de impresora; LOCATE, para el posicionamiento de caracteres en la pantalla; COLOUR, para seleccionar uno de los 16 colores de fondo y primer plano; PUT SPRITE, para establecer las características de los sprites; CIRCLE, para trazar círculos y elipses; DRAW, para dibujar figuras; LINE, para trazar líneas entre coordenadas especificadas, y PAINT, para rellenar figuras con un determinado color. También se proporciona una sentencia KEY para asignarles cadenas a las teclas de función. Hay otras sentencias para colocar valores en la RAM de video (VPOKE), escribir valores en los registros del chip de efectos sonoros (SOUND) y para controlar el motor de la cassette (MOTOR).

El MSX, sin embargo, implica algo más que un software estandarizado. La CPU se especifica como un Z80 que trabaja a 3,58 MHz. Debe haber al menos 32 Kbytes de ROM para almacenar el software MSX. Además, debe haber al menos ocho Kbytes de RAM. No existe límite máximo en cuanto a la cantidad de ROM y RAM permitida. Un ordenador MSX ha de incorporar un chip controlador de video TMS9918A de Texas Instruments (o equivalente) y un AY-3-8910, chip generador de sonido a tres voces. La salida de video debe ser capaz de visualizar o 32 columnas por 24 líneas, o bien 40 columnas por 24 líneas. En la actualidad, en las especificaciones no está prevista una visualización a 80 columnas. Se requiere una resolución de 256 por 192 pixels.

La interface para cassette se ha determinado, siendo las cassettes el medio físico principal para el almacenamiento de programas y de datos. Debe utilizar el sistema de codificación FSK a 1 200-2 400 bits por segundo. El teclado, reflejando el linaje japonés del MSX, no sólo tiene un trazado estándar, incluyendo teclas de función, sino que proporciona las dos categorías de signos silábicos *kana*, escritos en columnas verticales: *katakana* (fragmentos de ideogramas) e *hiragana* (abreviaciones de ideogramas, trazadas en cursiva); además suministra caracteres estandarizados para gráficos y, opcionalmente, un sistema de escritura con caracteres chinos.

Se provee software enchufable a través de una ranura estándar para cartuchos de ROM y hay un bus de entrada-salida, también estándar, de 50 patillas. Hay incluso una puerta estandarizada para dos palancas de mando.

Los formatos de disco están, asimismo, estandarizados, al igual que el sistema operativo en disco,



el MSX-DOS. Funcionalmente es equivalente al MS-DOS y permite leer archivos de datos de éste. También se afirma que es compatible con el enormemente popular sistema operativo en disco CP/M 2.2. También se han determinado formatos para los discos flexibles de 3 ½ pulgadas (Sony), de 5 ¼ pulgadas y de 8 pulgadas.

Todo esto significa que, de haberse ceñido a la normativa, cualquier programa escrito para una máquina MSX, almacenado en cualquier disco, está garantizado para su ejecución en cualquier otro ordenador MSX y será capaz de aprovechar por completo sus capacidades para gráficos y para sonido. Las ventajas, tanto para el fabricante como para el consumidor, son obvias.

Pero el sistema MSX tiene algunas desventajas. La primera de ellas es que todo "estándar" no conseguirá sacar partido de las innovaciones que se produzcan.

El segundo inconveniente reside en que los microprocesadores de ocho bits, de los cuales el de más éxito es el Z80, tienen una vida limitada, son incapaces de direccionar más de 64 Kbytes de memoria principal directamente, y tampoco pueden manipular datos cuyo valor sea superior a 256. Desde este punto de vista, se podría considerar que el estándar MSX es un intento por quemar el último cartucho para prolongar la vida del Z80, que está destinado a disfrutar tan sólo de un éxito a corto plazo en el mercado.

El MSX tal vez proporcione un indicador para el futuro. Es difícil suponer seriamente que el mercado del microordenador vaya a estar dominado por los procesadores de ocho bits dentro de cinco o diez años. Si el estándar MSX consigue algo, es probable que este algo sea recordarles a quienes tienen a su cargo el desarrollo y perfeccionamiento de los ordenadores que en toda innovación en el campo de la informática, la estandarización debe venir lo más temprano posible. El MSX puede constituir una oportuna aportación para aquellos fabricantes que actualmente tienen productos basados en el Z80 a la espera de ser vendidos; es difícil predecir si producirá algún impacto a largo plazo, como no sea convencer al resto del mundo de que la estandarización es importante. En el campo de los ordenadores de 16 bits, IBM ha demostrado que "querer es poder" con su ordenador personal, que se ha convertido en un estándar de facto. ¿Será capaz el estándar MSX de hacer lo mismo por el micro personal de ocho bits? Hasta el momento el MSX está respaldado por un total de 16 fabricantes, incluyendo a Yamaha, JVC, Hitachi, Sony, Sanyo, National, Pioneer, Canon, Fujitsu y Mitsubishi, de Japón, la empresa norteamericana Spectravideo y la coreana Daewoo. Ningún fabricante británico se les ha unido hasta la fecha. Sólo el tiempo y el mercado demostrarán si el mundo necesita más de lo mismo, o bien el tipo de innovaciones individualistas como las aportadas por sir Clive Sinclair.

Convención de diseño
Para lograr que los programas y los accesorios sean compatibles con todos los sistemas MSX, el diseño de un micro MSX sigue reglas estrictas. Una vez la máquina ha alcanzado las especificaciones básicas que reflejamos aquí, los diseñadores pueden agregar sus propios extras especiales

El eficaz transistor

En informática, el más significativo de los componentes electrónicos es el transistor. ¿Cuál es su función?

Los transistores tienen dos misiones esenciales: actuar como amplificador de una señal, o encender y apagar una corriente bajo el control de otra corriente. Es esta habilidad para actuar como un interruptor electrónico lo que los hace útiles en informática. Agrupándolos entre sí, uno puede construir circuitos que almacenen patrones encendido-apagado (*on-off*) que el ordenador puede tratar como números binarios. Otros circuitos son puertas lógicas que permiten sumar secuencias encendido-apagado entre sí, etc.

Si usted construye las puertas lógicas descritas en la página contigua, observará que los ordenadores construidos enteramente a partir de transistores serían máquinas sumamente grandes y caras, tal como lo fueron en realidad alguna vez. Los pequeños ordenadores, de precio razonable, todavía necesitan otro refinamiento: los circuitos integrados. Éstos son circuitos predefinidos con cientos de transistores grabados en un diminuto chip de silicio, encerrados en una cubierta de plástico negro. Usted puede comprar circuitos integrados simples (denominados *chips TTL*), que realizarán satisfactoriamente la mayoría de las tareas que se les encomiende. Cuatro puertas AND en un chip pueden adquirirse a un precio bastante asequible.

La práctica de colocar cada vez más circuitos en un único chip se conoce con las siglas VLSI (*Very Large Scale Integration*: integración a escala muy grande). Los chips VLSI poseen muchos miles de componentes comprimidos en ellos y pueden realizar tareas muy complicadas. El microprocesador es un chip de esta clase. Y también lo son los chips que generan la visualización en televisión, controlan las interfaces y producen la gama de efectos de sonido que pueden proporcionar muchas máquinas. Los principios son los mismos que los utilizados para tres puertas lógicas simples.

Los chips VLSI se construyen mediante tecnologías diferentes, según la relación que se desee obtener entre economía, rendimiento y consumo de energía eléctrica. El tipo que incorporan la mayoría de los ordenadores son los chips MOS (metal-óxido-silicio), mientras que muchos portátiles a pilas utilizan chips CMOS (metal complementario-óxido-silicio), que son más lentos pero necesitan muy poca alimentación eléctrica.

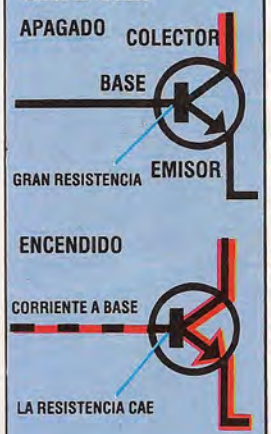
En el próximo capítulo de *Bricolaje* construiremos un sumador incompleto, basado en el circuito de la página 513 de la serie *Ciencia informática*.

El transistor como amplificador

Un transistor tiene tres partes, denominadas base, colector y emisor, con tres cables conectados a cada una de ellas. La base suele utilizarse para controlar una corriente que fluya desde el colector al emisor. Si se aplica una corriente a la base, otra puede fluir del colector hacia el emisor. Una corriente cambiante aplicada a la base hace que la resistencia del trayecto del colector al emisor varíe al unísono. Alimentando en el colector una gran corriente y variándola con una pequeña a través de la base, el emisor produce una señal grande que varía exactamente como la señal pequeña. El transistor se ha utilizado para amplificar la señal pequeña

El transistor como interruptor

La conmutación mediante un transistor supone la utilización a fondo de las propiedades de la base, del colector y del emisor. Un transistor sólo puede dejar pasar una determinada cantidad de corriente a través de su trayecto del colector al emisor. Tiene un punto de saturación en el cual la cantidad de corriente que fluye a través de ese trayecto ya no se ve afectada por las pequeñas variaciones en la cantidad de corriente que fluya a través de la base

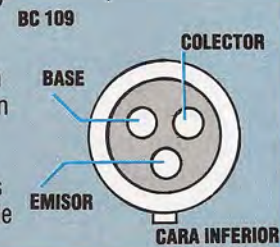


Creación de puertas lógicas

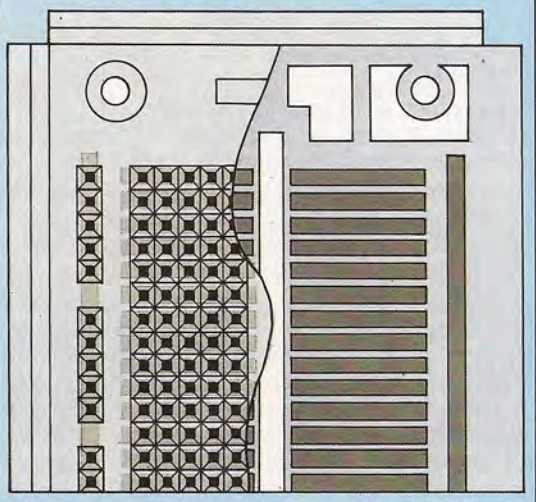
Estos sencillos circuitos (derecha) ilustran la forma en que se puede emplear la capacidad de conmutación del transistor para construir puertas lógicas. Usted mismo puede crearlas de una en una utilizando el mismo conjunto de componentes y un tablero de prueba. Es importante notar que la conmutación real está "transistorizada", o sea, no posee partes móviles. En estos ejemplos, las entradas a los circuitos son botones con indicadores LED (diodos emisores de luz). En un ordenador, las entradas a tales circuitos serían las salidas de otros. Después de haber construido y comprendido estas puertas, quizá se anime a construir un circuito más complejo alimentando la salida de una puerta a otra

Materiales

- 1 tablero de prueba
- 2 transistores BC109
- 2 LED rojos
- 1 LED verde
- 3 resist. de 500 ohm
- 2 resist. de 15 K ohm
- 2 interruptores
- 1 pila de 9 voltios
- 1 conector para pilas
- Trozos cortos de cable

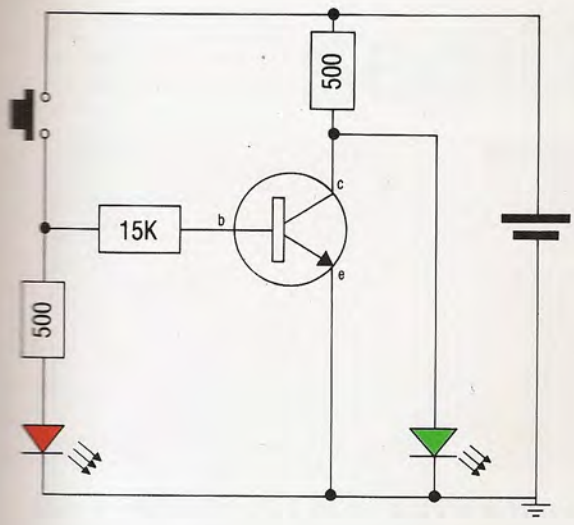


Tableros de prueba

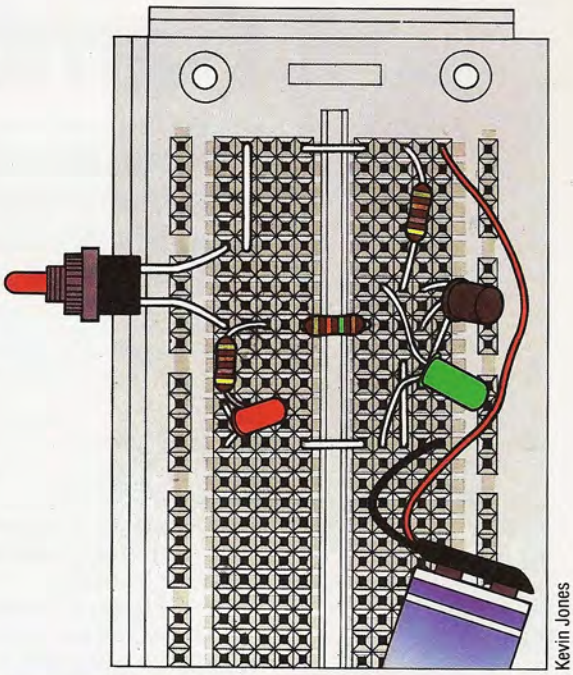


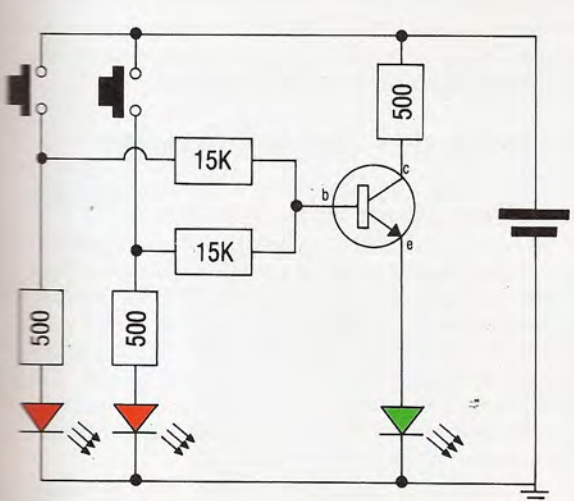
Kevin Jones

Una placa de circuito impreso como la de la figura proporciona una forma sencilla de experimentar con circuitos como los que mostramos sin perder tiempo ni trabajo en soldar los componentes. Consiste en una base siempre recuperable, sobre la cual se enchufan con fijeza los componentes. Las pinzas metálicas para los componentes actúan como conductores, de modo que cada grupo de cinco agujeros está conectado eléctricamente. Con esta matriz es fácil trasladar los diseños de circuitos simples al tablero, utilizando trozos cortos de cable para conectar grupos separados de agujeros. El tablero ilustrado aquí es más grande que el que usted necesita ahora, pero le será útil para futuros y más ambiciosos proyectos

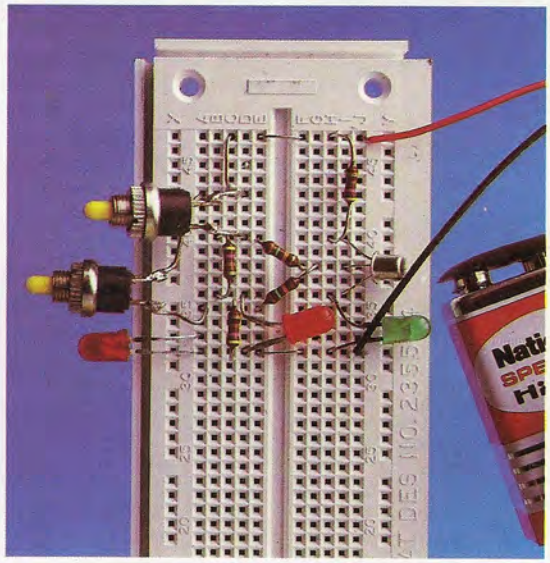


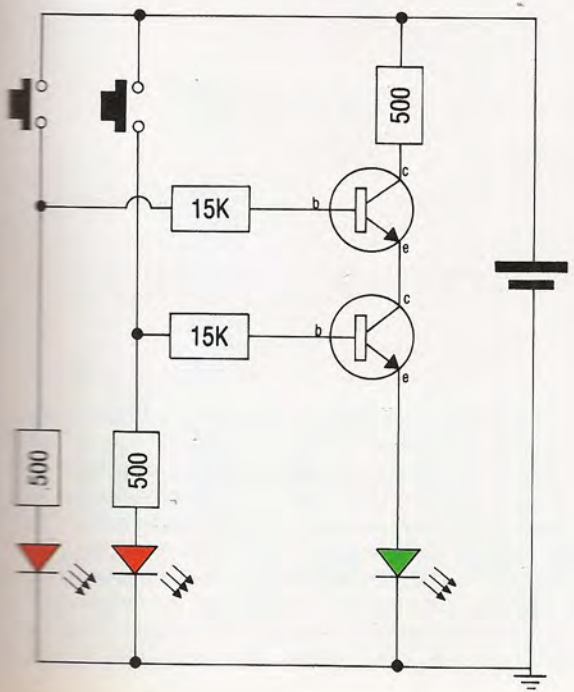
La puerta NOT
Ésta es la puerta lógica más sencilla, con una única entrada (indicada mediante el LED rojo y el interruptor) y una única salida (el LED verde). Con el interruptor abierto, no puede fluir ninguna corriente a través de la base hacia el emisor del transistor. Esto produce una gran resistencia en el colector y, como resultado, el flujo de corriente toma la ruta alternativa a través del LED verde. Por consiguiente, cuando no se aprieta el botón (o sea, cuando la entrada es 0), el LED verde se ilumina (la salida es 1). Al apretar el botón, se alimenta la base del transistor con una corriente, eliminando la resistencia proveniente del colector. Ahora la corriente fluye a través del transistor, evitando el LED. De modo que cuando se pulsa el botón (entrada 1) el LED no se ilumina (salida 0)


Kevin Jones

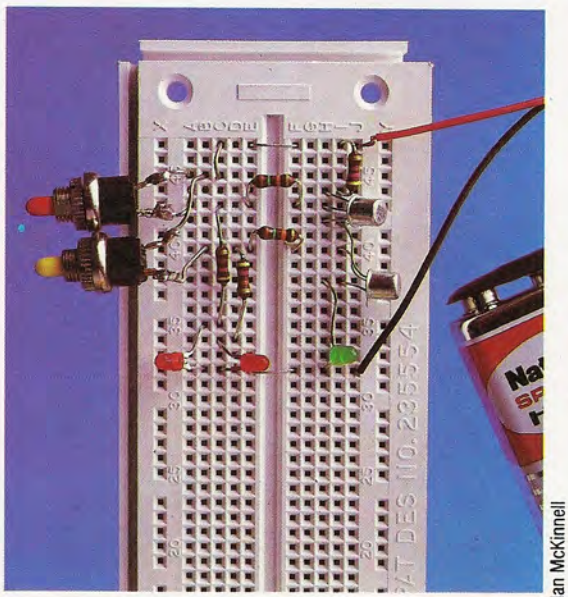


La puerta OR
El circuito utilizado para la puerta NOT se puede adaptar fácilmente para convertirlo en una puerta OR. La primera modificación consiste en colocar la salida LED de modo tal que la active una corriente que pase *a través* del transistor. En una puerta OR existen dos entradas, cada una de ellas con interruptor y LED. Cuando alguno de los interruptores alimenta una señal a la base, conmuta el transistor para permitir que la corriente atraviese el LED





La puerta AND
Para crear una puerta AND necesitamos dos transistores en el trayecto hacia la salida, cada uno con su propia entrada. Sólo cuando se pulsen ambos interruptores (es decir, cuando las dos entradas sean 1) se abrirán los dos transistores, permitiendo que la corriente fluya a través del LED


Ian McKinnell

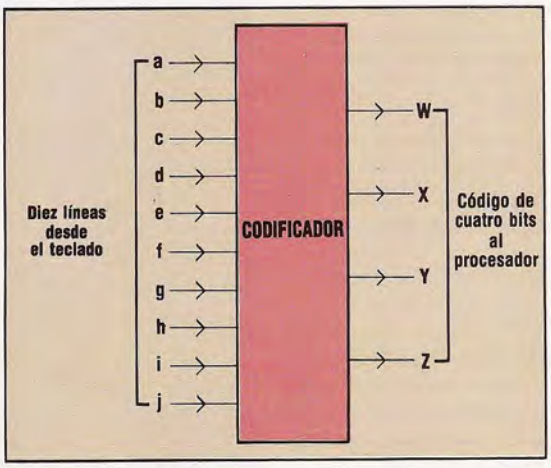
Cambiar de línea

Analicemos los circuitos codificadores y decodificadores, que traducen las instrucciones a señales eléctricas y viceversa

La CPU de un ordenador personal realiza su función mediante el envío de instrucciones en forma de impulsos eléctricos, canales inferiores de comunicación con dispositivos internos o periféricos.

El procesador envía instrucciones tanto a dispositivos internos (el acumulador, la ALU, etc.) como a equipos periféricos (p. ej., una impresora). Con frecuencia se suele reducir el número de líneas utilizadas por un dispositivo periférico que proporciona al procesador una entrada simplificada. Esto se denomina *codificación*. Un ejemplo de codificador es un circuito utilizado en conjunción con un teclado. Éste podría tener 64 líneas de salida, una de las cuales produce una señal cuando se pulsa la tecla correspondiente. Como por lo general se pulsa una sola tecla cada vez, cada una de las 64 posibles señales de salida se puede codificar como un número binario de seis bits (2^6=64). Esto significa que sólo se requieren seis líneas para transportar hasta el procesador la información relativa a cuál de las teclas se ha pulsado. El dispositivo que convierte 64 líneas en 6 líneas es un codificador.

Para demostrar este principio vamos a considerar un teclado mucho más sencillo, de sólo 10 teclas, por ejemplo, para que el usuario trate números entre 0 y 9, ambos inclusive. Un código binario de tres bits nos daría solamente ocho (2^3) combinaciones posibles, de modo que debemos diseñar nuestro codificador con 10 líneas de entrada y cuatro líneas de salida. Como sólo una de las 10 líneas se puede activar en cada momento, la tabla de verdad para el codificador será:

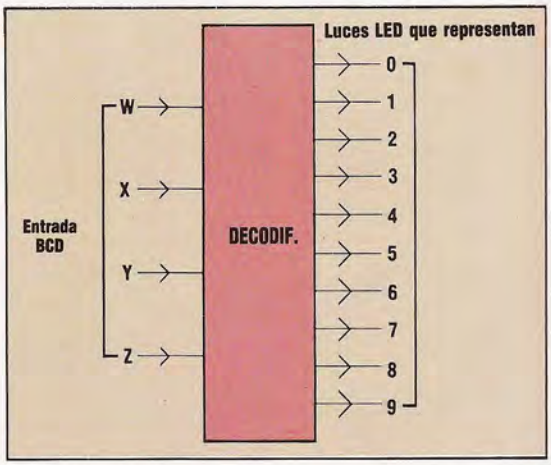


Decodificar es lo contrario de codificar. En vez de aceptar un gran número de líneas de entrada para producir un número pequeño de líneas de salida, un decodificador acepta un número pequeño de entradas (por lo general, en forma de códigos binarios provenientes del procesador), y a partir de esto selecciona una de entre las numerosas líneas de salida, que controlan la actividad de un dispositivo de salida, como puede ser una impresora o un plotter x-y. Los codificadores y decodificadores también se utilizan en el control de los movimientos de la cabeza de disco y para seleccionar los canales de salida provenientes de números de dispositivo.

Decimal	Entradas										Salidas			
	a	b	c	d	e	f	g	h	i	j	W	X	Y	Z
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	0	0	0	1	0	1
6	0	0	0	0	0	0	1	0	0	0	0	1	1	0
7	0	0	0	0	0	0	0	1	0	0	0	1	1	1
8	0	0	0	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	1	1	0	0	1

Diseño del decodificador

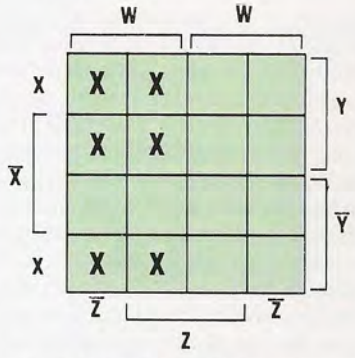
Ahora vamos a analizar cómo se puede diseñar un decodificador simple utilizando puertas AND, OR y NOT, a partir de la consideración del siguiente problema. Se requiere un decodificador para convertir códigos decimales codificados en binario (*Binary-Coded Decimal*: BCD) a una forma que encienda (coloque en *on*) una de las 10 luces LED correspondientes al valor decimal del código. Ciñéndonos al ejemplo, se trata de un circuito que producirá lo contrario que el codificador mencionado anteriormente.



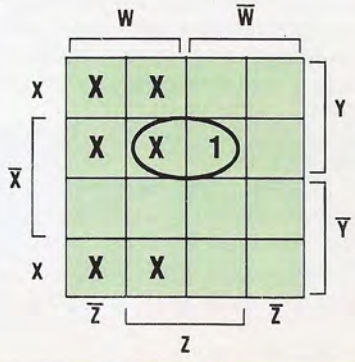
Los códigos decimales codificados en binario son las representaciones binarias de cuatro bits de los dígitos decimales de 0 a 9 y, por lo tanto, el decodificador tendrá cuatro líneas de entrada. Como se puede establecer en alto cualquier combinación de las cuatro líneas, hay 16 (2^4) posibles entradas. Nosotros sólo estamos interesados en las primeras 10 combinaciones y, por consiguiente, podemos aplicarles condiciones de "entrada invalidada" (X) a las otras seis. La tabla 1 (al margen) es la tabla de ver-

dad. Ahora bien, debemos considerar cada una de las diez salidas por separado. Parecería que la expresión booleana para cada salida debería constar de cuatro términos W, X, Y y Z, como en la tabla 2.

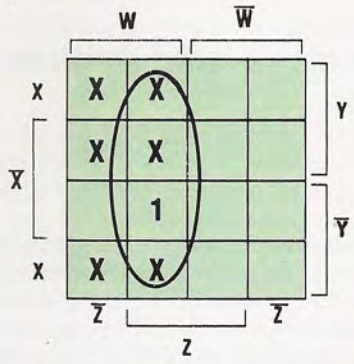
Sin embargo, podemos efectuar alguna simplificación. Como las condiciones de entrada invalidada son las mismas para cada una de las 10 salidas, podemos unir términos entre sí para simplificarlos. Abajo vemos un diagrama de Karnaugh de cuatro variables en el que están marcadas las seis condiciones de entrada invalidada.



Si ahora consideramos las salidas una a una, podremos también aquí efectuar alguna simplificación. Tomemos, por ejemplo, la salida para el dígito 3. Colocando en el diagrama la expresión booleana podemos ver que se puede dibujar un óvalo que la englobe.



De manera que la expresión para la salida se puede simplificar así: $\overline{X}.Y.Z$. Tomando la salida 9 como segundo ejemplo:



Se puede trazar otro óvalo que utiliza tres de las condiciones de entrada invalidada y representa la expresión booleana W.Z. Las demás salidas se simplifican de forma similar. Tal vez usted desee comprobar por sí mismo que los términos de salida simplificados son los reflejados en la tabla 3.

Ahora todo lo que queda por hacer es construir el diagrama del circuito a partir de las 10 expresiones booleanas. Como cada entrada se utiliza tanto en su forma normal como en su forma negativa, es más fácil construir el circuito disponiendo ocho líneas en paralelo que representen ambas modalidades. Cada una de las 10 salidas se puede entonces formar bajando las dos o tres líneas adecuadas hasta sus respectivas puertas AND. Abajo consignamos el diagrama del circuito decodificador.

Ejercicio 6

1) Se ha de diseñar un codificador de tres entradas para crear una salida 1 para las entradas 011, 101, 110 o 111. La salida debe ser cero para todas las otras combinaciones de entradas.

a) Dibuje la tabla de verdad para el codificador

b) Produzca una expresión booleana para la salida y simplifíquela

c) Dibuje el circuito codificador

Tabla 1

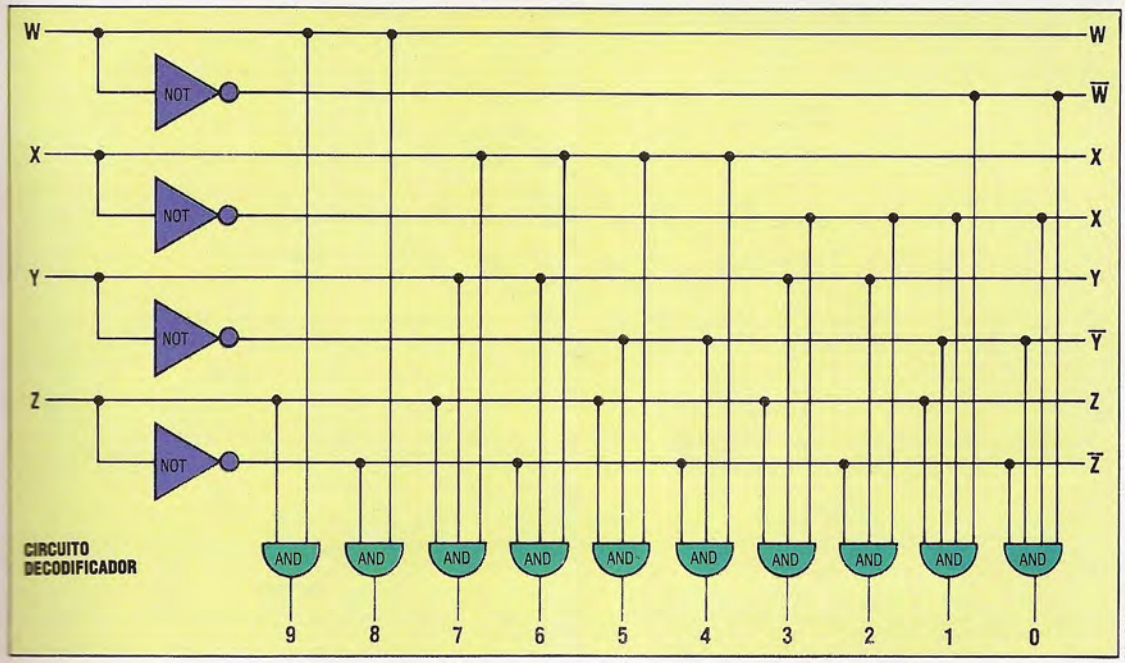
Entradas				Dígito BCD
W	X	Y	Z	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Tabla 2

Dígito BCD	Expresión booleana
0	$\overline{W}.\overline{X}.\overline{Y}.\overline{Z}$
1	$\overline{W}.\overline{X}.\overline{Y}.Z$
2	$\overline{W}.\overline{X}.Y.\overline{Z}$
3	$\overline{W}.\overline{X}.Y.Z$
4	$\overline{W}.X.\overline{Y}.\overline{Z}$
5	$\overline{W}.X.\overline{Y}.Z$
6	$\overline{W}.X.Y.\overline{Z}$
7	$\overline{W}.X.Y.Z$
8	$W.\overline{X}.\overline{Y}.\overline{Z}$
9	$W.\overline{X}.\overline{Y}.Z$

Tabla 3

Dígito BCD	Expresión booleana
0	$\overline{W}.\overline{X}.\overline{Y}.\overline{Z}$
1	$\overline{W}.\overline{X}.\overline{Y}.Z$
2	$\overline{X}.Y.\overline{Z}$
3	$\overline{X}.Y.Z$
4	$X.\overline{Y}.\overline{Z}$
5	$X.\overline{Y}.Z$
6	$X.Y.\overline{Z}$
7	$X.Y.Z$
8	$W.\overline{Z}$
9	$W.Z$



Liz Dixon

Bucles anidados

Los "bucles dentro de otros bucles" son tan útiles como delicados de tratar

Recibe el nombre de *bucle anidado* aquella parte de un diagrama en la que un bucle contiene uno o más ciclos. En este caso, las figuras nos ahorran explicaciones.

Ejemplo 1. En una posible nómina se representarán para cada uno de los trabajadores de una empresa los siguientes pasos: a) leer datos de identificación de los trabajadores; b) entrar los datos de las fichas diarias de cada trabajador; c) calcular la paga bruta, las retenciones y el líquido a percibir por el trabajo realizado; d) imprimir el correspondiente recibo (véase fig. 1).

Se observa que hay un bucle general que abarca todas las operaciones que se realizan para todos y cada uno de los trabajadores de la empresa. Mientras que hay otro, interno, que se encarga de la lectura de las fichas diarias relativas a un mismo trabajador individual, ya que cada uno de ellos pudo haber trabajado una cantidad variable de días. Para salir de este bucle interno, se efectúa la pregunta de si quedan más fichas de una misma persona; de ser así, entran los datos de una nueva ficha. Cuando aquéllas se han terminado, se continúa con el resto de las operaciones diagramadas hasta formular la pregunta: ¿es ésta la última nómina por realizar? En este caso, el bucle anidado dentro del general desempeña una misión de control y lectura de las fichas diarias. Repite el ciclo una cantidad variable de veces, que depende del número de fichas, o sea de los días trabajados por cada uno de los asalariados.

Figura 1

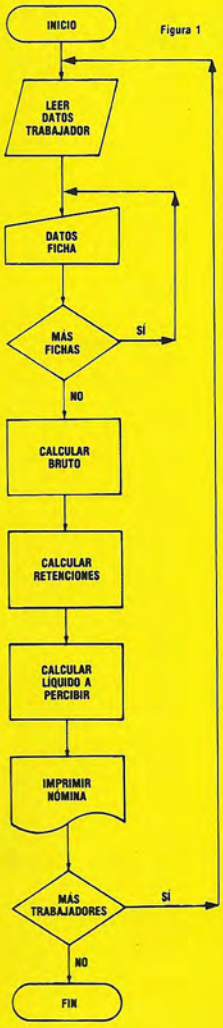
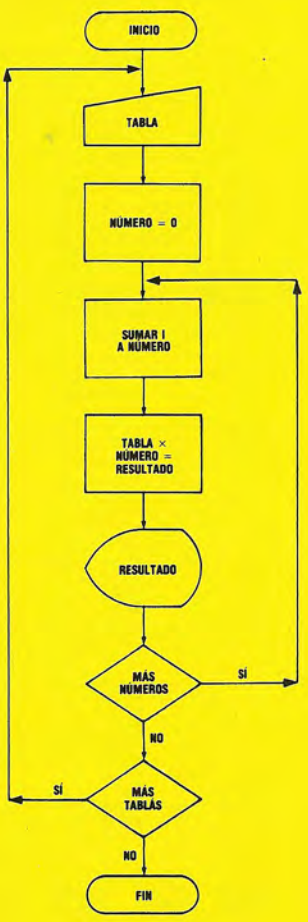


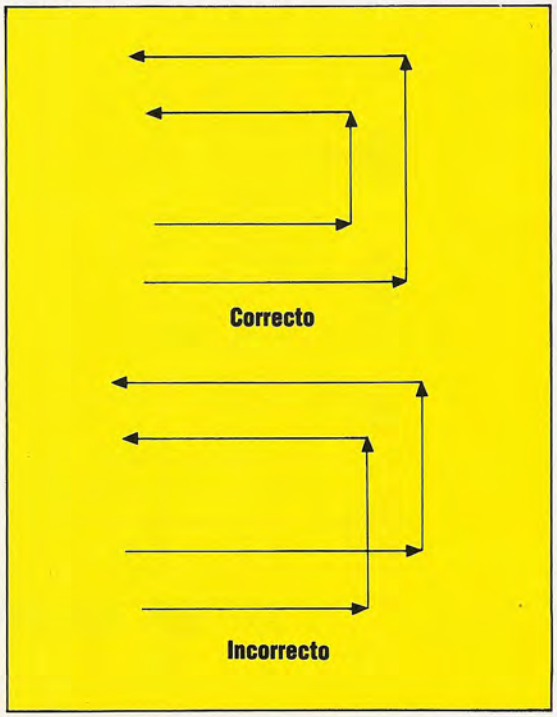
Figura 2



Ejemplo 2. Constituye una muestra de cómo las tablas de multiplicar ya explicadas anteriormente pueden ampliarse mediante el uso de dos bucles (véase fig. 2): el *general*, que pregunta si se desea o no visualizar una siguiente tabla tras imprimir la última, y el *interno*, anidado dentro de dicho bucle de tablas, que permite confeccionar la tabla con cuantos elementos factores queramos.

Algo que se olvida

Es importante observar que los bucles anidados son aquellos que contienen totalmente uno o más bucles, y que éstos jamás deben cruzarse.



La importancia de lo estándar

Japoneses y norteamericanos se han unido con el fin de producir el MSX, que intenta proporcionar una compatibilidad entre las diversas máquinas

El Spectravideo 318 es un ordenador personal de bajo costo cuya calidad es indudablemente la misma que la del BBC Micro y el Commodore 64. Incorpora un sintetizador de sonido a tres voces, gráficos con sprites de alta resolución, una palanca de mando y una puerta para cartuchos.

El teclado se aproxima mucho a las especificaciones MSX (véase p. 621). Es de diseño similar al del Sinclair Spectrum y posee teclas de relleno a presión y plásticas del mismo estilo.

El relleno del cursor se ha convertido en una palanca de mando incorporada. Un disco sustituye a las cuatro teclas usuales con flecha y el usuario puede situar el disco en el lugar apropiado para realizar desplazamientos hacia arriba, hacia abajo, a izquierda o derecha. También se puede encajar un mango de palanca de mando y generar las pulsaciones de teclas simplemente moviendo ésta. Se puede utilizar, además, la palanca de mando para posicionar el cursor sobre los errores mientras se corrige un programa.

El BASIC de Spectravideo es el último de una línea de intérpretes escrita por la empresa Microsoft. Esta versión se aproxima mucho al BASIC MSX, que es en sí mismo una ampliación del BASIC GW utilizado en máquinas como el IBM PC. Los programas en BASIC se desarrollan fácilmente con el editor de pantalla completo y facilidades generales tales como renumeración y numeración automática de líneas. A pesar de no contar con ninguna de las nuevas órdenes "estructuradas" de que disponen muchas de las modernas versiones de BASIC, como WHILE...WEND y REPEAT...UNTIL, esta versión sí posee IF...THEN...ELSE, que es necesaria para escribir programas pulcros y eficaces. El BASIC aprovecha al máximo los gráficos de Spectravideo.

La pantalla posee una resolución razonable de 256 × 192 puntos en 16 colores, aunque grupos pequeños de puntos deben compartir los mismos colores. Esto puede parecer pobre en comparación con máquinas como el BBC Micro, pero es su nivel de control sobre la pantalla lo que permite buenos

Ian McKinnell

El teclado del Spectravideo
El teclado del 318 se aproxima mucho a las especificaciones MSX. Se trata de un diseño al estilo del Spectrum, pero el espacio dispuesto entre las teclas y la calidad de éstas hacen que resulte perfectamente utilizable. Se trata de un teclado muy completo, con todas las teclas habituales, como Control, Escape, Tab y Backspace, más cinco de función (cada una de ellas con dos funciones), una de STOP, otra de SELECT y un conjunto de teclas para edición, como INS, DEL y COPY. Mediante la pulsación de una tecla de letra conjuntamente con la tecla LEFT GRPH o RIGHT GRPH se puede disponer de una serie de formas para gráficos, como bloques, puntos y líneas. La forma que produce cada una de las teclas está claramente indicada en el teclado

gráficos y no tan sólo sus especificaciones. El 318 posee todas las ampliaciones para gráficos del BASIC Microsoft, por ejemplo, órdenes individuales para dibujar puntos, líneas, recuadros, círculos, arcos y elipses. Una orden PAINT rellena cualquier forma cerrada con un color determinado. Si el usuario desea ejercer un mayor control, instrucciones especiales VPOKE y VPEEK leen y escriben directamente en la memoria de pantalla. También hay un minilenguaje para gráficos que se emplea con la orden DRAW para crear formas y dibujos complejos.

Por medio de GET se puede "capturar" cualquier rectángulo desde la pantalla y colocarlo en una matriz, y mediante PUT hacerlo reaparecer en ella. Estas órdenes hacen que resulte fácil producir patrones regulares, animación sencilla o efectos especiales tales como invertir una imagen. Por último, el 318 posee gráficos sprites. El chip de visualización 9929 les proporciona a los programadores el recurso de diseñar sus propias formas animadas, como personas, invasores del espacio o misiles.

La instrucción ON SPRITE GOSUB le permite al usuario preparar una "trampa para incidencias". En este caso, el programa continúa normalmente; pero si dos sprites colisionaran, el BASIC saltaría a una rutina especial que se ocupara de la colisión. Esto se podría utilizar para detectar un misil que hubiera acertado a una nave espacial, por ejemplo. Al hacer uso de este recurso de interrupción, no es necesario que el programador verifique continuamente todas las incidencias que se puedan producir. En consecuencia, la programación es más sencilla y el programa se ejecuta más rápidamente.

Un buen sonido es un requisito esencial, y el chip de sonido del Spectravideo posee tres voces y una gama de efectos especiales. Puede lograr resultados notables, a pesar de que es bastante difícil alcanzar los efectos más complejos mediante el BASIC. El sonido se reproduce a través del televisor, lo que permite controlar convenientemente el volumen.

El Spectravideo posee una pequeña selección de interfaces: dos puertas para palancas de mando, una puerta para cartuchos, una puerta para cassette y un conector de ampliación. Se encuentra a la venta una interesante gama de accesorios, pero tienden a ser bastante caros. Para conectar cualquier accesorio se necesita adquirir el miniampliador. Éste le permite al usuario agregar una opción extra, por lo general una ampliación de memoria de 16 o 64 Kbytes. La ampliación mayor se consigue mediante el superampliador, que permite la adición de hasta siete accesorios utilizando un sistema de ranuras similar al del Apple II. Se puede enchufar aquí más memoria, así como interfaces para impresora, unidades de disco y modems. Si es adicto a los juegos, puede optar por un accesorio muy interesante, el adaptador para juegos Coleco.

Aunque resulte extraño tratándose de un ordenador nuevo, el 318 exige una grabadora de cassette de la misma marca. Esta encarece el precio del sistema, si bien contribuye a proporcionarle una mayor fiabilidad y velocidad en cuanto a almacenamiento y recuperación de la información.

La palanca de mando Spectravideo
Una palanca de mando incorporada sustituye a las cuatro teclas con flecha con que cuenta la mayoría de los ordenadores. Sin el mango, uno puede colocar el disco en el punto apropiado para realizar desplazamientos hacia arriba, hacia abajo, a derecha o izquierda. Con el mango colocado, al mover la palanca de mando el disco gira hasta la posición adecuada. Además, la palanca permite efectuar movimientos en diagonal presionando en dos direcciones a la vez

Ian McKinnell

Puerta para cartuchos
Los cartuchos se enchufan a través de la parte superior de la carcasa en un conector firmemente montado

ROM
El BASIC del Spectravideo reside en dos ROM de 16 Kbytes

Conector de ampliación
Aquí se conecta una gama de cajas y opciones para ampliación

CPU
Como chip procesador se utiliza el popular Z80

Salida para monitor
Aquí se conecta un modulador separado para activar un aparato de televisión. Esto permite utilizar el Spectravideo con televisores que respondan a estándares distintos mediante la selección del modulador apropiado

Puerta para cassette
Para la grabadora de cassette exclusiva de Spectravideo se utiliza como interface un conector terminal

El hermano mayor
Una atractiva alternativa al 318 es el Spectravideo 328, versión más sofisticada del primero, con un teclado totalmente móvil, 80 Kbytes de RAM y un software para tratamiento de textos incorporado. Se pretende que sea un mejor punto de comienzo para aquellos usuarios que tengan la intención de pasar a un sistema de gestión completo

Palanca de mando incorporada
Un disco con un mango de palanca de mando sustituye al juego de teclas con flecha convencional

Controlador de E/S
Un chip 8255 se encarga de las interfaces, como las dos puertas para palanca de mando

Chip de sonido
El sonido a tres voces lo genera un chip 8910

Chip de visualización
Un disipador agregado a la ligera oculta un chip de visualización 9929

RAM
16 chips de RAM proporcionan 32 Kbytes de RAM

SPECTRAVIDEO 318

DIMENSIONES
410 x 220 x 80 mm

CPU
Z80

MEMORIA
32 K de RAM, de los cuales alrededor de 12 K están disponibles para programas en BASIC. 32 Kbytes de ROM

PANTALLA
24 filas de 40 columnas de texto, con gráficos de alta resolución de 256 x 192 a 16 colores y con facilidades para gráficos sprite. Para tareas de gestión se encuentra a la venta una opción de 80 columnas

INTERFACES
Conector de ampliación, puerta para cartuchos, dos puertas para palanca de mando

LENGUAJES DISPONIBLES
BASIC

TECLADO
Relleno plástico a presión, con teclas de función y edición y relleno para palanca de mando-cursor incorporado

DOCUMENTACION
Escasa y propensa a errores, pero la llegada de otras máquinas MSX habrá de ser un estímulo para la aparición de publicaciones independientes

VENTAJAS
Configuraciones MSX incluyendo un BASIC excelente, gráficos sprite, un teclado completo y palanca de mando incorporada. Capacidad de ampliación, en especial la posibilidad de convertirlo en un ordenador de oficina CP/M completo

DESVENTAJAS
Escasez de software. No tiene interface estándar para impresora. Requiere una grabadora de cassette exclusiva

Una de las configuraciones más atractivas del 318 en su capacidad de ampliación. Con la adición del superampliador, 64 Kbytes de RAM, la ficha de 80 columnas y una unidad de disco, el usuario puede disponer de un pequeño ordenador de oficina CP/M, con acceso al software para gestión profesional. Sin embargo, es difícil hallar programas que sean compatibles con el formato de disco Spectravideo. Esto seguirá siendo un problema mientras la máquina sea nueva y relativamente desconocida, pero se irá atenuando a medida que vaya haciéndose popular.

Estados financieros

Continuamos con nuestro estudio comparativo de tres paquetes de contabilidad, concentrándonos en sus facilidades para elaborar informes

Recuerde que los tres paquetes que estamos analizando son: *Cash trader*, de Quick Account; *Microledger*, de Lewis Ashley, y *Accountant*, de Compact Accounting Services. Además de registrar datos relativos a la compra y venta de bienes y servicios, los paquetes contables deben ser capaces de ofrecer una amplia gama de informes. Estos últimos se pueden dividir en dos categorías. Los informes de la gestión administrativa son para uso interno por parte de la empresa. Los informes financieros, o fiscales, están diseñados para que los lean organizaciones o grupos ajenos a la empresa.

Al igual que sucede con todos los sistemas computerizados, los datos básicos deben ser introducidos en el sistema antes de que se pueda elaborar con ellos algún informe. *Cash trader* y *Accountant*, por ejemplo, que no poseen facilidades para llevar archivos maestros de las cuentas de proveedores y clientes, no pueden producir los diversos informes normalmente relacionados con los sistemas de libros de compras y ventas. El *Microledger*, que posee un archivo maestro de los libros de compras y ventas, sí puede hacerlos.

El *Microledger* lleva un historial completo de transacciones para el período vigente de la cuenta de cada proveedor y cliente. También puede producir facturas que se enviarán a los clientes con el saldo adeudado y el detalle de las transacciones que lo motivan. Y puede producir avisos de remesa dineraria con destino a los proveedores (reflejando lo que se les paga). También puede proporcionar, a través de sus libros de compras y ventas, un índice de los títulos de las cuentas tanto por orden alfabético como numérico. A través del libro mayor, puede también servir una lista completa de las cuentas, un balance de comprobación y un impreso analítico ordenado según un código nominal o un código analítico, así como cada una de las cuentas.

Entre las opciones de salida impresa de que dispone el *Cash trader* figuran un resumen de los ingresos de la semana, un listado de todas las cuentas y un balance de comprobación. También puede producir extractos de la cuenta bancaria 0 de la caja, resúmenes trimestrales de las entradas o salidas del IVA, detalladas según un código IVA, y cuentas de balance.

El *Accountant* ofrece informes del libro mayor tales como un listado de las operaciones, una previsión de las cuentas (listando una cuenta determinada) y un balance de comprobación. Los informes del diario de ventas incluyen un informe agrupado de auditoría que refleja todas las transacciones pasadas para cada tipo de operación; un esquemático informe agrupado, que refleja los asientos en el libro mayor y en la cuenta de control del IVA; y un informe agrupado del IVA, que refleja los bienes y su impuesto detallados según la categoría del IVA. Los informes del diario de compras son similares, si bien lo que lucen es el IVA de entrada y no del IVA de salida, como es de suponer.

La relación de los deudores y acreedores constituye un buen ejemplo de informe administrativo para uso interno, laboriosísimo si se realiza a mano. Sin embargo, un sistema computerizado, con un programa razonablemente refinado, lo produce de forma automática con el mínimo esfuerzo adicional por parte del usuario. Se trata de un cuadro con las cantidades adeudadas o acreditadas, que se ordenan según vayan superando vencimientos de 30, 60 o 90 días. El programa lee los datos registrados comparándolos con las ventas o compras individuales y los clasifica por orden de fecha en una de las tres categorías. Cuando la cantidad se registra como pagada, se borra automáticamente de la lista de deudores y acreedores.

El informe del balance de comprobación es común a los tres sistemas. Éste imprime la descripción de cada cuenta y el total del debe y el haber actuales para cada una de ellas. Dado que los tres paquetes se basan en la contabilidad por partida doble, los pasivos y activos totales del sistema siempre deben ser iguales.

Del balance de comprobación, se pasa a elaborar el estado de pérdidas y ganancias y el balance de situación. El *Cash trader* simplifica su cuadro de pérdidas y ganancias restando sencillamente del total de los ingresos el total de los gastos. Esto puede ser suficiente para los pequeños comerciantes, pero no es un informe de pérdidas y ganancias completo y exhaustivo. Los otros dos paquetes acaban a las puertas de estos informes: llevan al usuario hasta la etapa del balance de comprobación.

No todos los informes analíticos se pueden producir con tan poco esfuerzo por parte del usuario como el balance de comprobación y el listado de deudores/acreedores. En realidad, estos informes, y algún que otro (como los listados de las operaciones) son simples "vuelcos de archivo" en los cuales a todo lo que contiene el archivo se le otorga un formato rudimentario más un título, y después se envía a la impresora. Hay cuadros mucho más selectos, cuyos criterios para la selección de datos presentan sus dificultades, con independencia del paquete que se esté utilizando.

Por ejemplo, tanto el *Accountant* como el *Microledger* poseen admirables facilidades para el análisis y la confección de presupuestos. Pero el usuario ha de poner bastante trabajo de su parte con el fin de explotar todas las ventajas que estos paquetes tienen en sus capacidades para elaborar informes de contabilidad analítica.

El *Accountant* permite que el usuario establezca

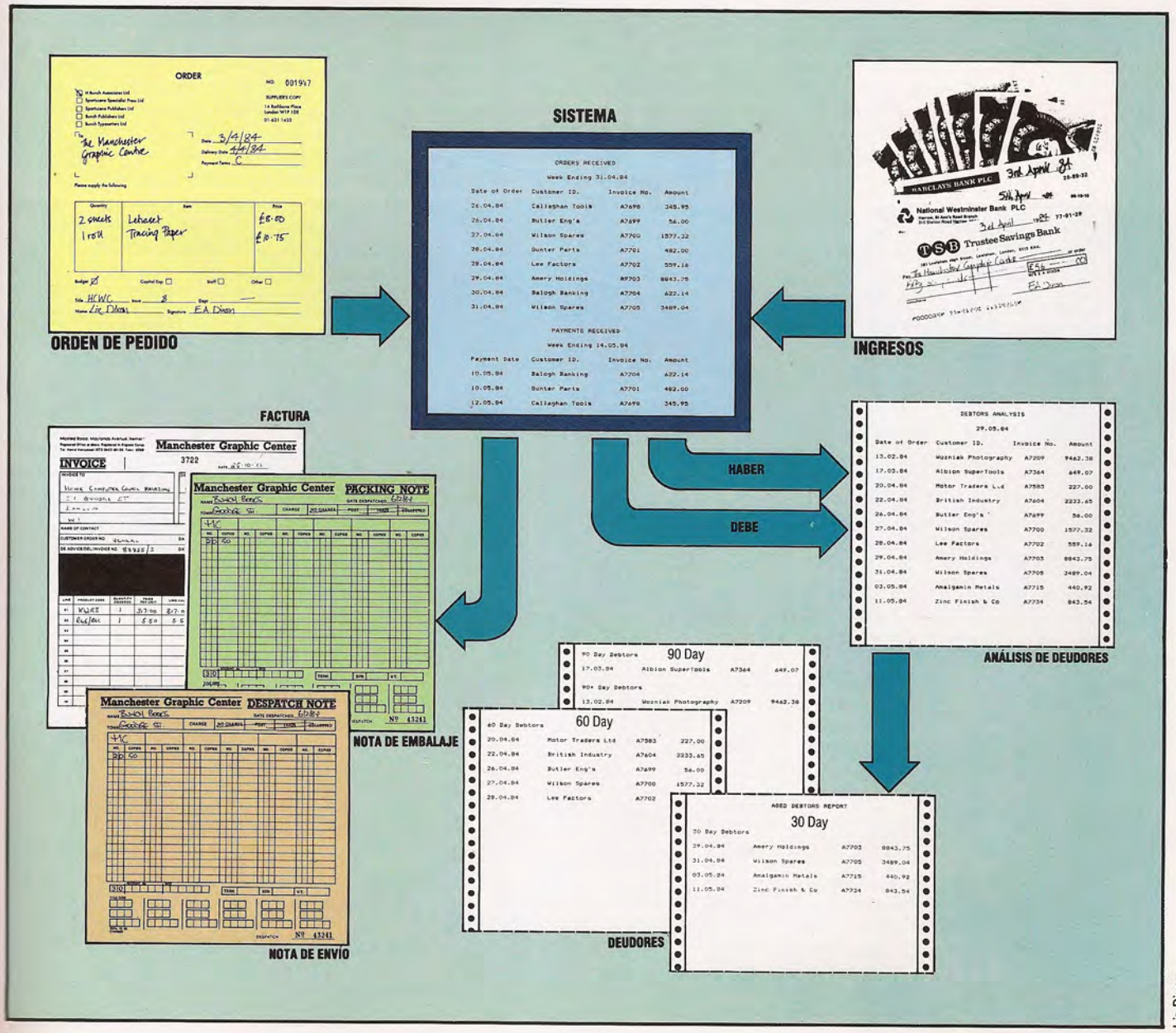
claves de grupos (o códigos) para analizar las cuentas individuales. Si, por ejemplo, usted lleva una empresa que consta de tres departamentos (supongamos, departamento de producción, de ventas y de almacén/compras), puede utilizar una clave de grupo para averiguar cuánto cuesta mantener cada departamento. Habrá costes comunes (y, por tanto, títulos de cuentas comunes) a los tres departamentos. Pero otorgando a los costes del departamento de compras la clave 01, al departamento de producción la 02 y al departamento de ventas la 03, usted puede identificar qué proporción de cada tipo de coste se debe asignar a cada uno.

Si se elige la opción "balance de comprobación formateado" del *Accountant*, obtendremos un balance de comprobación que clasifique las cuentas por su código analítico, y dé subtotales para cada departamento. Estos informes son muy valiosos para la toma de decisiones internas.

El *Microledger* posee una facilidad similar, sólo que en este caso el número del análisis está incorporado en el código de la cuenta. Recuerde que cada cuenta, ya sea ventas, compras o nominal, se identifica en este paquete mediante un número de tres dígitos. Existen dos niveles de análisis. El primer nivel corresponde al número de dos dígitos antepuesto al número de la cuenta. El segundo es un número de tres dígitos que se intercala entre el primer número de análisis y el código de la cuenta propiamente dicho (p. ej.: aa/bbb/111).

Para producir informes administrativos podemos servirnos de estos números para dos niveles de análisis de la siguiente manera: supongamos que en el libro mayor hay una sola cuenta de ingresos, pero la empresa posee cuatro vendedores. El primer número de análisis se podría utilizar para distinguir la cantidad de ingresos por ventas de las otras categorías de cuentas, mientras que los segundos tres dígitos serían más que suficientes para identificar las contribuciones efectuadas por los cuatro vendedores. Un informe que señalara los márgenes reportados para cada uno de los cuatro vendedores podría identificar a aquellos cuyo rendimiento estuviera por debajo de la media.

Informe inmediato
Un libro mayor computerizado ofrece un acceso casi instantáneo a una amplia gama de informes útiles. Aquí se muestra cómo el sistema compila información relativa a todas las operaciones y produce automáticamente una lista de los clientes y una clasificación según el vencimiento de su deuda. Un sistema más exigente dispondría de papeles preimpresos donde formularía incluso las facturas, los albaranes y los acuses de recibo



Un ángulo diferente

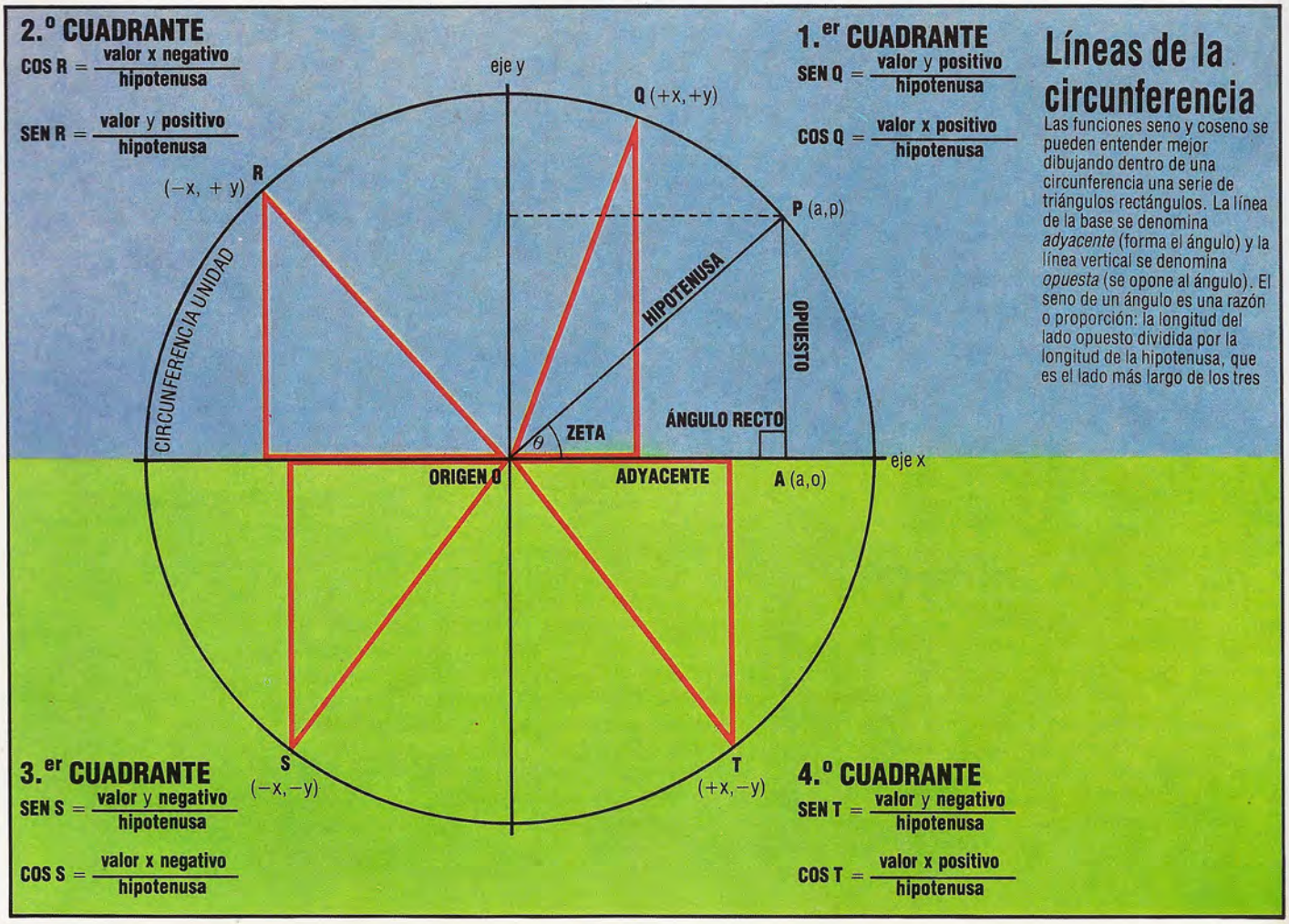
Las matemáticas son necesarias en numerosos programas. Si le atrae programar pero de esta disciplina sólo recuerda nebulosos conceptos, esta serie de artículos le será muy útil

¿Cuál es el tipo de matemáticas que realmente necesita usar el programador? Eso depende, y no es sorprendente, de la clase de programa que esté escribiendo. Las versiones de BASIC que se proporcionan con la mayoría de los ordenadores poseen numerosas sentencias y funciones incorporadas para manipular gráficos en pantalla (PLOT, CIRCLE, FILL, LINE, COLOUR, INK, PAPER, etc.), de modo que los problemas que supone trasladar y hacer rotar figuras sencillas en la pantalla no son muy graves. Aun así, existen ocasiones en las que se necesitarán las funciones trigonométricas de seno, coseno y tangente, para las que, afortunadamente, el BASIC dispone de un buen conjunto de facilidades.

Cuando se pasa a la estadística, sin embargo, el BASIC nos decepciona. La mayoría de las versiones del lenguaje no tienen funciones estadísticas incorporadas que ayuden a la manipulación de datos.

Si escribe guías para juegos o digitación en las que son importantes los tiempos de respuesta del usuario del programa, la mayoría de las versiones de BASIC volverán a decepcionarlo. Sencillamente, no le proporcionan al programador funciones de tiempo fiables. Éstas son las tres áreas principales (trigonometría, estadística y tiempos) que vamos a analizar en esta serie de capítulos dedicados a las matemáticas elementales.

Los estudiantes de bachillerato o formación profesional con frecuencia se preguntan cuál es la importancia que pueda tener la trigonometría en el mundo real. La respuesta considera el hecho de que la "trigo" (como la suelen llamar) es el nexo entre la geometría euclidiana, que trata con puntos, líneas y curvas, y el álgebra, que permite llegar a soluciones matemáticas con valores conocidos manipulando valores desconocidos, o incógnitas. To-



Líneas de la circunferencia

Las funciones seno y coseno se pueden entender mejor dibujando dentro de una circunferencia una serie de triángulos rectángulos. La línea de la base se denomina *adyacente* (forma el ángulo) y la línea vertical se denomina *opuesta* (se opone al ángulo). El seno de un ángulo es una razón o proporción: la longitud del lado opuesto dividida por la longitud de la hipotenusa, que es el lado más largo de los tres

Kevin Jones

memos la parábola, por ejemplo. Si trata de una curva que posee muchas propiedades: partiendo de ellas se pueden descubrir muchas cosas útiles "a mano", o sea, con papel cuadriculado, un transportador, una regla y un lápiz. Mucho más provechoso es, sin embargo, comprender cómo la curva puede corresponderse con la fórmula algebraica: $y = x^2$.

Esta sencilla fórmula nos permite calcular valores para cualquier punto de la curva sin tener que recurrir cada vez a dibujarla.

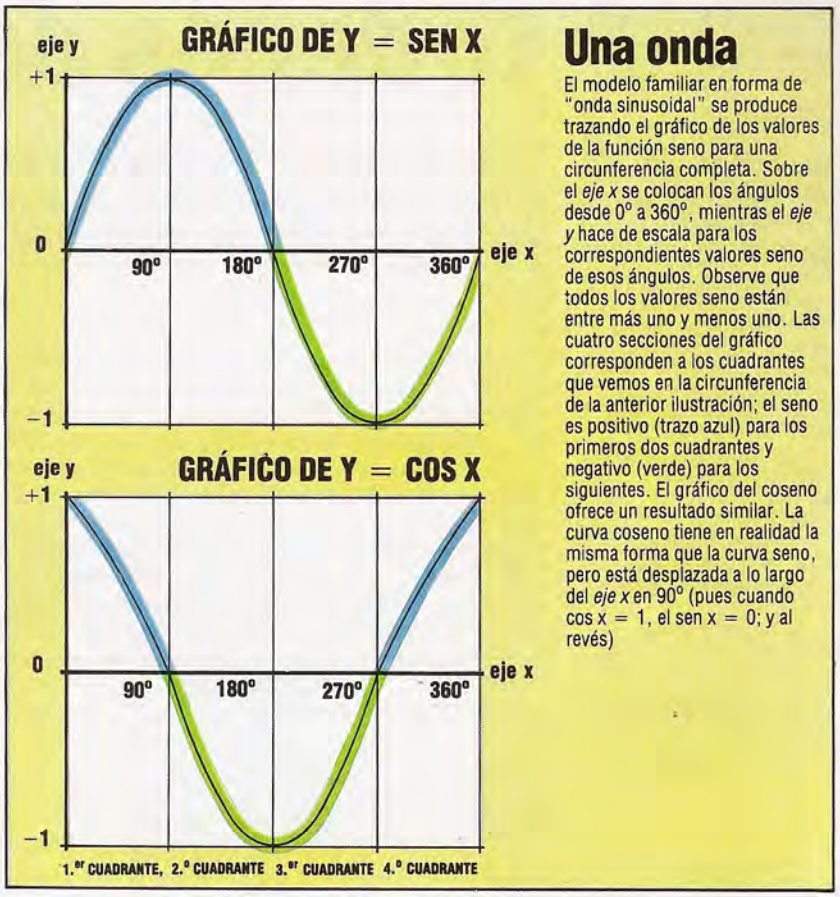
Los dos primeros capítulos de esta serie tienen como objetivo proporcionar una breve visión de los conceptos trigonométricos y ver cómo éstos se pueden fácilmente codificar en programas en BASIC. Empecemos por echar un vistazo a las funciones trigonométricas básicas *seno* y *coseno*.

Seno y coseno son dos formas de relacionar la razón de los lados en los triángulos rectángulos. El coseno y el seno también guardan una relación directa con la circunferencia. Todo triángulo rectángulo se puede trazar de modo que quepa dentro del área de una circunferencia denominada *unitaria*. Se llama de este modo porque su radio vale una "unidad"; no importa la unidad de lo que sea, porque lo que cuenta es la proporción o *razón* que ahora explicamos. En la ilustración de la izquierda vemos una línea que ha "girado" 40°. La posición inicial del giro es, por convención, el eje horizontal derecho, y la dirección del giro va "en sentido contrario a las agujas del reloj". El eje horizontal se denomina *eje de las x* y al ángulo de rotación lo llamaremos *zeta*, que se simboliza utilizando la letra griega θ. Si bajamos verticalmente un punto (p o Q o R...) de la circunferencia con una línea que llegue al *eje de las x* se obtiene un triángulo rectángulo.

El coseno de θ se define como la proporción que guarda la longitud del lado *adyacente* del triángulo (la parte que descansa a lo largo del *eje x*) con la *hipotenusa* (el radio de la circunferencia). Si usted se molesta en medirla, observará que la circunferencia del ejemplo tiene un radio de 5,22 cm y si ahora mide el lado adyacente del triángulo inscrito dentro de la misma, siendo el ángulo de 40°, veremos que vale aproximadamente 4 cm. Dividiendo 4 cm por 5,22 cm obtenemos la razón 0,76628, que es el valor aproximado del coseno para 40°. Si usted dispone de una calculadora que posea una tecla COS (también puede mirar las tablas trigonométricas, si sabe cómo hacerlo), pruebe dando entrada a 40 COS. Obtendrá como resultado 0,76604044. Esta razón se mantiene siempre, con independencia del tamaño de la circunferencia en la que esté inscrito el triángulo rectángulo.

En la circunferencia del ejemplo usted puede dibujar otros triángulos rectángulos con diferentes valores de θ. Comprobará entonces que por mucho o poco que valga θ, el valor de cos θ (cos es la abreviatura de coseno) nunca será mayor que 1 ni menor que 0, en valor absoluto. Usted mismo puede dar con la causa, con sólo pensar que si dividimos un valor menor por otro mayor el resultado nunca supera la unidad (note que el lado adyacente siempre es más pequeño que la hipotenusa).

Lo que sí puede pasar es que nos encontremos con un cos θ con valor negativo. Y esto se explica del siguiente modo: como puede ver, el giro que hace a θ más grande o más pequeño tiene por centro el llamado punto de origen 0, donde se encuentran los dos ejes *x* e *y*. Pues bien, los matemáticos



Una onda

El modelo familiar en forma de "onda sinusoidal" se produce trazando el gráfico de los valores de la función seno para una circunferencia completa. Sobre el *eje x* se colocan los ángulos desde 0° a 360°, mientras el *eje y* hace de escala para los correspondientes valores seno de esos ángulos. Observe que todos los valores seno están entre más uno y menos uno. Las cuatro secciones del gráfico corresponden a los cuadrantes que vemos en la circunferencia de la anterior ilustración; el seno es positivo (trazo azul) para los primeros dos cuadrantes y negativo (verde) para los siguientes. El gráfico del coseno ofrece un resultado similar. La curva coseno tiene en realidad la misma forma que la curva seno, pero está desplazada a lo largo del *eje x* en 90° (pues cuando cos x = 1, el sen x = 0; y al revés)

han convenido en que las mediciones que se hagan desde 0 a cualquier punto sobre el *eje x* a la derecha de 0 tengan signo positivo, y las longitudes a la izquierda sean negativas (para el *eje y*, hacia arriba tendremos los valores positivos y hacia abajo los negativos). Cuando θ rebasa los 90° (un ángulo recto), y se para, por ejemplo, en R, el lado adyacente, según lo establecido, mide −3,5 cm, valor negativo. La razón −3,5/5,22 es negativa y vale −0,670, que es lo que muy aproximadamente le daría su calculadora si digitara el coseno del ángulo 138,9°. Le resultará fácil percatarse, ayudado del dibujo, que todo ángulo entre 90° y 270° tiene coseno negativo. Por otro lado, todos los cosenos de ángulos entre 0° y 90° y entre 270° y 360° son positivos. Advierta que la hipotenusa (o radio) siempre es considerada por los especialistas en trigonometría como positiva.

Si comprendió bien lo anterior, le resultará claro cuanto le añadiremos sobre la función seno. Ésta se define como la razón lado opuesto/hipotenusa. Luego lo único que cambia es que ahora mirará el *eje y* (vertical) y no el *eje x*. Cuando θ se aproxima al grado cero, casi desaparece el lado opuesto, pero en compensación el adyacente vale tanto como la hipotenusa. Entonces, se entiende que para θ = 0° será cos θ = 1, pues lado adyacente = hipotenusa, y un valor dividido por el mismo valor siempre da la unidad. Al mismo tiempo, como el lado opuesto es cero, se sigue que sen 0° = 0. Sólo le falta verificar que para θ situado en el primero y segundo cuadrantes (valores entre 0° y 180°) el seno es positivo, mientras que los ángulos situados en el tercer y cuarto cuadrantes (valores entre 180° y 360°) tienen seno negativo: el *eje y* en este caso va hacia abajo.

Órdenes de comienzo

Una vez escrito el programa en lenguaje ensamblador, el programador debe cuidar de las "directrices ensambladoras", válidas para los dos procesadores

Dónde localizar las instrucciones en lenguaje máquina

BBC Micro

En la modalidad directa, dé entrada a:
PRINT.~PAGE
que proporciona la dirección hexa del comienzo del espacio reservado.
Después dé entrada a:
PAGE = PAGE + N
donde N, decimal, es el número de bytes que usted desea reservar

En el capítulo anterior escribimos un programa sencillo en lenguaje máquina que sumaba dos números en el acumulador y almacenaba el resultado en la memoria. No había nada asombroso en lo que hacía el programa; pero al escribirlo cubrimos muchos puntos importantes para el programador de lenguaje máquina. Volvamos ahora a mirar el programa, con las direcciones de posición incluidas, como si fuera a cargarse en $0000 y el resultado acumulado se fuera a almacenar en la dirección $0009. Las dos versiones del programa son:

Dirección de la posición	Lenguaje máquina	Lenguaje ensamblador
Z80		
0000	A7	AND A
0001	3E 42	LD A,$42
0003	CE 07	ADC A,$07
0005	32 09 00	LD BYTE1,A
0008	C9	RET
6502		
0000	18	CLC
0001	A9 42	LDA #$42
0003	69 07	ADC #$07
0005	8D 09 00	STA BYTE1
0008	60	RTS

Observe que la cuarta instrucción (que almacena el contenido del acumulador en $0009) de ambos programas no especifica ninguna dirección de destino en la columna de lenguaje ensamblador. En cambio, utiliza una dirección simbólica, BYTE 1. Pero en la versión de la instrucción de lenguaje máquina vemos el opcode 32 que significa "transferir lo contenido en el acumulador" seguido de 09 00, la forma *lo-hi* de los dos bytes de la dirección $0009.

Éste es otro aspecto del proceso de traducción (o ensamblaje) del lenguaje ensamblador al lenguaje máquina. Así como se usan expresiones mnemotécnicas razonablemente significativas para las instrucciones (STA y RET, p. ej., en lugar de códigos hexas como 8D y C9) porque gracias a ellas nos resulta más fácil escribir y leer los programas, del mismo modo con frecuencia utilizaremos símbolos como BYTE1 en lugar de direcciones hexas o números tan opacos como $0009. Esto se parece mucho a la inicialización de variables con constantes en un programa en BASIC, y el razonamiento es exactamente el mismo en ambos casos: el programa resulta más legible, se reduce la posibilidad de que se produzcan errores al escribir tales números y se consigue que el programa resulte mucho más fácil de modelar. Por ejemplo, si colocamos la sentencia donde a la variable se le asigna por primera vez el valor constante, haremos que el nuevo valor se utilice automáticamente a lo largo del programa, sin que precise una ulterior edición por nuestra parte.

Esto es fácil de comprender con un programa en BASIC; pero en nuestro programa en lenguaje ensamblador, ¿dónde está el equivalente de la sentencia LET BYTE1 = $0009? De momento esta instrucción no existe. Cuando lleguemos realmente a ensamblar el lenguaje máquina no debemos olvidar que esto se debe hacer por nosotros mismos. No obstante, si estuviéramos utilizando un programa ensamblador para que realizara el ensamblaje por nosotros, entonces podríamos efectuar una sentencia de asignación de este tipo al comienzo del programa (aquí proporcionamos la versión Z80 del programa, si bien estas directrices de ensamblador se podrían utilizar con ambos procesadores):

Z80		
0000		BYTE1 EQU $0009
0000	A7	AND A
0001	3E 42	LD A,$42
0003	CE 07	ADC A,$07
0005	32 09 00	LD BYTE1,A
0008	C9	RET

BYTE1 está colocado en una columna propia, denominada campo de etiqueta, del cual hablaremos más adelante. En el campo del opcode se utiliza una nueva expresión mnemónica (EQU, por *equate* —igualar— o sea, "establecerlo igual a..."); y en el campo del operando se proporciona el valor que se le ha de asignar a BYTE1 (en este caso, $0009).

Es importante observar que a pesar de que EQU aparece en el campo de opcodes y tiene un aspecto mnemónico, no pertenece a la mnemotécnica del lenguaje ensamblador, ni siquiera forma parte del conjunto de instrucciones del Z80 o del 6502. Expresiones mnemotécnicas de este tipo se conocen como *pseudoops* (pseudocódigos de operación) o *directrices de ensamblador*. EQU dice al programa ensamblador: "siempre que encuentres el símbolo alfanumérico que me precede (BYTE1, en este caso) debes reemplazarlo por el valor que me sigue ($0009, en este caso)". Recuerde que cuando utilizamos un programa ensamblador lo escribimos en este lenguaje, ya sea como un archivo en discocinta o directamente en el teclado, y después llamamos al programa ensamblador para que lo convierta en un programa en lenguaje máquina. La salida de un ensamblador generalmente es un listado completo en este lenguaje como los que hemos estado produciendo, más el programa en lenguaje máquina que se compone sencillamente de una serie de bytes hexas. El código de lenguaje máquina se puede guardar como un archivo para uso posterior o bien cargar inmediatamente en la memoria para su ejecución.

Al realizar el ensamblaje, por nosotros, se puede hacer que el ensamblador lleve a cabo otras tareas

que hemos venido efectuando de forma manual: adosar las direcciones de posición a cada una de las líneas del programa, por ejemplo. Otro pseudo-op, ORG, hace esto por nosotros. Se le agrega al programa de este modo:

Z80			
0000		ORG	$A000
A000		BYTE1 EQU	$0009
A000	A7	AND	A
A001	3E 42	LD	A,$42
A003	CE 07	ADC	A,$07
A005	32 09 00	LD	BYTE1,A
A008	C9	RET	

Observe que la dirección de posición que acompaña a la primera línea del programa es $0000, pero la dirección de la línea siguiente es $A000, lo que refleja el efecto de la sentencia ORG. Además, observe que en las líneas que contienen pseudo-ops no aparece ningún byte en lenguaje máquina, precisamente porque éstos no forman parte del programa y no se deben traducir a lenguaje máquina. Debido a que son configuraciones del programa ensamblador y no elementos del conjunto de instrucciones de la CPU, los pseudo-ops difieren de un programa ensamblador a otro. EQU, por ejemplo, en ocasiones se sustituye por "=" y ORG por ".=". Sin embargo, como el efecto es el mismo, nosotros vamos a seguir empleando ORG y EQU como si éstos estuvieran estandarizados.

Quizá a usted se le haya ocurrido, mientras leía acerca de las directrices para el ensamblador, que casi desde el comienzo de esta serie de lecciones hemos estado utilizando un pseudo-op; así, por ejemplo, "$", el indicador hexa. Éste no es nada más que una directriz que advierte al ensamblador de que lo que le sigue ha de tratarse como un número hexadecimal. Del mismo modo, "#", que hemos introducido en la lección anterior, es el indicador de "datos inmediatos", que significa que lo que va a continuación es una cantidad absoluta y no un indicador ni un símbolo. Si apuramos esta idea, en realidad podemos considerar al lenguaje ensamblador en sí mismo como una larga serie de pseudo-ops. La verdad es que no existe ningún impedimento para que usted se invente su propia mnemotécnica siempre que cada expresión mnemónica creada por usted se corresponda con una y sólo con una instrucción del lenguaje máquina. Un programa ensamblador muy popular para el Vic-20 hace eso: utiliza una versión no estandarizada del lenguaje assembly 6502, en parte para que se ajuste con el formato de la pantalla de 22 columnas del Vic.

En este curso nos ceñiremos, como hemos hecho hasta ahora, al empleo de las expresiones mnemotécnicas del lenguaje assembly habituales entre los fabricantes de chips, pero no viene mal recordar de cuando en cuando que eso que llamamos código de lenguaje máquina es simbólico. La CPU es indiferente a todo excepto a las indicaciones de voltaje de sus patillas de entrada-salida, de modo que cómo las describamos es sólo un asunto a convenir.

Habiendo terminado por el momento con los pseudo-ops, volvamos a inspeccionar nuestro programa para encontrar otros puntos interesantes. En particular, vamos a comparar las traducciones que hemos hecho aquí de las instrucciones LDA y LD A con sus traducciones en nuestros programas anteriores. Anteriormente habíamos escrito:

```
AD ?? ??    LDA $????      (6502)
3A ?? ??    LD A,($????)   (Z80)
```

que significa "cargar el acumulador con el contenido del byte situado en la dirección ????". La traducción del opcode "cargar el acumulador" es $AD (6502) y $3A (Z80). (Note las diferencias con la segunda línea del programa que estamos tratando.) Esta incorpora una instrucción muy parecida: "cargar el acumulador con el valor inmediato $42". Los opcodes ahora son $A9 y $3E, para el 6502 y Z80 respectivamente; ¿por qué distintos? Quizá usted ya lo haya deducido por sí mismo. A pesar de que estamos efectuando la misma clase de operación (transferir datos al acumulador) en ambos programas, la fuente de estos datos es diferente. Por consiguiente, para la CPU se trata de operaciones distintas y poseen opcodes diferentes.

En el primer caso deseamos transferir al acumulador un dato contenido en el byte cuya dirección se indica. Nada se dice acerca del contenido de dicho byte; a la CPU simplemente se le proporcionan instrucciones para que traslade este contenido al acumulador. La CPU decodifica los tres bytes en código de lenguaje máquina, AD ?? ?? y 3A ?? ??, los cuales vienen a decirle: "interprete los dos bytes que siguen a este opcode como la dirección absoluta donde se halla el dato".

En el segundo caso, el dato a cargar en el acumulador está, sin embargo, en el byte que sigue al opcode, de modo que la CPU decodifica los dos bytes en código de lenguaje máquina, A9 42 y 3E 42, los cuales le dicen ahora: "interprete el byte que sigue a este opcode como el dato mismo". Hay algo en el opcode (A9 o 3E) que le dice a la CPU que cargue el acumulador desde el siguiente byte. Dado que su contador del programa siempre contiene la dirección de la siguiente orden a ejecutar, la CPU puede calcular la dirección del byte fuente y después llevar a cabo una sencilla operación "cargar el acumulador desde un byte direccionado".

Esto refuerza la impresión de que la mayoría de las operaciones que lleva a cabo la CPU son procedimientos muy simples y nada complicados. Un buen número de sus operaciones (que representan la quinta parte de su repertorio completo) consisten en trasladar los datos contenidos en un byte direccionado de la memoria a alguno de sus registros internos. Ésa es la tarea de todas estas operaciones "primitivas", y todo cuanto distingue a una instrucción de otra es la forma en la que se presenta la dirección del byte fuente (véase p. 464).

Investigar en profundidad las microoperaciones de la CPU es sin duda confuso al principio, pero nos será utilísimo para unificar coherentemente los numerosos detalles que se verán más tarde. Detalles que no le son necesarios si todo lo que usted desea es escribir programas en lenguaje ensamblador para conseguir mayor velocidad y eficacia. Para esto no necesita más que captar la idea, aprenderse todas las instrucciones, y recibir unos cuantos consejos sobre programación. Pero si desea tener pleno conocimiento de lo que está haciendo, necesitará algo más que limitarse a agregar otro lenguaje de programación a sus conocimientos informáticos. Descubrirá no sólo cómo trabaja su procesador, sino que esta investigación hará que el aprendizaje de otros programas en lenguaje ensamblador resulte mucho más sencillo e interesante.

Spectrum

En la modalidad directa, dé entrada a:

```
LET RTOP = PEEK
(23730) + 256*PEEK (23731)
LET RTOP = RTOP-N
PRINT"RTOP =  ";RTOP
```

donde N es el número de bytes que desea reservar para su programa. Su espacio reservado empieza en 1 + RTOP

Commodore 64

Utilice el buffer de cassette desde el $033C hasta $03FB

Atención
Debe ajustar estos indicadores de memoria inmediatamente después de conectar su máquina cuando no haya programa en BASIC en la memoria

Ejercicio de ensamble

1) En el recuadro de la derecha damos la versión en lenguaje ensamblador de un programa sencillo. Ensamble el programa en lenguaje máquina y determine las direcciones de posición
2) ¿Qué instrucción falta en este programa?
3) ¿Cuál es el efecto del programa en los registros y en la RAM a que se refiere?
4) ¿Qué significan las palabras "dato inmediato"? ¿Qué otros tipos de datos podría haber?
5) Si se trata a BYTE1 como una dirección, ¿en qué página de la RAM aparece?

Nota: Los valores dados en este programa sólo son teóricos: si deseara ejecutarlo, debería escoger las posiciones y los valores adecuados para su máquina

Dirección de la posición	Lenguaje máquina	Lenguaje ensamblador		
6502				
		START	EQU	$A000
		BYTE1	EQU	$45
		BYTE2	EQU	$38
			ORG	START
			LDA	#BYTE1
			CLC	
			ADC	#BYTE1
			STA	BYTE1
			ADC	#BYTE2
			STA	BYTE2
Z80				
		START	EQU	$A000
		BYTE1	EQU	$45
		BYTE2	EQU	$38
			ORG	START
			LD	A,BYTE1
			AND	A
			ADC	A,BYTE1
			LD	(BYTE1),A
			ADC	A,BYTE2
			LD	(BYTE2),A

El efecto de las instrucciones en lenguaje máquina

Instrucción 1: CARGAR al acumulador (LOAD) el byte con DIRECCIÓN (ADDRESS)1
Instrucción Z80: LDA, (ADDR1)
Instrucción 6502: LDA ADDR1

Instrucción 2: SUMAR (ADD) el contenido del byte con DIRECCIÓN (ADDRESS) 2
Instrucción Z80: ADC A, (ADDR2)
Instrucción 6502: ADC ADDR2

Instrucción 3: ALMACENAR (STORE) el contenido del acumulador en la DIRECCIÓN (ADDRESS) 2
Instrucción Z80: LD (ADDR2),A
Instrucción 6502: STA ADDR2

El efecto que producen las instrucciones para transferencia de datos del tipo LDA ADDR1 o LD (ADDR2),A, es siempre el de copiar el contenido de la posición fuente en la posición de destino. Por lo tanto, este contenido de la posición se copia encima en la de destino; la posición fuente no se ve afectada por la transferencia de datos

La aportación de Oriente

SORD constituye una de las más innovadoras compañías japonesas incorporadas al mercado mundial de la informática

Cortesía de Sord

Takayoshii Shiina

Probablemente la más interesante de las firmas japonesas de ordenadores sea, también, una de las menos conocidas: SORD. La mayoría de los nombres nipones familiares al oído, de Hitachi a Sony, pertenecen a enormes empresas con miles de empleados e inmensos recursos, pero SORD es una firma pequeña que sólo cuenta con unos centenares de empleados.

En nombre SORD proviene de una combinación de SOftware y haRDware. Y se trata de una elección muy adecuada, porque la empresa siempre ha prestado la misma atención tanto al desarrollo de software como a las máquinas.

Los comienzos de la empresa fueron en cierto sentido vacilantes. Su presidente, Takayoshii Shiina, abandonó la universidad para unirse a Rikei Industries, una compañía de moderado éxito de segundo rango en la Bolsa de valores de Tokio en 1967. En la misma se dedicó a reorganizar la política de comercialización de la empresa.

Hacia 1970 Shiina y un amigo suyo ya estaban preparados para fundar su propia compañía y así se formó SORD, con un capital de 650 000 yens (unas 450 000 pesetas a la cotización actual). Sin embargo, Shiina siguió trabajando para Rikei hasta el mes de diciembre de aquel año. Los primeros productos de SORD fueron un tester lógico de precio económico y algunos trabajos de programación.

Para 1971, SORD estaba empezando a llevar una cantidad razonable de negocios, entonces relacionados principalmente con la escritura de software por encargo. Hacia 1973, SORD había empezado a fabricar y hacia finales de 1974 tenía una unidad de disco flexible importada que trabajaba con una interface desarrollada por SORD. Enseguida vino el SMP-80/20, uno de los primeros ordenadores japoneses basados en el 8080 de Intel.

EL SMP-80/20 fue un producto que tuvo mucho éxito y las órdenes de pedido empezaron a llover. Pero Shiina tenía ambiciosos planes de ampliación y en 1977 formó grupo al 20 % con Toppan, una de las mayores empresas editoras del Japón. La inyección de dinero que esto significó ayudó a SORD a desarrollar un considerable apoyo de software para su creciente lista de productos para ordenadores, y para 1981 había desarrollado el PIPS.

PIPS era un paquete de programas muy avanzado para su tiempo. Fue uno de los primeros ejemplos de software integrado que se produjeron en el mundo y, más que ningún otro factor, ayudó a consolidar la posición de SORD en el mercado. PIPS combina las funciones de una hoja electrónica, un procesador de textos y una base de datos en forma tal que incluso las personas que jamás hayan usado con anterioridad un ordenador pueden aprender a utilizarlo en unas pocas horas.

El extraordinario éxito que obtuvo PIPS en Japón sólo se puede comprender si lo enmarcamos en el contexto del mercado del ordenador doméstico de aquel entonces. En Japón era una práctica generalizada y aceptada que los ordenadores se vendieran sin software de apoyo.

En Japón, el tipo de software para aplicaciones que tan familiar era en Europa y Estados Unidos para contabilidad, facturación, etc., era virtualmente desconocido. Ello se debía a dos causas. En primer lugar, la renuencia japonesa a importar algo que ellos podía hacer por sí mismos significaba que al país estaba llegando muy poco software norteamericano. En segundo lugar, los japoneses no suelen sentirse atraídos por el aprendizaje de lenguas extranjeras, ocupando una posición, en cuanto a aptitud, sólo ligeramente superior a la de los británicos. Si un manual de Digital Research sobre CP/M, por ejemplo, le resulta misterioso a un británico, a un japonés es posible que le parezca totalmente incomprensible. Los altos precios del software para aplicaciones del mercado japonés deja-

PIPS (Pan Information Processing System) es el paquete de gestión exclusivo de SORD. Se trata de un programa de uso general que se puede preparar para manipular la mayoría de las tareas de gestión contable y empresarial, desde contabilidad hasta gráficos de presentación. Demuestra un curioso enfoque japonés de la comercialización, ya que sólo se puede conseguir de forma gratuita con ciertas máquinas SORD. El PIPS podría suponer un buen motivo para adquirir un SORD, pero también limita la popularidad de un producto útil y apasionante

ron abierta una brecha para que SORD la explotara. Así fue cómo la nueva firma comenzó a desarrollar la serie M200 de ordenadores basados en el Z80 y, con posterioridad, la serie M23. Con la serie M343 se satisficieron exigencias mayores y de multiusuario. El M5 se introdujo para surtir el mercado personal de juegos, y ahora tenemos la serie M68, que incorpora tanto procesadores Z80 como Motorola 68000, los pequeños ordenadores de oficina M243 y el M285, un ordenador de 32 bits que ejecuta software VAX-11 para aplicaciones de CAD (*Computer Aided Design*: diseño auxiliado por ordenador).

Ninguna otra compañía del mundo posee una gama de productos informáticos tan amplia en el mercado; además, cada uno de ellos viene con un conjunto de software de apoyo. En lo que SORD parecería fallar es en que define a sus productos en términos de "conectar y usar". Es decir, considera sus máquinas como sistemas a pleno funcionamiento, completos con hardware y software. Los usuarios de SORD, por lo general, no pueden elegir otros productos de software para sus máquinas.

La empresa tomó algunas medidas para rectificar esta situación al proporcionar el sistema operativo SB-80 como una opción para acompañar a su serie de ordenadores basados en el Z80 M23. El SB-80 equivale al sistema operativo CP/M 2.2 de Digital Research y permite utilizar software CP/M, por primera vez, en ordenadores SORD. Otro paso que ha dado SORD en esta dirección ha sido el poner a disposición del usuario el p-system UCSD (University of California at San Diego).

Para quienes deseen seguir siendo fieles al software de SORD, la empresa ofrece varias eficaces versiones de BASIC, su propio sistema operativo, PIPS, y un procesador de textos parecido al Wang.

Aparte de los ofrecimientos de CP/M y p-system, SORD ha tendido a cubrir el campo agotado por los fabricantes de miniordenadores como DEC, ofreciendo sistemas que para su funcionamiento dependen en gran medida de software casero.

Entretanto, SORD continúa presentando innovadores diseños de hardware pero no consigue, según afirman los críticos, apoyar a los clientes con la documentación adecuada para sus productos. Durante los dos últimos años SORD ha hecho esfuerzos gigantescos por proporcionar una buena documentación, tanto para el PIPS como para su procesador de textos.

Los próximos años probablemente serán críticos para SORD, enfrentada como está al poderío industrial de las firmas japonesas de ordenadores más grandes y de la IBM. Shiina cree fervientemente en la empresa a pequeña escala y en el individualismo, y ha conseguido hacer de SORD una empresa internacional. ¿Será capaz de mantenerse en los primeros puestos en este mundo tan competitivo como es hoy el del ordenador?

Las opciones SORD
Las máquinas más conocidas de SORD son su ordenador personal, el M5, y su máquina portátil de oficina M23P. El M23P estableció un nuevo estándar para los portátiles mediante la utilización de discos flexibles Sony y una LCD (visualización en cristal líquido) de 80 columnas

Ian McKinnell

La ley del más apto

Millones de microprocesadores controlan las más diversas aplicaciones, pero sólo dos diseños dominan el mercado: el Z80 y el 6502

Los ordenadores de un chip fueron casi un resultado accidental. En 1972 Datapoint pidió a Intel —la empresa fabricante de chips— que ideara uno capaz de reemplazar los numerosos chips TTL (transistor-transistor-lógico) que empleaban los terminales de ordenador de la época. La pieza que produjeron se llamó 8008. Podía procesar datos de a ocho bits por vez y habría sido el "reemplazo lógico" ideal para los terminales de Datapoint de no ser por un inconveniente: operaba con demasiada lentitud. A pesar de que Datapoint decidió no adoptarlo, ingenieros y aficionados pronto repararon en la capacidad potencial del 8008 como la CPU para un ordenador de uso general y así nació el ordenador para una mesa de oficina.

Como pronto se reparó en las limitaciones del 8008 en lo referente a velocidad y potencia, Intel se propuso diseñar un sustituto. El chip que desarrollaron, el 8080, se convirtió rápidamente en el rey del mercado.

Aproximadamente en la misma época en que Intel dio a conocer el 8080, sus competidores de Motorola lanzaron al mercado un microprocesador de ocho bits conocido como el 6800. Si bien los conceptos de diseño del 8080 y del 6800 eran bastante distintos, ambos microprocesadores resultaban igualmente potentes y adecuados a su función de servir de base al diseño de un microordenador.

Aunque el 8080 y el 6800 eran igualmente eficaces, un accidente histórico allanó el terreno para el fenomenal éxito de un tercer chip, el Z80. En 1974, Gary Kildall —actualmente presidente de Digital Research— produjo para Intel un sistema operativo de disco denominado CP/M. Ello permitió que los ordenadores basados en el 8080 pudieran utilizar las recién introducidas unidades de disco flexible Shugart. Intel rechazó el sistema operativo de Kildall porque consideró que el software existente bastaba para ser utilizado con los sistemas de ordenador de unidad principal estándar de la época.

No obstante, los ordenadores pequeños se volvían cada vez más populares y el CP/M facilitó en gran medida el tratamiento de archivos de dichos sistemas. Este hecho aseguró durante años el dominio del mercado por el 8080 y dejó relativamente al margen al Motorola 6800. Se hicieron varios intentos de crear sistemas operativos de disco parecidos para el 6800, pero todo el impulso se volcó en el 8080 y el 6800 quedó en segundo plano.

A medida que crecía el mercado de productos basados en el microprocesador, los fabricantes de chips luchaban por ofrecer nuevos diseños, pero siempre se encontraban con una barrera: la reticencia del mercado para aceptar lo nuevo a menos que supusiera ventajas considerables. Las inversiones en diseño de hardware y en producción de software también frenaron la adopción de cualquier nuevo microprocesador incompatible.

La elección vital
La mayoría de los micros personales eligen entre el procesador 6502 (como el BBC Micro) o el Z80 (p. ej., el Spectrum). El Dragon, una de las pocas máquinas que utiliza otros chips, incorpora el 6809

Una idea genial dio la oportunidad inesperada a un nuevo diseño de chip: el Z80. Zilog —un equipo de ingenieros de diseño que anteriormente habían trabajado para Intel en el 8080— se dio cuenta de que era posible ampliar el conjunto de instrucciones. En síntesis, no se habían aprovechado todas las combinaciones posibles de unos y ceros que el 8080 podía reconocer como instrucciones. Mediante el empleo de combinaciones binarias no utilizadas por el chip de Intel, Zilog logró diseñar un microprocesador capaz de funcionar igual que el 8080 cuando se le proporcionaban instrucciones específicas de éste pero que además podía ofrecer una considerable mejora en rendimiento. De este modo consiguieron crear un chip que utilizaba software escrito para el 8080.

Además de esta innovación, Zilog presentó otra importante ventaja comercial. Mientras que el chip de Intel dependía de un chip de generador de reloj

Chips en croquis

Los microprocesadores evolucionaron a partir de dos fuentes principales: los que surgieron de los microprocesadores originales de Intel y los que se desarrollaron a partir del chip 6800, de la empresa rival Motorola. Este esquema muestra el modo en que se desarrollaron los chips, así como algunas de las máquinas a las que fueron incorporados. Muchos de los chips poco conocidos aparecen en los micros menos populares. Es posible que el Apple III sea la única máquina de oficina que utiliza un procesador 6502. El Olivetti M20 es el único micro de uso general que utiliza un Z8000. En ambos casos, la elección de un microprocesador poco común y su posterior carencia de software han inhibido el éxito de la máquina. Algunos ordenadores de enorme éxito, como el IBM PC, logran popularizar por sí solos un chip

Intel

4004

8008

4004 y 8008: Diseñado para reemplazar grandes cantidades de circuitos integrados de TTL, el 4004 era un chip muy sencillo que sólo podía manipular datos en grupos de cuatro bits. Intel abordó rápidamente el procesamiento con ocho bits mediante el 8008. Aficionados e ingenieros reconocieron su potencial y comenzaron a crear sus propios ordenadores de "fabricación casera"

8080

8085

8080: Con la llegada del sistema operativo CP/M, éste se convirtió en el primer chip que podía emplearse para construir micros personales y de oficina. Al actualizar Intel el 8080, hació el 8085, compatible con el primero, con funciones adicionales y menos chips de soporte. El 8085 es asequible en una versión CMOS de baja potencia a menudo presente en máquinas portátiles como el Tandy Modelo 100

Tandy TRS-80 Modelo 100

8086

8088

8088 y 8086: El 8088 es una versión reducida que puede emplear chips de soporte más antiguos, razón por la que durante un tiempo fue el chip de 16 bits más conocido. Su utilización en el Sirius y en el IBM PC lo convirtió en el más popular de los chips de 16 bits. La mayoría de las máquinas utilizan ahora el 8086, de mayor rendimiento, pero plenamente compatible

IBM PC

Motorola

6800

6800: Es el rival del 8080, con capacidades semejantes pero una concepción de diseño totalmente distinta. Aparecieron dos escuelas: la de los que prefirieron el enfoque del 8080 de Intel y la de los que escogieron el modo de trabajar del 6800 de Motorola

6809

6809: El perfeccionamiento que Motorola hizo del 6800 se tradujo en el 6809, probablemente el más capaz de todos los chips de ocho bits. Sin embargo, apareció demasiado tarde para causar impacto y sólo ha sido utilizado en unas pocas máquinas

Dragon 32

68000

68000: El éxito del tan aclamado chip de 16 bits de Motorola se ha visto obstaculizado por la falta de software barato y el dominio del 8088. No obstante, Sinclair ha elegido la versión reducida del 68008 para su QL

68008

Sinclair QL

Apple Lisa

MOS Technology

6502

Commodore PET

6502: MOS Technology diseñó su propio chip de ocho bits que, pese a no ser compatible con el 6800, derivaba en gran parte de éste. Su bajo precio lo volvió muy atractivo para aficionados y diseñadores y por ello fue utilizado en la primera generación de máquinas tan vendidas como PET y Apple. Sigue siendo una elección popular para micros personales y ha sido empleado en ordenadores como el Oric y el BBC

Apple III

Zilog

Z80

Z80: Un equipo de diseñadores abandonó Intel y formó Zilog con el propósito de producir esta versión perfeccionada y totalmente compatible del 8080

Tandy TRS-80 Modelo 1

Z8000

Z8000: El primer chip de 16 bits de Zilog no ha sido popular en el mercado de ordenadores de uso general. Tal vez ello responda a un mal momento de salida y a la adopción del 8088 por IBM. El Z800 es el segundo intento de Zilog de producir un chip de 16 bits, que promete ser del todo compatible con el Z80 (y, por ende, con el 8080)

Olivetti M20

específico así como de un chip de control del sistema, el equipo Zilog logró combinar en un solo chip toda la lógica necesaria para un ordenador basado en un microprocesador. A pesar de que resultaba relativamente caro, el hecho de que pudiera reemplazar varios chips que cumplían otras funciones lo volvió muy atractivo para los fabricantes.

Aunque el 6800 había tenido poca suerte en comparación con el 8080, no por ello dejaba de ser popular entre algunos diseñadores y programadores. Más adelante Motorola diseñó un microprocesador de ocho bits altamente sofisticado, conocido como el 6809, que superaba al 6800. Lamentablemente, cuando el 6809 salió al mercado, una empresa rival llamada MOS Technology ya había presentado un nuevo perfeccionamiento del 6800 denominado 6502. Éste es el más popular de todos los procesadores incluidos dentro de lo que se conoce como la serie 6500. En dicha serie, la totalidad de sus integrantes utiliza el mismo conjunto de instrucciones, pero se diferencian unos de otros en cuanto a potencia y capacidades.

El 6502 de MOS Technology sigue un criterio de diseño de espíritu muy próximo al del 6800 de Motorola, pero no es compatible con éste ni en necesidades de hardware ni en compatibilidad de software. Por su parte, el Z80 incorpora todo el conjunto de instrucciones del 8080 y puede reemplazarlo en un sistema informático, si bien en este caso hay que realizar profundas modificaciones de diseño.

El 6502 ofrece un conjunto de instrucciones con el que cualquier programador del 6800 se sentiría a sus anchas, así como capacidades avanzadas y necesidades de interface ligeramente más sencillas. Sin embargo, no ofrece compatibilidad de software ni la posibilidad de un reemplazo chip por chip. Dados estos hechos, resultaría difícil imaginar que el 6502 gozara de su destacada posición actual de no ser por otro golpe de suerte: el 6502 fue utilizado en el ordenador Apple, que ha tenido un éxito sin precedentes.

Cuando apareció el Apple, los microordenadores de mesa estaban dominados por los diseños de bus basados en el S-100. Dichos micros se basaban en un circuito principal que transmitía potencia y señales a un circuito apropiado para cada función a cumplir. En consecuencia, un sistema S-100 mínimo requeriría una fuente de alimentación, un circuito principal, un circuito de CPU, otro de memoria, un tercero de VDU y probablemente un circuito para impresora y otro separado para unidades de disco. Es fácil ver lo costoso que sería un sistema S-100 en comparación con un sistema de un solo circuito, como el Apple.

Aunque el ordenador era relativamente barato, el principal adelanto de Steve Wozniak y su equipo de Apple tuvo lugar con un elemento de software de aplicaciones llamado VisiCalc. Este programa alcanzó gran popularidad entre los hombres de negocios, pues se dieron cuenta de que podían utilizarlo para generar presupuestos financieros con más rapidez y facilidad que si acudieran a la calculadora, lápiz y papel. VisiCalc tuvo tanto éxito que dio pie a que Apple vendiera masivamente su ordenador y de este modo el 6502 se estableció como uno de los principales diseños de microprocesador. Commodore también optó por el 6502 para el PET y sus sucesores.

Todavía se dio en Gran Bretaña un nuevo paso adelante cuando Acorn produjo el BBC Micro, que también se basa en este chip. Originalmente se había determinado que incorporara un Z80, pero ningún fabricante británico logró presentar un diseño adecuado en el plazo fijado.

Los chips cuentan
Los chips sofisticados reducen la cantidad de chips necesarios en un circuito. Cuando Apple mejoró el Apple II, la nueva versión IIe contaba con la mitad de chips principales

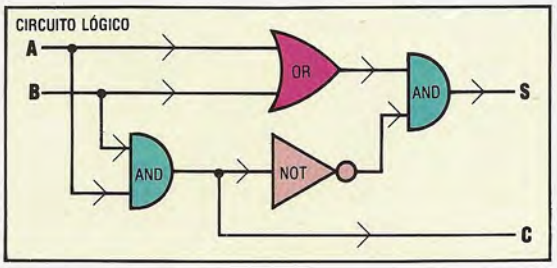
Mientras el chip 6502 establecía su dominio en el diseño de ordenadores de ocho bits, en el mercado comenzaron a aparecer ordenadores de 16 bits. Intel ofreció para estos ordenadores el 8088 y el 8086, al tiempo que Motorola producía el 68000 y Zilog el Z8000. Aunque estos tres diseños de 16 bits tienen sus méritos, ninguno es compatible con sus antecesores de ocho bits. Afortunadamente para Intel, Digital Research y Microsoft se apresuraron a ofrecer sistemas operativos para el 8086/8088 (CP/M-86 y MS-DOS, respectivamente), al tiempo que Zilog y Motorola fueron muy mal atendidos por el ramo del software. El hecho de que IBM adoptara el 8088 en su ordenador PC también ha dado un nuevo estímulo al chip de Intel.

La lucha por el dominio del mercado entre los chips de 16 bits promete ser una repetición de la historia del chip de ocho bits. Al igual que el Z80 y el 6502, el 8086 de Intel (y su versión reducida, el 8088) se han estandarizado. Entre las principales razones que dan cuenta de este hecho figuran el soporte de software de los sistemas operativos MS-DOS y CP/M-86 y su elección por los micros más vendidos, sobre todo el IBM y el Sirius. El chip Z8000 de Zilog sólo ha sido utilizado en un micro de uso general: el Olivetti M20. La Olivetti luchó por proporcionar software a la máquina y finalmente lanzó una ficha con un 8086 para que pudiera utilizar software MS-DOS y CP/M-86. Desde entonces, Zilog se ha propuesto diseñar un nuevo chip, el Z800, que no sólo cuenta con 16 bits, sino que puede utilizar software basado en el procesador Z80.

A pesar del creciente y rápido desarrollo en el campo de los chips de 16 bits, la mayoría de los ordenadores que se venden actualmente se basan en los diseños de ocho bits del Z80 y del 6502. Indudablemente, los ordenadores de 16 bits ofrecen ventajas de velocidad y potencia con respecto a sus antecesores, si bien en vista de la ingente cantidad de software que ya se ha desarrollado, las máquinas de ocho bits aún cuentan con una vida prolongada.

Medidas a medias

En esta ocasión emplearemos dos circuitos integrados para construir un sumador incompleto



Liz Dixon

Las puertas lógicas individuales que realizamos en el capítulo anterior son la base de circuitos digitales más complejos. Uno de esos grupos de puertas lógicas es el sumador incompleto, que analizamos en una lección de *Ciencia informática* (véase p. 513). Este circuito se emplea para sumar dos únicos bits. El sumador incompleto utiliza dos entradas, los bits sumandos, y ofrece dos salidas, la suma, más un bit de arrastre. La tabla de verdad que lo representa es la siguiente:

A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

La salida S es la suma de los dos bits de entrada. Cuando ambos bits valen uno, la suma en binario es 10. Este resultado no puede representarse con un único bit de salida, ya que ésta se desborda en un segundo bit. Dicho desbordamiento es el bit de arrastre.

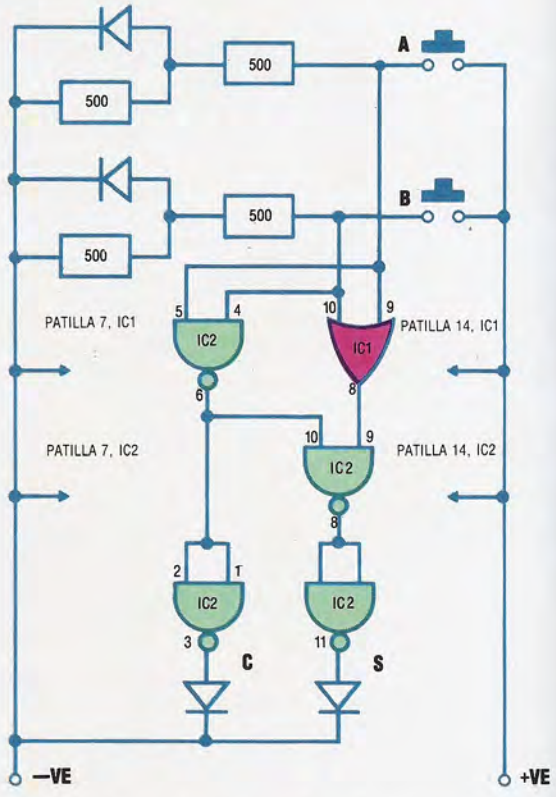
El sumador incompleto no resulta de mucha utilidad en los ordenadores de ocho bits; en realidad, lo que hace falta es un circuito que sume dos palabras de ocho bits. Dicho circuito puede construirse tomando como base 16 sumadores incompletos. Los dos primeros bits se suman mediante el primer sumador incompleto y el bit de la suma forma el primer bit del resultado. El bit de arrastre se añade al resultado de la segunda suma, el arrastre de dicha suma al tercero y así sucesivamente, enlazándolos de este modo.

Un sumador incompleto necesita alrededor de 10 transistores en las puertas que ya hemos construido. Sin embargo, las puertas lógicas AND, NAND, OR, NOR y otras son asequibles a muy bajo costo en grupos de cuatro en circuitos integrados simples. A partir de ellos, este sumador puede construirse más simplemente.

El circuito lógico de un sumador es el que mostramos a continuación. Se trata de la forma más sencilla de circuito. Utiliza tres tipos de puertas lógicas: OR, AND y NOT. Puesto que cada uno de los circuitos integrados que emplearemos sólo contiene un único tipo de puerta, el circuito lógico ha sido simplificado para utilizar menos puertas distintas. El circuito que construiremos utiliza cuatro puertas NAND y una sola puerta OR. El número de circuitos integrados se ha reducido a dos. Este circuito es más complejo que el circuito de una sola puerta que construimos en la página 624; será preciso, pues, prestar especial atención para cerciorarse de que todos los componentes quedan correctamente situados en el tablero de montaje de prueba.

En cuanto haya terminado de construir este circuito, es posible que piense que es demasiado el esfuerzo para resultados tan simples. Aunque es mucho más fácil que construir el circuito a partir de componentes separados como son los transistores,

CIRCUITO ELECTRÓNICO



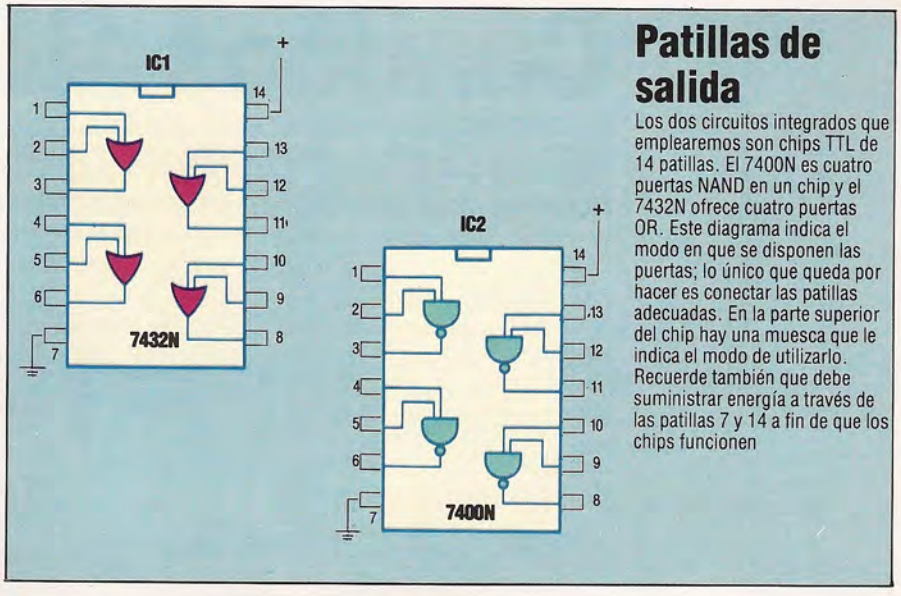
resulta difícil imaginar un ordenador entero construido de esta forma.

En la práctica, los chips rara vez se emplean de este modo y sólo ocasionalmente aparecen en un ángulo del circuito realizando una tarea de menor importancia. De los chips mayores salen y entran más señales, de manera que el chip en su totalidad es un dispositivo completo capaz, por ejemplo, de sumar dos números de cuatro bits.

El nivel de complejidad crece hasta tal punto que determinados chips son capaces de realizar por sí mismos tareas completas.

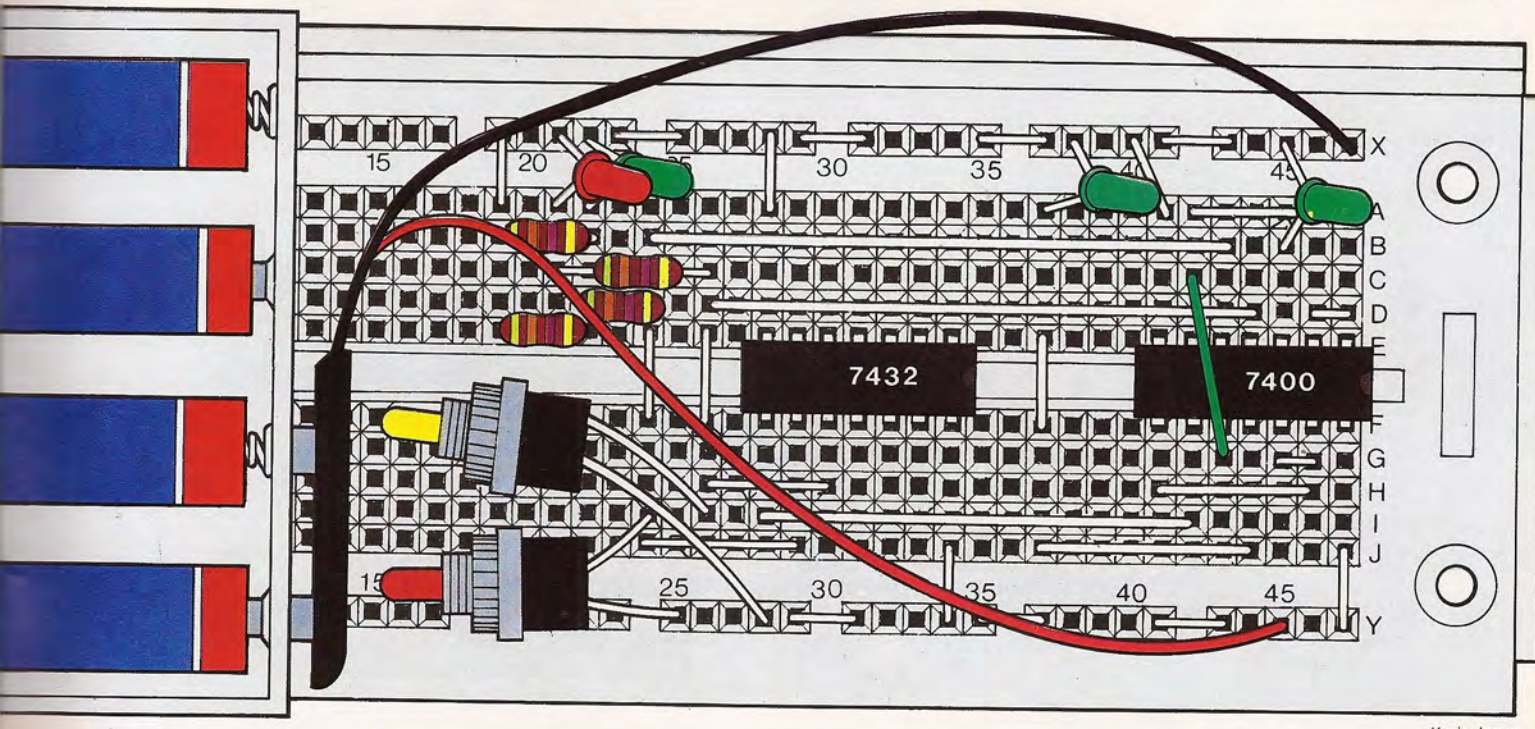
Materiales

4 resistencias de 500 ohmios, de 1/4 de vatio
4 diodos emisores de luz (LED)
2 conmutadores de funcionamiento por presión
1 circuito integrado 7400N
1 circuito integrado 7432N
4 pilas HP7 o equivalentes
1 soporte de pilas
1 conector para pilas
1 tablero de montaje de prueba
Trozos cortos de cable



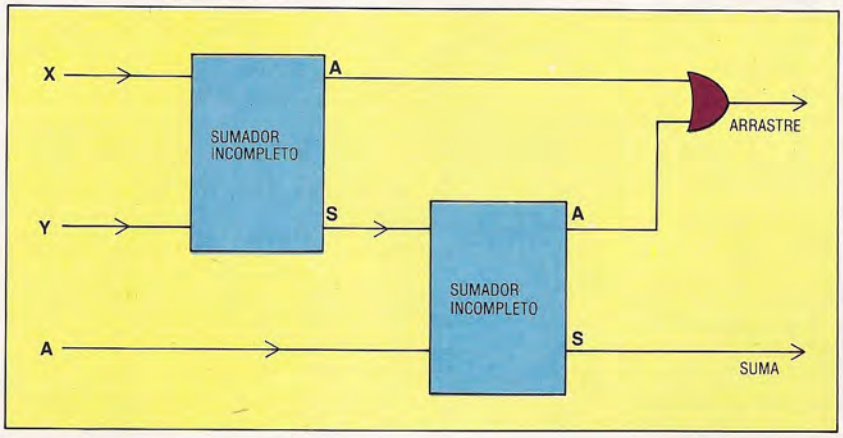
Patillas de salida

Los dos circuitos integrados que emplearemos son chips TTL de 14 patillas. El 7400N es cuatro puertas NAND en un chip y el 7432N ofrece cuatro puertas OR. Este diagrama indica el modo en que se disponen las puertas; lo único que queda por hacer es conectar las patillas adecuadas. En la parte superior del chip hay una muesca que le indica el modo de utilizarlo. Recuerde también que debe suministrar energía a través de las patillas 7 y 14 a fin de que los chips funcionen



Kevin Jones

A través del circuito
Una vez diseñado el circuito electrónico, el paso siguiente consiste en acomodar los componentes en el tablero de montaje de prueba. Puede adquirir tableros prefabricados o, simplemente, utilizar una fotocopia en un circuito vacío. Conviene que el tablero se parezca al máximo al circuito original ya que, cuanto más claro sea el diseño, más fácil resultará su construcción. Cópielo con exactitud pues todos los componentes están en la posición correcta



Sumador completo
¿Quiere ampliar como ejercicio su sumador incompleto convirtiéndolo en un sumador completo? Dicho circuito no sólo suma dos bits, sino también el arrastre de cualquier posición anterior de bit. Una serie de sumadores completos puede sumar palabras binarias completas. El modo más simple de crear un sumador completo consiste en construir dos sumadores incompletos como los que aquí se muestran. La señal suma del primer sumador incompleto debe reemplazar uno de los conmutadores de entrada del segundo sumador incompleto. La salida de arrastre del primer sumador incompleto debe ser pasada por OR junto con la del segundo para producir la señal del LED de arrastre

Caminos alternativos

En este capítulo del curso de lógica introducimos dos puertas nuevas que nos abren una ruta diferente para diseñar circuitos

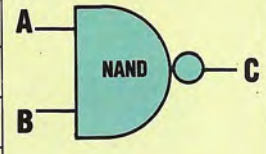
Si es posible resolver todos los problemas lógicos utilizando puertas AND, OR y NOT, ¿para qué complicarnos la vida estudiando otros tipos de puertas? Porque con estas nuevas puertas podemos reducir el costo de fabricación del circuito. Es posible resolver todos los problemas lógicos utilizando alguna de las siguientes técnicas:

a) puertas AND, OR y NOT juntas
b) sólo puertas NAND
c) sólo puertas NOR
d) una combinación de todas ellas

Analicemos esos dos nuevos tipos de puertas. Como ocurre con todo circuito y sus elementos, la función de cada puerta queda mejor descrita por su tabla de verdad.

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

LA PUERTA NAND



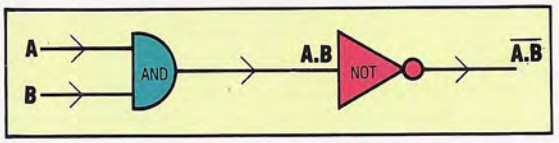
NAND es la abreviación de No AND y, si comparamos esta tabla de verdad con la de una puerta AND (véase p. 488), podemos ver que en la columna de salida todos los unos han sido cambiados por ceros y viceversa.

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

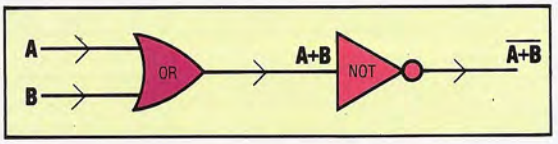
LA PUERTA NOR

De manera semejante, NOR es apócope de No OR y la comparación de las columnas de salida de esta tabla con las de la tabla de una puerta OR (véase p. 488) vuelve a mostrar que todos los unos y los ceros han sido invertidos.

En álgebra booleana no existen símbolos especiales para las operaciones NAND y NOR, si bien podemos representar cada función utilizando los símbolos AND, OR y NOT que ya conocemos. Una puerta NAND equivale a este simple circuito:



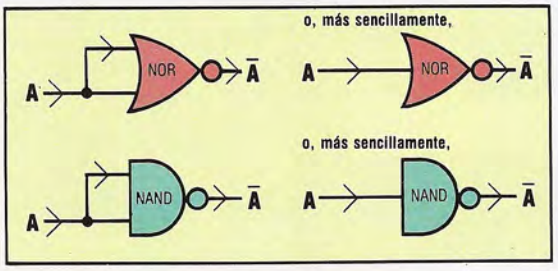
y la puerta NOR es equivalente a una puerta OR seguida de una puerta NOT:



El empleo de NAND y NOR

Del mismo modo que es posible dibujar circuitos AND/OR/NOT que equivalen a NAND y NOR, también podemos representar cada una de estas tres puertas básicas mediante una serie de puertas NOR o de una serie de puertas NAND.

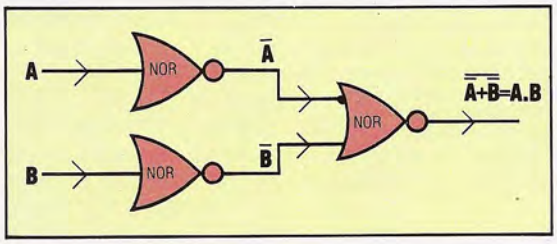
Puertas NOT: La anulación puede conseguirse conectando ambas entradas, ya sea utilizando una puerta NOR o una puerta NAND:



Puertas AND: En términos de álgebra booleana, la salida de una puerta AND cuyas entradas son A y B es A.B. No obstante, podemos manipular esta expresión hasta darle una forma más útil.

$$A.B = \overline{\overline{A}.\overline{B}} \qquad \text{(puesto que } A = \overline{\overline{A}}\text{)}$$
$$= \overline{A + B} \qquad \text{(ley de Morgan)}$$

Así, es posible hacer el circuito poniendo NOT(A) y NOT(B) a través de una puerta NOR:

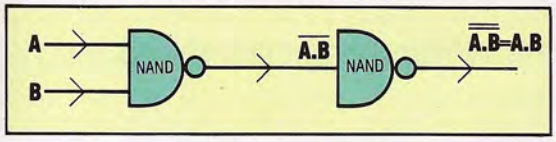


También es posible crear una puerta AND utilizando puertas NAND. La salida de una puerta NAND

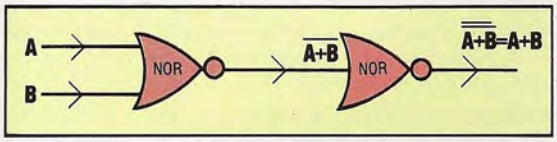
es $\overline{A.B}$. Si esta salida es invertida, obtendremos:

$$\overline{\overline{A.B}} = A.B$$

En consecuencia, el circuito será:



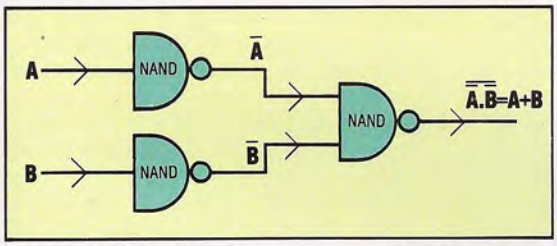
Puertas OR: Del mismo modo que enlazar dos puertas NAND equivale a una puerta AND, si enlazamos dos puertas NOR obtendremos un circuito que es equivalente a una puerta OR:



La salida exigida de una puerta OR es A + B. Recurriendo a las reglas del álgebra booleana, podemos manipularla hasta darle forma característica de una puerta NAND:

$$A + B = \overline{\overline{A} + \overline{B}}$$
$$= \overline{A.B}$$

y, por lo tanto, el circuito correspondiente con puertas NAND es:

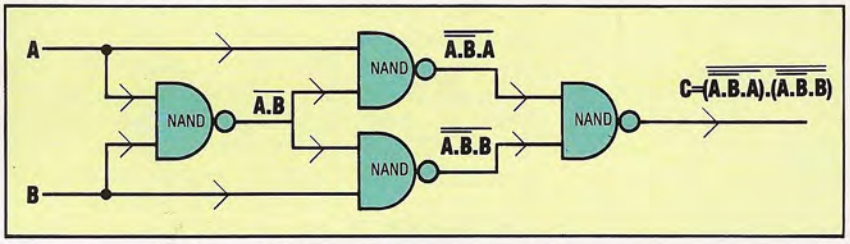


Si queremos construir un circuito utilizando únicamente elementos NAND o NOR, podemos seguir los métodos de simplificación ya conocidos, pero primero debemos manipular la expresión de álgebra booleana definitiva hasta darle la forma adecuada. Para circuitos que incorporan puertas NAND, aplicamos las reglas del álgebra booleana hasta crear una expresión que se compone de grupos de AND conectados por OR y utilizamos las leyes de Morgan tantas veces como sea necesario hasta que la expresión quede totalmente en forma de NAND. Para circuitos en forma de NOR, empleamos las mismas reglas, según muestra el ejemplo. Para ver cómo se utilizan dichas reglas, hagamos un repaso de la puerta OR Exclusiva (XOR), que aparece en la página 527. La salida de una puerta XOR puede definirse mediante la expresión $C = \overline{A.B}.(A + B)$.

Tomemos esta expresión y convirtámosla de modo que podamos construir un circuito para la puerta XOR exclusivamente con puertas NAND. En primer lugar, manipularemos la expresión hasta obtener grupos de AND conectados por OR.

$$C = \overline{A.B}.(A + B)$$
$$= (\overline{A.B}.A) + (\overline{A.B}.B) \text{ (prop. distributiva)}$$
$$= \overline{(\overline{A.B}.A)}.\overline{(\overline{A.B}.B)} \text{ (ley de Morgan)}$$

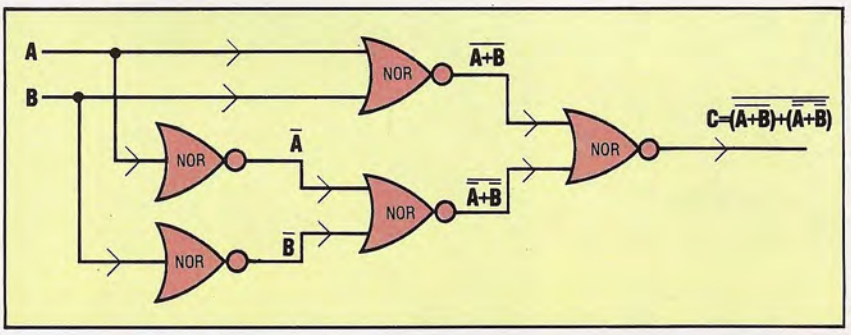
Cuando se dibuja el circuito de una expresión tan complicada como ésta, conviene comenzar por la salida y retroceder hasta las entradas. Procure seguir este diagrama de circuito desde la salida para ver cómo se construyó.



En cuanto a la forma NOR, debemos comenzar una vez más con la expresión original simplificada de la puerta XOR y manipularla formando grupos de OR conectados por AND. Este primer paso puede darse aplicando la ley de Morgan a la parte izquierda de la expresión:

$$C = \overline{A.B}.(A + B)$$
$$= (\overline{A} + \overline{B}).(A + B)$$
$$= \overline{(\overline{A} + \overline{B})} + \overline{(\overline{A} + \overline{B})}$$

También en este caso, esta expresión se convierte mejor en un diagrama de circuito comenzando por la salida y retrocediendo.



Solución al ejercicio 6 de la p. 627

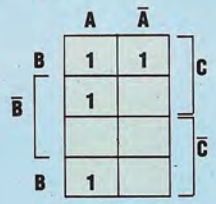
1a)

Entradas			Salida
A	B	C	P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

b) La expresión booleana de P es:

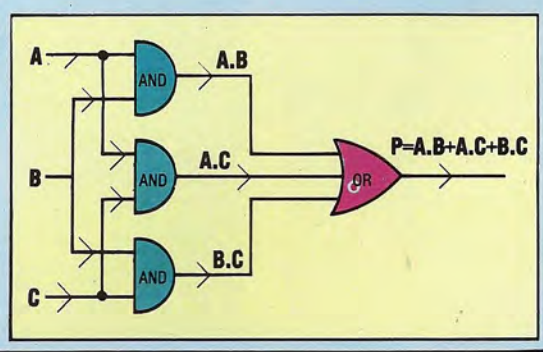
$$P = \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C} + A.B.C$$

Puede simplificarse utilizando un diagrama de Karnaugh:



Así, $P = A.B + A.C + B.C$

c) El circuito es:

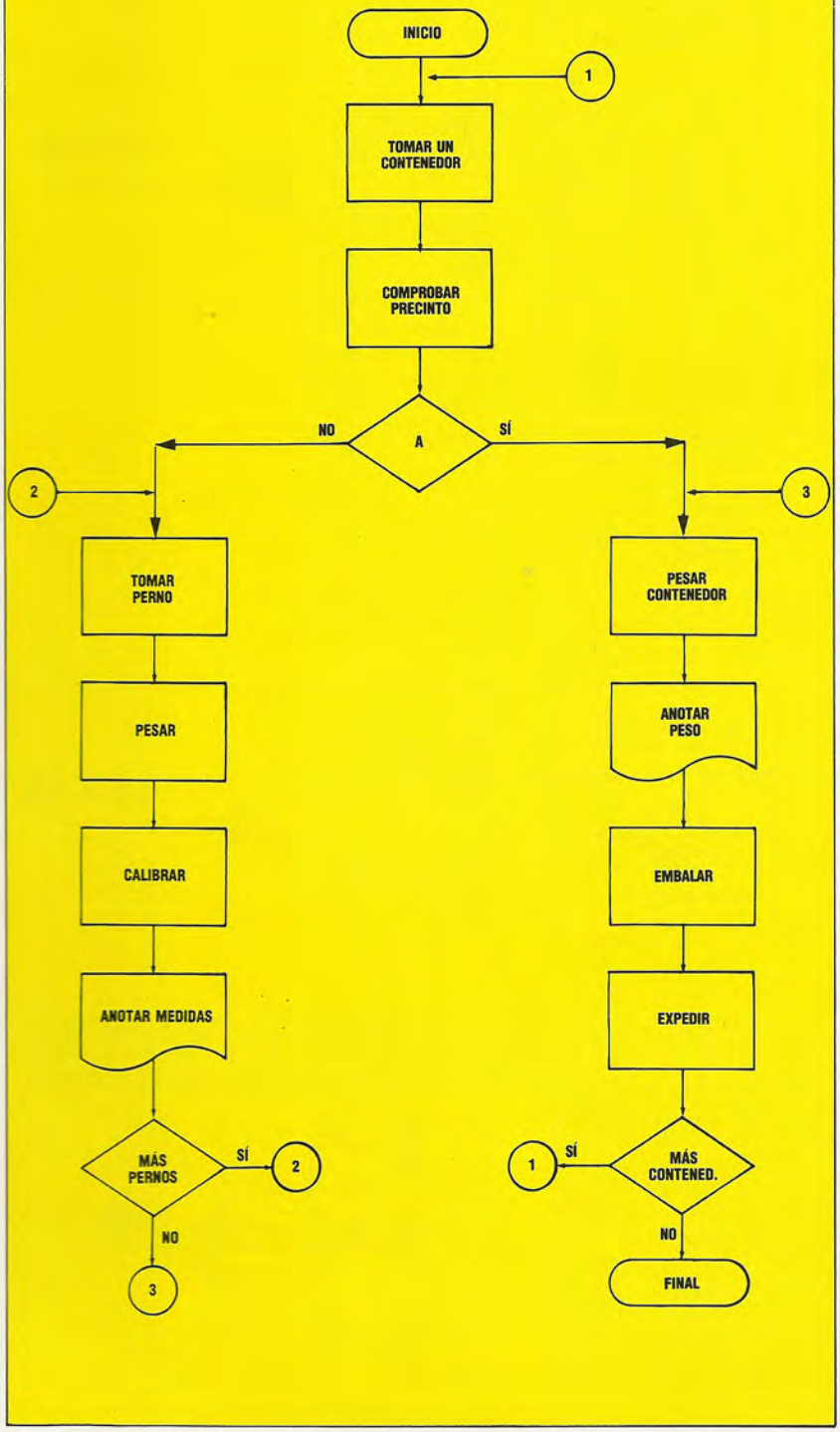


Liz Dixon

Conectores

Hay dos símbolos que, en los diagramas largos y complejos, resultan imprescindibles

Figura 1



Estos símbolos se denominan *conector de partes* y *conector de páginas*. Un ejemplo ilustrará el uso del primero: un operario, situado al final de una línea de producción, recibe dos clases de contenedores —A y B—, con un número indeterminado de pernos. Su cometido en la cadena se reduce a lo siguiente: tomará un contenedor y, luego de comprobar si pertenece a la clase A, debe pesarlo, anotar su peso, embalarlo y expedirlo. En cambio, si el contenedor es de clase B, tomará un perno, lo pesará, calibrará y anotará sus medidas. Así ha de continuar, un perno tras otro, hasta agotar su número, llegado el último de los cuales pesará el contenedor, anotará su peso, lo embalará y, tras expedirlo, proseguirá con el siguiente.

Obsérvese en el ejemplo la utilización de los símbolos conectores de partes (fig. 1) del diagrama. Éstos realizan la misma función que las líneas del flujo, con la ventaja de que, conforme el ordinograma va creciendo, el número de líneas empleadas aumenta, pudiéndose dar el caso de que la profusión de éstas, con sus posibles cruces, llegue a inducir a error. Mediante estos pequeños círculos, puede accederse igualmente al punto marcado por el conector de entrada, cuya contraseña interior (una letra o número, preferentemente) coincide con la figura colocada en el conector de salida. El conector que marca la entrada a la secuencia debe ser único, es decir, no pueden existir dos conectores de entrada con una misma clave. Pero no hay límite para el número de conectores de salida.

Equiparando el caso al de las líneas de flujo, se pueden utilizar tantos conectores como sea conveniente en un mismo diagrama, y es posible que se alternen indistintamente con líneas de flujo en el mencionado ordinograma, pues no existe regla alguna que lo impida.

El conector de páginas (fig. 2), en cambio, es de carácter obligatorio siempre que un mismo ordinograma ocupe más de una hoja, figurando como contraseña el número de la página a que se refiere.

Llegados a este punto, puede comprobarse la gran ventaja, en cuanto a comodidad y claridad se refiere, que representa utilizar un conector en lugar de una línea de flujo para conectar puntos del diagrama representados en diferentes páginas.

Figura 2

1. Conector de partes **2. Conector de páginas**

Pizarra electrónica

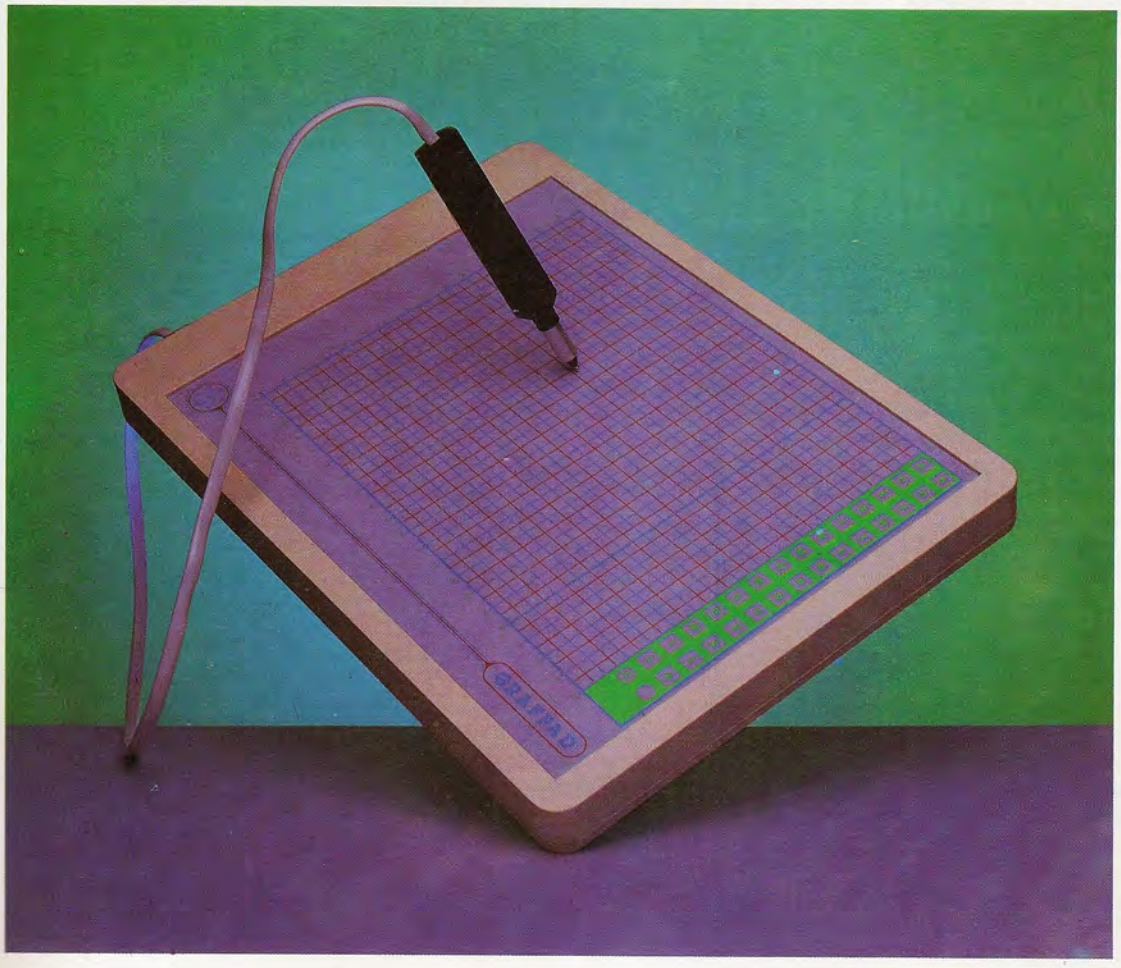
El Grafpad es un tablero de gráficos que permite crear detallados diseños y dibujos en un microordenador

Los tableros de gráficos, o digitalizadores, constituyen uno de los periféricos más polifacéticos y útiles de los microordenadores. Es evidente su uso como ayuda al dibujo y al diseño, desde el dibujo artístico al diseño de circuitos electrónicos y el trazado de mapas. Además de sus aplicaciones directas para el dibujo, ofrecen un útil dispositivo adicional de entrada. Una ficha colocada sobre el tablero gráfico puede tener todas las características de un programa, ya sea en palabras o gráficamente. Basta con tocar la orden apropiada con el cursor (o lápiz), y el software expondrá la opción escogida.

Estos sistemas solían ser coto cerrado de máquinas específicas, exclusivamente al alcance de diseñadores e ingenieros. Sin embargo, los precios han disminuido lo bastante para que los usuarios de ordenadores personales puedan probar por sí mismos este accesorio. La Grafpad, que aquí analizamos, es uno de los principales digitalizadores de bajo costo y ofrece buenas especificaciones por un precio razonable. Existen versiones exclusivas para el BBC Micro, Commodore 64 y el Spectrum de Sinclair. La que ilustramos corresponde a la versión para el BBC.

El Grafpad consta de tres elementos: la tablilla propiamente dicha, un cursor conectado y el software de control. La tablilla se conecta con el BBC a través de la puerta para usuario y el cursor se conecta a su lado. La superficie de la tablilla se presenta cuadriculada con 16 por 20 casillas y tiene una barra de mando (un listón aparte en el que se han inscrito letras individuales). La barra de mando puede utilizarse para controlar parte del software sin necesidad de recurrir a un teclado. Encima de ésta se extiende una cubierta de plexiglás que protege la superficie de la tablilla. Es posible diseñar los propios segmentos de programas (*overlays*) mediante las órdenes y rejillas que usted desee.

Dentro de la tablilla hay una rejilla de 320 por 256 cables, separados aproximadamente por una distancia de 1,2 mm. La punta del cursor es un minúsculo interruptor. Cuando se presiona con el cur-

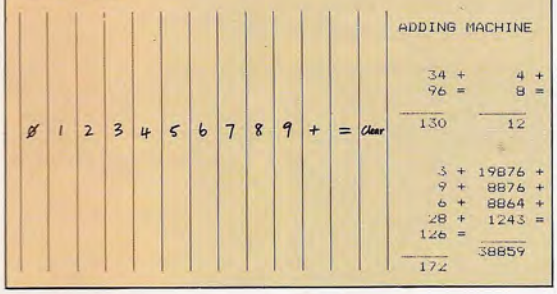


Ideas gráficas
El Grafpad puede utilizarse con su propio software para crear diseños y dibujos o con sus programas como dispositivo de entrada

Máquina sumadora

He aquí la versión inglesa de un programa en el BBC Micro que utiliza el Grafpad como dispositivo de entrada para una sumadora. Bajo la cubierta de la tablilla se sitúa un segmento de programa con las claves de la sumadora. Al tocar la clave pertinente con el cursor, el software se pondrá en operación

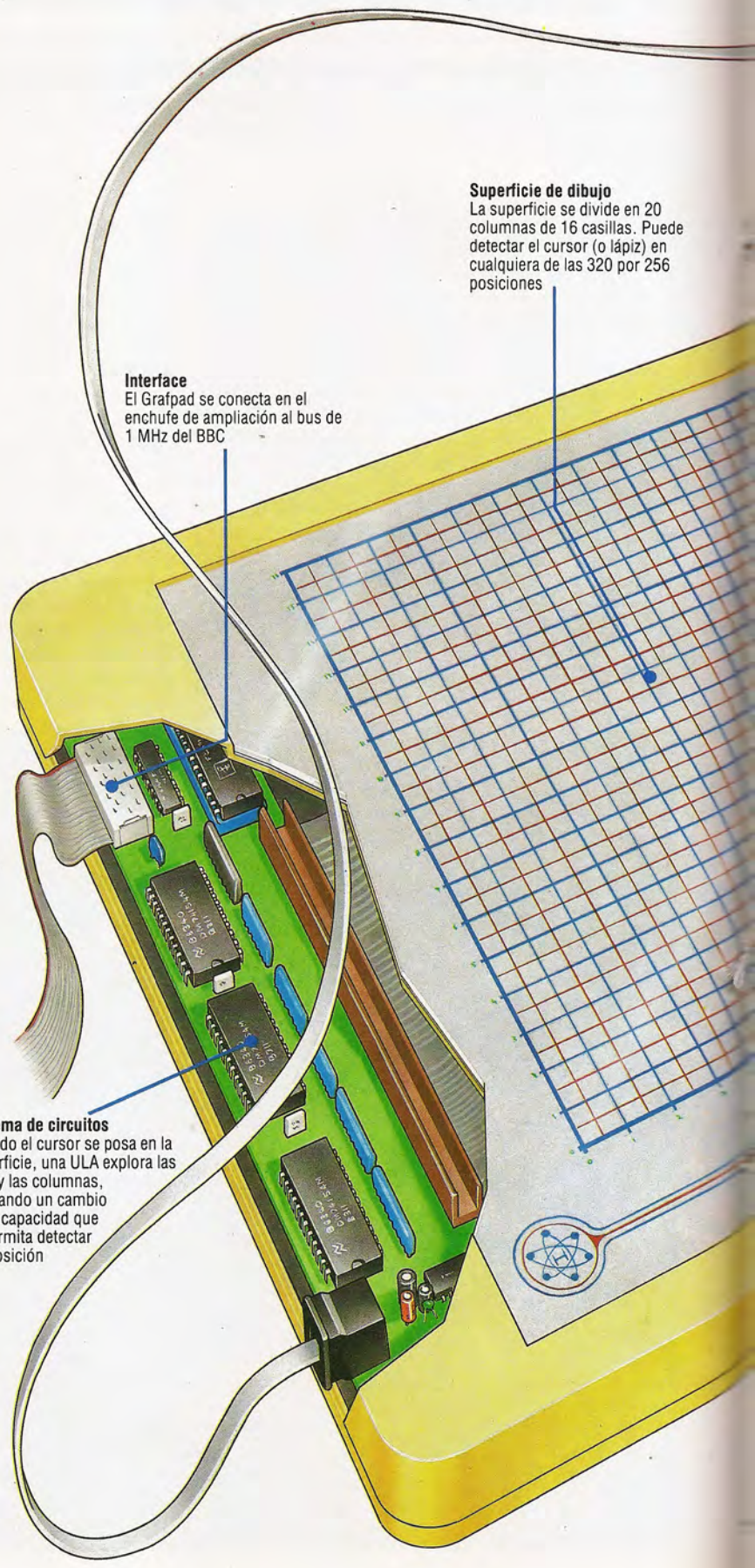
```
 10 REM GRAFPAD DEMO
 20 REM
 30 REM An adding machine using the Grafpad
 40 REM
 50 MODE 1
 60 PRINT "ADDING MACHINE":PRINT
 70 HIMEM=&29FF
 80 XADJUST=7
 90 REM
100 REM set up co-ordinate table
110 REM
120 DIM B(15):FOR I=1 TO 15:B(I)=I*25:NEXT I
130 R1=0:R=0
140 REM
150 REM load Grafpad driver program
160 REM
170 *LOAD "PADREAD" 2A00
180 PEN%=&2A00
190 REM
200 REM MAIN PROGRAM STARTS HERE
210 REM
220 REM wait for pen to be pressed
230 CALL PEN%
240 !X%=!X%-XADJUST:IF !X%<0 THEN !X%=0
250 IF ?U%>0 THEN 230
260 REM beep to register pen press
270 SOUND 1,-15,120,1
280 X=!X%
290 REM convert position to 0-12
300 I=0
310 IF X>B(I) THEN I=I+1:GOTO 310
320 REM interpret codes 0-12
330 IF I<10 THEN R=R*10+I
340 IF I=10 THEN PRINT R;" +":R1=R1+R:R=0
350 IF I=11 THEN PRINT R;" =":PRINT "          ":R1=
R1+R:PRINT R1:R1=0:R=0:PRINT:PRINT
360 IF I=12 THEN PRINT:PRINT"CLEAR":PRINT:R=0:R1=0
370 REM wait until pen is lifted again
380 CALL PEN%
390 IF ?U%=0 THEN 380
400 REM loop for next pen press
410 GOTO 230
```

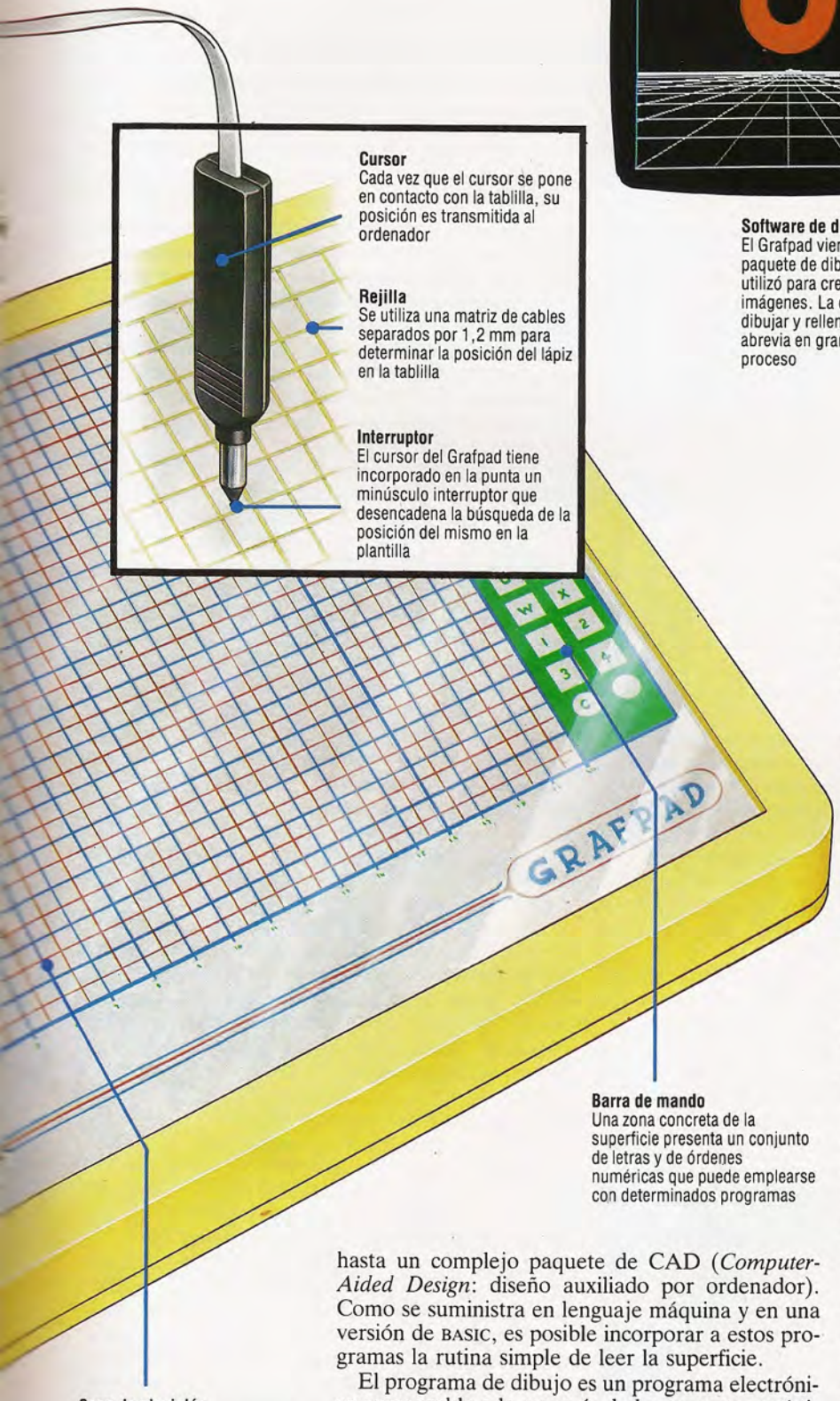


sor la cubierta de plexiglás de la tablilla, un chip ULA (*Uncommitted Logic Array*: disposición lógica no comprometida) pulsa cada uno de los alambres hasta detectar la posición del lápiz mediante un cambio en la capacidad. Esta exploración tiene lugar 2 000 veces por segundo, lo que convierte la localización del cursor en un proceso muy rápido. Para ayudar a que el sistema funcione de manera fiable, conviene sostener el cursor por la banda metálica que rodea la punta.

Cuando se posa el lápiz en la superficie, el ordenador recibe la señal de "cursor en acción" y un informe de sus coordenadas en la tablilla. El efecto exacto que crea queda determinado por el software. En la pantalla puede aparecer un cursor en forma de cruz en la posición correspondiente, o puede desencadenarse una orden específica. Es aquí donde empieza a notarse la economía de la Grafpad. El lápiz sólo puede detectarse en una rejilla de 320 por 256 posiciones, lo que vuelve muy difícil dibujar detalles muy tenues o finos. Asimismo, la tablilla es muy pequeña: una hoja de papel Din A-4 es una zona de trabajo muy sensible.

El Grafpad dispone de tres paquetes de software, que van desde una simple rutina de demostración, pasando por un sencillo programa de dibujo,





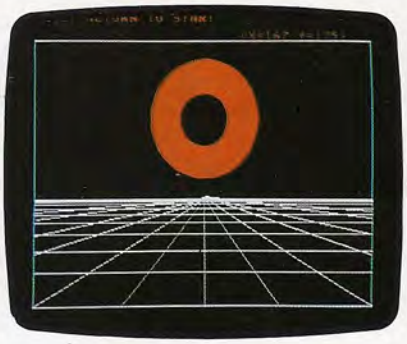
Cursor
Cada vez que el cursor se pone en contacto con la tablilla, su posición es transmitida al ordenador

Rejilla
Se utiliza una matriz de cables separados por 1,2 mm para determinar la posición del lápiz en la tablilla

Interruptor
El cursor del Grafpad tiene incorporado en la punta un minúsculo interruptor que desencadena la búsqueda de la posición del mismo en la plantilla

Barra de mando
Una zona concreta de la superficie presenta un conjunto de letras y de órdenes numéricas que puede emplearse con determinados programas

Capa de plexiglás
Una lámina de plexiglás protege la parte superior de la tablilla. A ésta se le pueden adherir hojas de superposición



Software de diseñador
El Grafpad viene con PROG2, un paquete de dibujo a pulso que se utilizó para crear estas imágenes. La capacidad para dibujar y rellenar círculos abrevia en gran medida el proceso

hasta un complejo paquete de CAD (*Computer-Aided Design*: diseño auxiliado por ordenador). Como se suministra en lenguaje máquina y en una versión de BASIC, es posible incorporar a estos programas la rutina simple de leer la superficie.

El programa de dibujo es un programa electrónico comparable a la mayoría de los paquetes artísticos existentes, incluso a aquellos que no cuentan con tablilla. Presenta todas las características básicas: líneas, casillas, círculos, triángulos y "dibujo a pulso", y puede rellenar una zona específica con determinado color. No obstante, carece de posibilidades más complejas como poder copiar y trasladar secciones del dibujo. A decir verdad, no tiene nada que un software para teclado no pueda hacer, si bien el Grafpad permite trazar diseños. La versión para el BBC sólo ofrece cuatro colores simultáneos y presenta la desventaja de que sus tiempos de respuesta son lentos.

El programa CAD es, lisa y llanamente, una demostración de algunos de los principios en juego. En primer lugar, usted crea una serie de caracteres que utilizará en la construcción de los diseños. En el caso de la electrónica, dichas formas pueden ser componentes como transistores, resistencias, etc. También puede crear puertas lógicas, diseños de muebles y hasta dibujos de azulejos. Una vez creados, usted se traslada al tablero de dibujo real donde puede repetir y acomodar libremente las formas e incorporarles líneas rectas.

Esto es muy parecido al funcionamiento de un auténtico paquete de CAD. Sin embargo, el software de la Grafpad no se plantea un uso serio. Entre los medios que necesitaría figuran la capacidad de etiquetar los diagramas, rotar los dibujos y ponerlos a escala, ampliar un fragmento específico de la pantalla, situar con toda exactitud formas pequeñas, etc. Es básica una mayor flexibilidad para corregir los errores y, en un sentido general, el programa CAD no permite utilizar el Grafpad como dispositivo de entrada. Pese a la pequeña barra de mando, muchas órdenes requieren entrada por el teclado y la operación general resulta pesada.

El Grafpad propiamente dicho es un periférico polifacético que ofrece mucho a cambio de lo que cuesta. Su superficie, resolución y fiabilidad son restringidas a fin de que su precio sea bajo. Sin embargo, el software que acompaña al sistema resulta decepcionante y la unidad atraerá más que a nadie a aquellos que quieran escribir sus propios programas. Pese a todo, mediante el esfuerzo adecuado, los tableros de gráficos como éste permitirán que los usuarios estudien nuevas posibilidades y se convertirán en un estímulo considerable para el diseño gráfico más avanzado de los micros personales.

Hacer inventario

Prosiguiendo con nuestra serie sobre informática contable, analizamos ahora cómo se pueden controlar eficazmente los envíos y pedidos de mercancías

En una empresa perfectamente organizada, donde el propietario o el gerente están al corriente de la demanda de sus clientes y el género de que disponen, pocas veces se producirá exceso o falta de existencias. Estas dos últimas situaciones son consecuencia de una información deficiente. Y los sistemas computerizados de almacén son un modo excelente de evitar una información insatisfactoria.

Para cumplir con la tarea del control de almacén, los ordenadores tienen que proporcionar varias informaciones a gerencia. La empresa necesita saber el volumen de existencias, la lenta o rápida rotación que caracteriza a determinados artículos, en qué momento debe procederse a una reorganización, así como el valor de lo que hay en existencia.

El sistema se propone controlar sus movimientos. Éstos pueden descomponerse en las siguientes categorías: salidas de género enviado a tenor de los pedidos; entradas de género recibido de los proveedores; género preparado para cubrir pedidos y género en curso.

A estas cuatro categorías hay que añadir otro tipo de movimiento según las devoluciones de género realizadas por los clientes o por la empresa a sus proveedores. A menudo el inventario también presenta discrepancias respecto de lo que hay realmente y de lo que tendría que haber en las estanterías.

El sistema debe valorar también las existencias. Por ello, además de registrar cantidades y controlar los movimientos de género, el programa tiene que dar información sobre precios.

Los sistemas de control de almacén se dividen en dos clases bastante diferenciadas entre sí, según que su destino sean pequeñas empresas de venta al por menor y cadenas de distribución, o bien sean fábricas. En este segundo caso, el sistema de almacén ha de tener en cuenta ciertas salidas de almacén que se incorporan al proceso de fabricación y vuelven a él en forma de unidad producida. Muchos sistemas de control de existencias basados en el microordenador intentan satisfacer las necesidades de ambos tipos de empresa. En este capítulo trataremos sólo de las empresas de venta al por menor.

Puesto que el control de almacén está relacionado con tantos aspectos de las actividades de una empresa, es corriente que los sistemas de existencia *integren* una serie de programas ("integración" significa que dos o más paquetes de aplicación son capaces de intercambiarse valores y datos). Un sistema plenamente integrado incluiría, por ejemplo, un libro de compras, un sistema para procesar pedidos de compra, un módulo para facturación, un libro de ventas y un procesador de pedidos.

La integración reporta varias ventajas. Tomemos, por ejemplo, una empresa que ha integrado su sistema de control de almacén con un sistema para procesar pedidos de ventas. Si ambos sistemas están en condiciones de comunicarse entre sí, es posible actualizar automáticamente los archivos de existencias al tiempo que se procesa el pedido. Y, si el sistema que procesa las ventas puede acceder al archivo de existencias para obtener una completa descripción del artículo y su precio de venta, con sólo introducir el código de existencias, el operador tendrá que manejar menos datos... y tendrá menos oportunidades de hacer entradas erróneas.

El punto de partida de todo sistema de control de almacén es, obviamente, el archivo de datos de existencias. Cada sistema tendrá un modo de identificar todos los artículos mediante un código numérico específico y un texto descriptivo. El programa utiliza ese número como clave de archivo.

Se trata de un sistema de almacén relativamente simple pero la adición de otros paquetes más sofisticados lo convierte en un potente instrumento. El *Stock recording system* de Dragon Data para el Dragon 64, provisto de unidad de disco flexible, es un ejemplo del sistema más sencillo.

Este paquete ofrece al usuario ocho caracteres alfanuméricos para codificar el artículo, más un código de grupo de productos de dos dígitos. Ello significa que cualquier artículo puede incluirse en uno de los 50 grupos de productos (el máximo que ofrece el sistema). Si a cualquier artículo en existencia se le asigna un número de menos de ocho dígitos, por ejemplo, el 445, el sistema lo ajustará automáticamente por la derecha. Es decir que introducir 445 es lo mismo que introducir 00445 o 00000445.

Esta cuestión de la alineación es importante. Por ejemplo, el *Stock control system* de ACT Pulsar, que opera en micros más poderosos como el IBM PC y el Sirius, permite a los usuarios elegir entre un sistema de codificación de ajuste por la derecha o por la izquierda. De lo cual resulta que esos sistemas de codificación de existencias son totalmente distintos e incompatibles.

Cortesía de J. Sainsbury Plc.

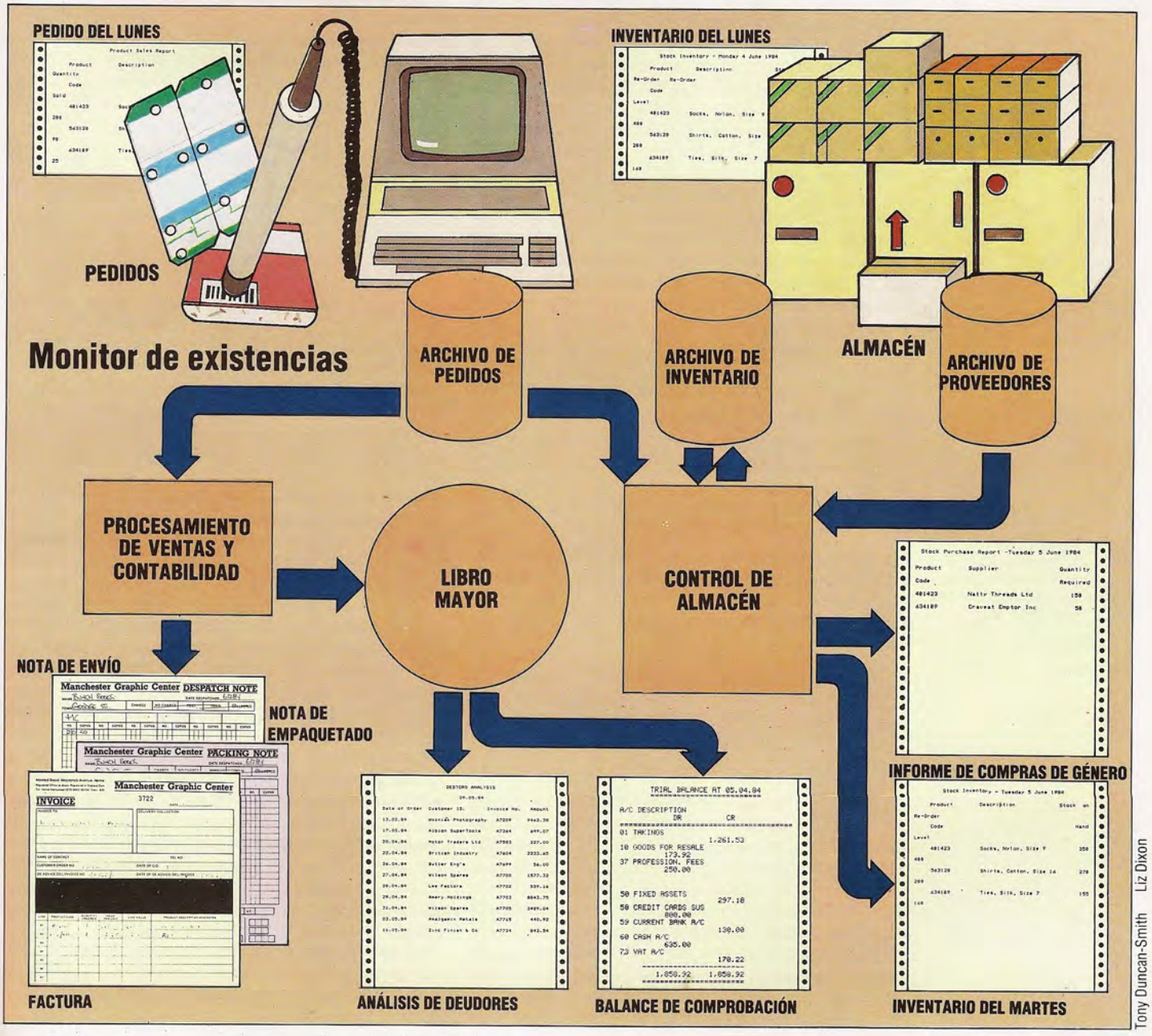
Mantener las estanterías llenas
Las cajas registradoras automatizadas pueden extraer directamente la información sobre los productos de las etiquetas con código de barras y registrar las ventas en un ordenador central de control de existencias. Esta realimentación instantánea permite a las grandes tiendas tener la seguridad de que las estanterías y los almacenes contienen los productos pertinentes en las cantidades adecuadas

El código del producto puede tener hasta 16 caracteres alfanuméricos. El sistema de ajuste por la derecha es un sistema de códigos numéricamente ordenados. El sistema de ajuste por la izquierda está dirigido a usuarios que cuentan con sistemas de codificación bastante más complejos y que incluyen alfanuméricos como el PX445/44. Permite tener códigos de diversas longitudes para distintos productos y resulta útil en los sistemas en los que el usuario quiere emplear el código de producto para identificar alguna característica del artículo en existencia, como puede ser el modelo, la talla o el color de unas prendas determinadas.

El paquete *Powerstock* de Omicron, que se utiliza en el Sirius, es un sistema más costoso destinado a usuarios cuyas necesidades son más complejas. Presenta un sistema de codificación aún más sutil y se define por grupos de existencias. Un grupo de existencias puede ser cualquier conjunto de registros de existencias relacionados por un proceso común o por necesidades de información. Lo que lo diferencia del sistema de codificación *Pulsar* de ajuste por la izquierda es que cada grupo de existencias se procesa por separado y a cada uno se le pueden asignar diversas reglas de procesamiento. Recuerde que cualquiera que sea el número de código asignado a una línea de productos en el *Pulsar*, todos los números de código se procesan de la misma manera.

En consecuencia, la estructura codificadora de los sistemas de control de existencias ha de ser lo bastante flexible para que los usuarios puedan identificar y subdividir los artículos en existencia. Los sistemas más simples que operan en los ordenadores personales baratos suelen ofrecen menos flexibilidad debido a las limitaciones, una vez más, de memoria y almacenamiento. El sistema de Dragon Data, por ejemplo, está diseñado para manejar un máximo de 350 artículos en existencia. El sistema *Powerstock* de Omicron no es restrictivo y su número máximo de artículos depende de la configuración del ordenador del usuario

Control de almacén
Los sistemas integrados de ventas y de control de existencias de las empresas con gran movimiento pueden beneficiarse enormemente con un sistema automatizado de punto de venta que permite mantener al día el inventario y los libros de contabilidad. Los datos de venta pueden incorporarse en un precinto a modo de una etiqueta Kimball o un código de barras pegado al producto. Éstos son captados por un lector óptico adosado a la caja registradora, que a su vez podría ser un microordenador, o transmitir los datos a un ordenador para su procesamiento



Grados de precisión

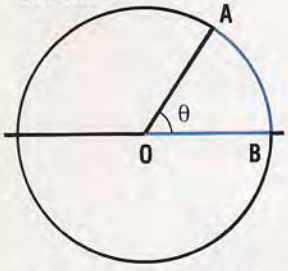
Esta vez estudiaremos qué tratamiento tienen las funciones seno y coseno en los programas BASIC y los métodos de comprobación de ambas para detectar posibles fuentes de error

Puesto que el BASIC cuenta con las funciones COS (coseno) y SIN (seno), será tarea fácil calcular la posición de un punto en una línea después de hacerlo rotar un cierto número de grados. El COS de θ nos dará un punto en el *eje x* (la abscisa *x*) y el SIN de θ dará el punto en el *eje y* (la ordenada *y*). No obstante, a la hora de emplear estas funciones conviene recordar que la mayoría de las funciones del BASIC operan en radianes y no en grados. Otra cuestión que se debe controlar es que los valores de θ dados a conocer pueden no ser fiables a medida que θ se aproxima a 0 o a 1. Pronto comprenderá la diferencia que hay entre grados y radianes.

Cuando se dibuja una porción de circunferencia (denominada *arco*) con longitud exactamente igual al radio, al ángulo central se le llama radián (véase la ilustración). Si además el radio del círculo lo tomamos como unidad de medida, esta porción de la circunferencia tendrá la longitud de una unidad. La fórmula para hallar la longitud de una circunferencia es $2\pi r$, por lo que el recorrido de una vuelta completa medirá 2π radianes. Pero según una expresión más conocida, una revolución completa —el giro necesario para trazar una circunferencia— es de 360°. Por lo tanto, esos 360° equivalen a 2π radianes. Y así tenemos un modo sencillo de relacionar grados y radianes:

$360° = 2\pi$ radianes
$180° = \pi$ radianes
$90° = \pi/2$ radianes
$1° = \pi/180 = 0{,}0174$ radianes

Un radián



Cuando la longitud del radio, OB, se toma como unidad y el arco AB mide lo mismo que el radio, entonces al ángulo central θ se le llama *radián*. Muy brevemente: si $\overline{OB} = \widehat{BA}$ entonces $\widehat{AOB}$ = 1 radián

Un programa BASIC que tuviera que hallar el coseno de un ángulo medido en grados, primero tendría que convertir la medida del ángulo de grados en radianes y luego utilizar la función COS. Pruebe con esto:

```
10 INPUT "DE ENTRADA ANGULO EN GRADOS";A
20 LET B# = A* 0,0174
30 LET C# = 2 COS (B#)
40 PRINT "EL COSENO DE ";A;" GRADOS ES ";C#
50 END
```

El símbolo # indica que las variables del programa son de doble precisión (cuestión que abordaremos más adelante en este mismo capítulo). Una sencilla modificación de dicho programa utilizando la función seno dará entrada a todos los valores de θ de 0° a 360° y proporcionará el seno de dichos valores en forma de tabla. Si dichos valores se pasan al *eje y* de una gráfica (donde el *eje x* representa los valores de θ en radianes), aparecerá el gráfico de onda de seno conocido por los entusiastas de la alta fidelidad y los ingenieros electrónicos (véase el diagrama p. 635). Esta conocida curva no es más que la representación de los puntos de la intersección de la hipotenusa con la circunferencia unitaria sobre el *eje y* para todos los ángulos de rotación. En síntesis, es un modo alternativo de describir matemáticamente una circunferencia.

Unas pocas versiones de BASIC permiten que las funciones SIN y COS operen en grados o en radianes mediante la utilización de un "interruptor de software", pero la mayoría de éstas no cuentan con él. Si prefiere trabajar siempre en grados, es posible recurrir a una "función definida por el usuario" que le facilite las conversiones. Ésta es una de tantas posibles:

```
10 REM FUNCION DEFINIDA POR EL USUARIO
   PARA TRABAJAR EN GRADOS
20 DEF FNGSIN (G#) = SIN(G#*0.017453293)
30 INPUT "DE ENTRADA AL ANGULO EN
   GRADOS";G#
40 PRINT "EL SENO DE ";G#;" GRADOS ES";
   FNGSIN(G#)
50 END
```

La línea 20 define una función denominada GSIN (que significa grados-seno), que utiliza como único parámetro la variable de doble precisión G#. La mitad derecha de la definición sólo muestra cómo debe obtenerse el valor que ha de ser calculado por la función (el seno de un ángulo en grados). Para echar mano de una función definida por el usuario, basta con que utilice como de costumbre el nombre de la función (poniendo entre paréntesis el valor a operar). No obstante, note que la línea que contiene la definición tiene que ser ejecutada antes de poder llamar a la función.

Uno de los problemas que surgen al usar la función seno en BASIC es que no todas las versiones la manejan correctamente cuando se aproxima a 0 el valor de θ. Es obvio que, a medida que θ se aproximará a 0, el valor de SIN θ también se aproximará a 0 (véase p. 634). En síntesis, a medida que el ángulo se aproxima a 0, el arco de la circunferencia que define θ también se acerca a 0, y el punto en que la hipotenusa corta el círculo se acerca a 0 en el *eje y*. Por desgracia, la precisión del BASIC es limitada. Dicho de otro modo, este lenguaje sólo puede manejar valores grandes hasta cierto límite y valores pequeños hasta cierto valor también. Si θ es muy pequeño (p. ej., 1.0E-36, es decir, 1×10 elevado a la menos 36), es posible que el BASIC no pueda resolverlo y simplemente devuelva un valor de 0 para el seno de dichos números. Antes de utilizar la función seno, intente someter a prueba su BASIC poniendo en práctica el programita que detallamos a continuación:

Errores de redondeo por aproximación a cero

```
1 REM PRUEBA DE ERRORES PARA LA FUNCION SIN PROXIMA A CERO
10 LEXT X = 10/6
20 PRINT "ITERACION","  VAL DE X","  VAL DE SIN(X)"
30 FOR I = I TO 40
40 LET X = X / 10
50 PRINT I,X,SIN(X)
60 NEXT I
70 END
```

ITERACIÓN	VAL DE X	VAL DE SIN(X)
1	.166667	.165896
2	.0166667	.0166659
3	1.66667E-03	1.66667E-03
4	1.66667E-04	1.66667E-04
5	1.66667E-05	1.66667E-05
6	1.66667E-06	1.66667E-06
7	1.66667E-07	1.66667E-07
8	1.66667E-08	1.66667E-08
9	1.66667E-09	1.66667E-09
10	1.66667E-10	1.66667E-10
11	1.66667E-11	1.66667E-11
12	1.66667E-12	1.66667E-12
13	1.66667E-13	1.66667E-13
14	1.66667E-14	1.66667E-14
15	1.66667E-15	1.66667E-15
16	1.66667E-16	1.66667E-16
17	1.66667E-17	1.66667E-17
18	1.66667E-18	1.66667E-18
19	1.66667E-19	1.66667E-19
20	1.66667E-20	1.66667E-20
21	1.66667E-21	1.66667E-21
22	1.66667E-22	1.66667E-22
23	1.66667E-23	1.66667E-23
24	1.66667E-24	1.66667E-24
25	1.66667E-25	1.66667E-25
26	1.66667E-26	1.66667E-26
27	1.66667E-27	1.66667E-27
28	1.66667E-28	1.66667E-28
29	1.66667E-29	1.66667E-29
30	1.66667E-30	1.66667E-30
31	1.66667E-31	1.66667E-31
32	1.66667E-32	1.66667E-32
33	1.66667E-33	1.66667E-33
34	1.66667E-34	1.66667E-34
35	1.66667E-35	1.66667E-35
36	1.66667E-36	1.66667E-36
37	1.66667E-37	1.66667E-37
38	1.66667E-38	1.66667E-38
39	0	0
40	0	0

Damos también una ejecución de este programa utilizando MBASIC Microsoft. Este intérprete concreto de BASIC maneja bastante bien el SIN de pequeñas cifras y no crea problemas hasta que el valor de θ es menor que 10E-38 (una coma decimal seguida de 37 ceros).

Este programa depende de un margen dinámico adecuado en el manejo que el BASIC haga de las operaciones aritméticas de coma flotante. Conviene recordar que antes de que usted pueda utilizar con seguridad cualquier operación matemática del BASIC tendrá que saber cuál es el margen de números que puede manejar con exactitud.

Recuerde que el nombre de variable, como por ejemplo X o TREND, será automáticamente de *precisión simple* (es decir, capaz sólo de almacenar hasta siete dígitos). Por otra parte, las variables pueden especificarse como de, o pasarse a, precisión simple añadiendo un signo de exclamación, como en X! o TREND!. Las variables de *doble precisión* (que pueden almacenar hasta 17 dígitos) se especifican añadiendo el símbolo #, como en X# o TREND#. Las variables que sólo pueden almacenar números enteros se especifican en muchas versiones mediante el agregado del símbolo de porcentaje, como en X% o TREND%.

Concluimos este artículo con un breve programa que le permite comprobar cuántos dígitos pueden almacenarse en una variable de su versión de BASIC, así como una salida impresa del programa cuando funciona utilizando BASIC Microsoft. Hay dos versiones, una para probar cifras pequeñas y otra para grandes. La salida impresa de las cifras pequeñas muestra que a medida que los números se vuelven muy pequeños (menos que 3,3 × 10E-38), el BASIC redondea los números a cero. En las cifras grandes (mayores de 3,3 × 10E37), se produce un desbordamiento y los resultados son poco fiables. Es posible que si necesita operar con cifras muy grandes o muy pequeñas, tenga que preparar rutinas aritméticas específicas para superar dichas limitaciones.

Los números pequeños en BASIC

```
1 REM PRUEBAS CON NUMS. PEQUEÑOS EN BASIC
10 LET X# = .00003333333333#
20 PRINT "ITERACION","  DOBL PREC","  ","SIMPL PREC"
30 PRINT
40 FOR I = 1 TO 40
50 LET X# = X# / 10
60 LET X! = X#
70 PRINT I,X#,X!
80 NEXT I
90 END
```

ITERACIÓN	DOBL PREC	SIMPL PREC
1	.0000033333333333	3.33333E-06
2	.00000033333333333	3.33333E-07
3	3.333333333D-08	3.33333E-08
4	3.333333333D-09	3.33333E-09
5	3.333333333D-10	3.33333E-10
6	3.333333333D-11	3.33333E-11
7	3.333333333D-12	3.33333E-12
8	3.333333333D-13	3.33333E-13
9	3.333333333D-14	3.33333E-14
10	3.333333333D-15	3.33333E-15
11	3.333333333D-16	3.33333E-16
12	3.333333333D-17	3.33333E-17
13	3.333333333D-18	3.33333E-18
14	3.333333333D-19	3.33333E-19
15	3.333333333D-20	3.33333E-20
16	3.333333333D-21	3.33333E-21
17	3.333333333D-22	3.33333E-22
18	3.333333333D-23	3.33333E-23
19	3.333333333D-24	3.33333E-24
20	3.333333333D-25	3.33333E-25
21	3.333333333D-26	3.33333E-26
22	3.333333333D-27	3.33333E-27
23	3.333333333D-28	3.33333E-28
24	3.333333333D-29	3.33333E-29
25	3.333333333D-30	3.33333E-30
26	3.333333333D-31	3.33333E-31
27	3.333333333D-32	3.33333E-32
28	3.333333333D-33	3.33333E-33
29	3.333333333D-34	3.33333E-34
30	3.333333333D-35	3.33333E-35
31	3.333333333D-36	3.33333E-36
32	3.333333333D-37	3.33333E-37
33	3.333333333D-38	3.33333E-38
34	0	0
35	0	0
36	0	0
37	0	0
38	0	0
39	0	0
40	0	0

Los números grandes en BASIC

```
 1 REM PRUEBAS CON NUMS. GRANDES EN BASIC
10 LET X# = 3.3333333333334#
20 PRINT "ITERACION","  DOBL PREC","  "," SIMPL PREC"
30 PRINT
40 FOR I = 1 TO 40
50 LET X# = X# * 10
60 LET X! = X#
70 PRINT I,X#,X!
80 NEXT I
90 END
```

ITERACION	DOBL PREC	SIMPL PREC
1	33.3333333333334	33.3333
2	333.3333333333334	333.333
3	3333.333333333334	3333.33
4	33333.33333333334	33333.3
5	333333.3333333334	333333
6	3333333.333333334	3.33333E + 06
7	33333333.33333334	3.33333E + 07
8	333333333.3333334	3.33333E + 08
9	3333333333.333334	3.33333E + 09
10	33333333333.33334	3.33333E + 10
11	333333333333.3334	3.33333E + 11
12	3333333333333.334	3.33333E + 12
13	33333333333333.34	3.33333E + 13
14	333333333333333.4	3.33333E + 14
15	3333333333333334	3.33333E + 15
16	3.3333333333334D + 16	3.33333E + 16
17	3.3333333333334D + 17	3.33333E + 17
18	3.3333333333334D + 18	3.33333E + 18
19	3.3333333333334D + 19	3.33333E + 19
20	3.3333333333334D + 20	3.33333E + 20
21	3.3333333333334D + 21	3.33333E + 21
22	3.3333333333334D + 22	3.33333E + 22
23	3.3333333333334D + 23	3.33333E + 23
24	3.3333333333334D + 24	3.33333E + 24
25	3.3333333333334D + 25	3.33333E + 25
26	3.3333333333334D + 26	3.33333E + 26
27	3.3333333333334D + 27	3.33333E + 27
28	3.3333333333334D + 28	3.33333E + 28
29	3.3333333333334D + 29	3.33333E + 29
30	3.3333333333334D + 30	3.33333E + 30
31	3.3333333333334D + 31	3.33333E + 31
32	3.3333333333334D + 32	3.33333E + 32
33	3.3333333333334D + 33	3.33333E + 33
34	3.3333333333334D + 34	3.33333E + 34
35	3.3333333333334D + 35	3.33333E + 35
36	3.3333333333334D + 36	3.33333E + 36
37	3.3333333333334D + 37	3.33333E + 37
38	1.7014118346046930 + 38	1.70141E + 38
39	1.7014118346046930 + 38	1.70141E + 38
40	1.7014118346046930 + 38	1.70141E + 38

Banderín de salida

Daremos una atenta mirada a la instrucción suma (ADD) y al "registro indicador de estado" (PSR), cuyo indicador de arrastre es fundamental en la suma

La instrucción suma en el lenguaje assembly tanto del Z80 como del 6502 es ADC (abreviatura de "ADd with Carry": suma con arrastre), una mnemotecnia de gran importancia para la programación en dicho lenguaje. El concepto de un bit "de arrastre" tiene especial significado. Consideremos la suma de dos números hexadecimales en el acumulador:

```
   A7      =     10100111
+  3E      =  +  00111110
-------------------------
   E5      =     11100101
```

Puesto que el acumulador es un registro de ocho bits, tanto los números a sumar como la suma propiamente dicha no podrán superar el intervalo de 0000 0000 a 1111 1111 en binario, o sea de $00 a $FF en hexa (como aparecen aquí), pues de lo contrario no encajarán en el acumulador. En consecuencia, ¿ello significa que estamos limitados a adiciones en las que la suma es inferior a $100 (en decimal, 256)? Veamos otra adición en el acumulador, una que viole dicha restricción:

```
   FF      =     11111111
+  FF      =  +  11111111
-------------------------
   1FE     =     111111110
```

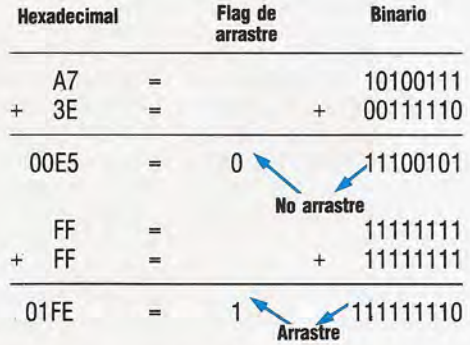
Se trata de la adición de dos números de un solo byte lo más grandes posible. La suma no parece viable, pues requiere un acumulador de nueve bits y el que está a nuestra disposición es de ocho. La solución de la dificultad queda sugerida en el planteamiento del problema: sólo necesitamos un bit adicional en el acumulador para contener el número más grande que puede generar la suma de cifras de un solo byte. El bit adicional sólo hace falta en la suma, no en los operandos de la adición, y en caso de que haya un "arrastre" del bit más significativo del acumulador.

Registro indicador de estado

Así pues, ese bit adicional se conoce como "bit de arrastre" (*carry bit*) y está localizado en un registro compuesto de ocho bits asociado con el acumulador y conocido como *registro indicador de estado* (*Processor Status Register*: PSR). Este importante registro está conectado con el acumulador y la ALU de modo tal que los bits individuales del PSR se activan o se desactivan después de cualquier operación en el acumulador, según sean los resultados de dicha operación. El contenido del registro indicador de estado puede considerarse como un número simple, pero con frecuencia resulta más ilustrativo tratarlo como una disposición de ocho indicadores simples, que reciben el nombre de *flags* o *banderas*, cuyo estado individual muestra los efectos específicos de la última operación (un flag es una variable cuyo valor indica la verdad de alguna condición, por lo que no se toma como valor absoluto. Una variable de flag sólo acostumbra presentar dos estados o condiciones: sí o no, activado o desactivado, 0 o 1).

Cuando en el acumulador se realiza cualquier operación que produce el arrastre del octavo bit, el flag de arrastre del PSR se posicionará automáticamente en 1; toda operación que no provoque un arrastre restablecerá (o posicionará en 0) el flag de arrastre. Esto sólo se aplica a operaciones que podrían provocar un arrastre. Algunas operaciones como cargar al o almacenar desde el acumulador no afectan al flag de arrastre. A partir de este punto, cada vez que abordemos una nueva instrucción en lenguaje assembly, nos interesará saber cuáles son los bits del PSR (o registro de flag) afectados. Lógicamente, tendremos que saber más cosas sobre los otros flags del PSR, pero concluyamos antes nuestro análisis del flag de arrastre.

En la mayoría de los casos, cuando sumemos dos números de un solo byte, no sabremos por adelantado cuáles serán, por lo que tendremos que estar dispuestos a que el resultado de dicha suma supere $FF; esto suele suponer reservar dos bytes de RAM para contener el resultado de la suma. Volvamos una vez más a los ejemplos anteriores:



En ambas sumas, el resultado de la adición está representado por un número de dos bytes. En el primer caso, el flag de arrastre se restablece en cero, porque no hay arrastre del octavo bit en la suma (el resultado de dos bytes es $00E5, del que el byte *hi* —o byte alto— es $00). Pero como en el segundo caso hay un arrastre del octavo bit, el flag de arrastre se enciende y el byte *hi* del resultado es $01.

Por lo tanto, para comprobar que obtenemos el resultado correcto de una adición, debemos almacenar el contenido del acumulador en el byte *lo* (o byte bajo) de la posición de dos bytes y a continua-

ción almacenar el flag de arrastre como byte *hi* de dicha posición. No existe una instrucción exclusiva para almacenar el flag de arrastre, pero el código operativo ADC fue formulado pensando en dicha operación: ADC significa realmente "sumar el operando de la instrucción al contenido actual del flag de arrastre y añadir luego ese resultado al contenido del acumulador". Así, la adición es un proceso en dos etapas; en la primera se utiliza el estado actual del flag de arrastre, mientras que en la segunda se actualiza el estado de dicho flag de arrastre.

Todo ello significa que antes de iniciar la suma debemos considerar el estado actual del flag de arrastre, ya que será sumado al resultado de la suma propiamente dicha: de ahí que se disponga de CLC y AND A, las dos instrucciones no explicadas en capítulos anteriores. La primera, una instrucción del 6502, significa "desactivar el flag de arrastre" (*CLear the Carry flag*), y es eso exactamente lo que hace. La versión del Z80, AND A, significa "activar un AND (y) lógico entre el acumulador y él mismo". Aunque no está diseñada exclusivamente para restaurar el flag de arrastre, ése es el efecto que produce y no afecta nada más, por lo que a menudo se la utiliza como el equivalente de la CLC del 6502 para el Z80.

Por lo tanto, luego de desactivar el flag de arrastre y tras realizar la suma, a continuación debemos almacenar su contenido. Ello se consigue añadiendo el valor inmediato $00 al byte *hi* del resultado. Esto no afectará al byte si el flag de arrastre está desactivado, pero le sumará 1 si se halla activado.

Todo lo que hemos dicho en este capítulo no es sino el primer método de la aritmética de un solo byte. En resumen, se deberá:

1) Desactivar el flag de arrastre
2) Cargar el acumulador con un número
3) Sumarle el segundo número
4) Almacenar el contenido del acumulador en el byte *lo* de una posición
5) Cargar el acumulador con el contenido del byte *hi*
6) Sumarle el valor inmediato $00
7) Almacenar el contenido del acumulador en el byte *hi*

Cuando este procedimiento se convierte a lenguaje assembly, obtenemos:

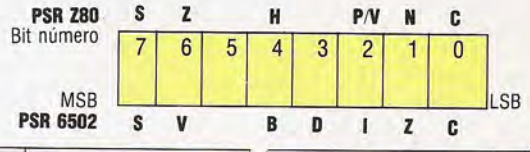
COMÚN PARA AMBOS PROCESADORES		
Etiqueta	**Directriz**	**Operando**
BYTE1	EQU	$FF
BYTE2	EQU	$FF
LOBYTE	EQU	$A000
HIBYTE	EQU	$A001
	ORG	$A020

Z80		6502	
Opcode	**Operando**	**Opcode**	**Operando**
LD	A,$00	LDA	#$00
LD	(HIBYTE),A	STA	HIBYTE
AND	A	CLC	
LD	A,BYTE1	LDA	#BYTE1
ADC	A,BYTE2	ADC	#BYTE2
LD	(LOBYTE),A	STA	LOBYTE
LD	A,(HIBYTE)	LDA	HIBYTE
ADC	A,$00	ADC	#$00
LD	(HIBYTE),A	STA	HIBYTE
RET		RTS	

Recuerde que los valores dados a LOBYTE, HIBYTE y ORG sólo son a modo de ejemplo y que usted ha de escoger valores adecuados para la máquina que utiliza. Note que las dos primeras instrucciones del programa cargan $00 en el HIBYTE a fin de que se vea corrompida por datos aleatorios. No tenemos que desactivar LOBYTE del mismo modo porque su contenido de partida está sobreescrito en el byte *lo* del resultado.

Conviene resaltar una vez más las diferencias de enfoque entre el lenguaje assembly del Z80 y del 6502, como ya se ha visto en el ejemplo. El código del 6502 se interpreta fácilmente en cuanto uno se acostumbra a él; la mnemotecnia propiamente dicha y el uso de "#" para señalar datos inmediatos clarifica el significado de cada instrucción. La versión del Z80 es menos directa pues la mnemotecnia LD se utiliza para todas las transferencias de datos, ya sea que entren o salgan del acumulador. Además, no existe el símbolo "#" que señale datos inmediatos y sólo se indica mediante la ausencia de paréntesis en el operando. De este modo, LD A,BYTE1 significa "cargar el acumulador con los datos inmediatos BYTE1"; mientras que LD A,(HIBYTE) significa "cargar el acumulador desde la dirección HIBYTE". En la lista completa de lenguaje assembly no hay ambigüedad respecto del significado de dichas instrucciones ya que el valor hexadecimal del código operativo identifica exclusivamente la instrucción. Esto parecería dar por sentada la cuestión: el código operativo podría ser exclusivo, pero si hay diversas opciones de códigos operativos exclusivos, ¿cómo hace el ensamblador (o la persona que realiza el ensamblaje) para elegir entre ellos? La respuesta reside en la modalidad de direccionamiento, que será el tema que nos ocupará en el próximo capítulo.

Por último, debemos mencionar que el registro de estado del proceso contiene otros flags además del de arrastre, cuestión que ahora analizaremos fugazmente y sobre la que volveremos más adelante con mayor detalle:



BIT PSR	Z80	6502	BIT PSR
7	(S) = SIGNO	(S) = SIGNO	7
6	(Z) = CERO	(V) = OVERFLOW	6
5	no usado	no usado	5
4	(H) = MEDIO ARRASTRE	(B) = RUPTURA	4
3	no usado	(D) = MODO BCD	3
2	(P/V) = PAR./OVERFLOW	(I) = INTERRUPCION	2
1	(N) = RESTA	(Z) = CERO	1
0	(C) = ARRASTRE	(C) = ARRASTRE	0

Para nuestro objetivo, en este momento los flags importantes son el de arrastre, el de signo y el de cero. Hemos visto que después de una adición la bandera de arrastre contiene el valor del arrastre del octavo bit del acumulador. La bandera de signo siempre es una copia del octavo bit (bit 7) del acumulador y la bandera de cero se posiciona en 1 si el contenido del acumulador es cero y se reacomoda en 0 si el contenido no es cero.

Soluciones al ejercicio de assembly de la p. 638

1) Los programas ensamblados se ofrecen en el cuadro de la derecha.
Note que los símbolos BYTE1 y BYTE2 se utilizan como datos simbólicos inmediatos y también como direcciones simbólicas. Sin embargo, cuando se utilizan como direcciones, es necesario ensamblarlos en forma de dos bytes.

2) La instrucción "regreso de una subrutina" está ausente del final de ambos programas. En la versión del 6502, el código completo tendría que incluir la siguiente línea:

```
A00D          60          RTS
```

y la versión del Z80 necesita esta línea:

```
A00D          C9          RET
```

3) El valor $45 se carga en el registro de acumulador como dato inmediato y luego se le suma $45, de modo que el acumulador contiene el valor $8A (o sea, 45 + 45 = 8A en hexa). Este total acumulado se almacena después en la RAM, dirección $0045. El valor $38 es ahora añadido al acumulador como dato inmediato, por lo que el acumulador pasa a contener el valor $C2 ($8A + $38 = $C2). Finalmente, este total se almacena en la RAM en la posición $0038.

4) "Datos inmediatos" son aquellos que realmente se almacenan en la instrucción. En las instrucciones que vimos en los programas de ejercicios (como LDA #$9C y LD A,$E4), los valores $9C y $E4 son los datos a cargar en el acumulador. Están almacenados en las instrucciones de las que son operandos e implican el contenido del byte inmediatamente posterior al opcode. Si no se dispone de datos, debe almacenarse en alguna otra parte de la memoria y hay que referirse a él por su dirección más que por su valor.

5) Como el valor del BYTE1 es $45, si se escribe como dirección será la posición de memoria $0045. Sin lugar a dudas, esa dirección está en la página cero de la memoria.

Dirección de posición	Lenguaje máquina	Lenguaje Assembly		
6502				
0000			EQU	$A000
0000		BYTE1	EQU	$45
0000		BYTE2	EQU	$38
0000			ORG	
A000	A9 45		LDA	#BYTE1
A002	18		CLC	
A003	69 45		ADC	#BYTE1
A005	8D 45 00		STA	BYTE1
A008	69 38		ADC	#BYTE2
A00A	8D 38 00		STA	BYTE2
Z80				
0000		START	EQU	$A000
0000		BYTE1	EQU	$45
0000		BYTE2	EQU	$38
0000			ORG	
A000	3E 45		LD	A,BYTE1
A002	A7		AND	A
A003	CE 45		ADC	A,BYTE1
A005	32 45 00		LD	(BYTE1),A
A008	CE 38		ADC	A,BYTE2
A00A	32 38 00		LD	(BYTE2),A

Ejercicio

Tal vez queramos examinar el contenido del registro indicador de estado (PSR) y nos convenga mostrar este número como byte binario y no hexadecimal. Aquí tiene la versión de una "subrutina de conversión de decimal a binario" del Spectrum. En este ejercicio se le pide que lo incorpore en el programa monitor de la pág. 598.

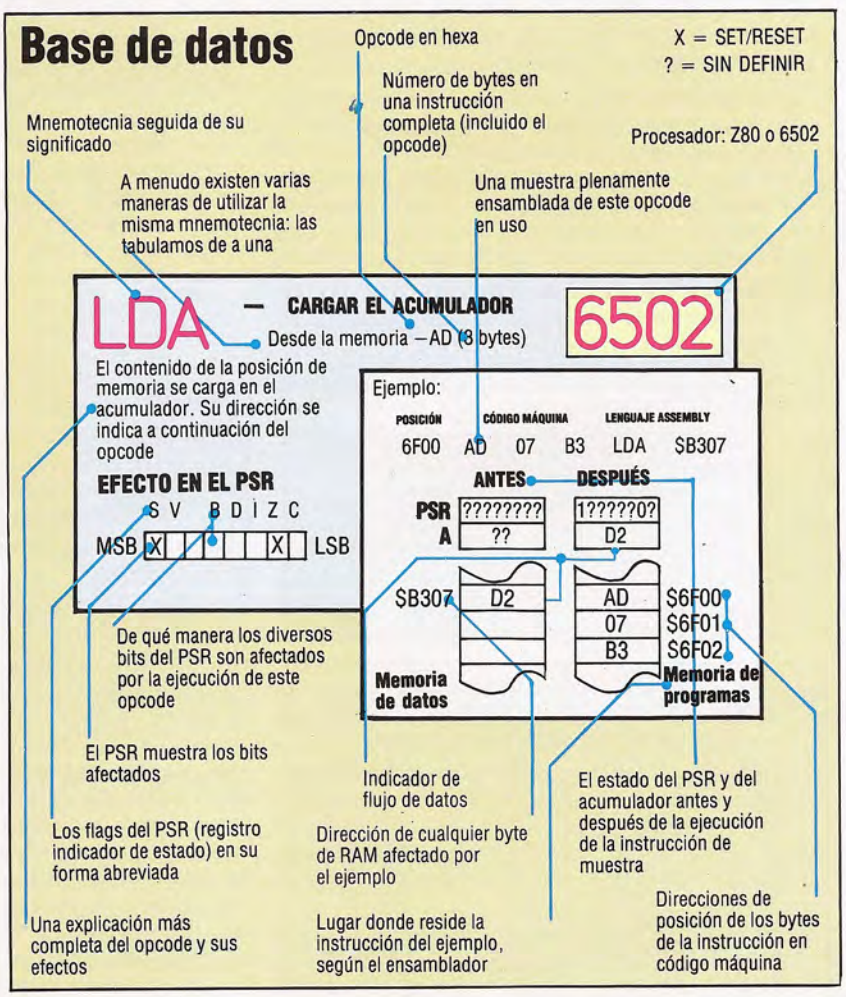
```
7000 REM******S/R BYTE BINARIO*******
7001 REM*CONVIERTE UN NUMERO N (256)*
7002 REM*EN BINARIO DE 8 CARACTERES*
7003 REM*REPRESENTACION EN B$*
7010 B$ = ""
7020 FOR D = 8 TO 1 STEP-1
7030 LET N1 = INT(N/2)
7040 LET R = N-2*N1
7050 LET B$ = STR$(R) + B$
7060 LET N = N1
7070 NEXT D
7080 RETURN
```

Complementos al BASIC

En el Commodore 64, cambie la línea 7050 de la subrutina por:

```
7050 B$ = MID$ (STR$(R),2) + B$
```

Base de datos



LDA — CARGAR EL ACUMULADOR — 6502

Inmediato —A9 (2 bytes)

El contenido del byte siguiente al opcode es cargado en el acumulador

EFECTO EN EL PSR

	S	V		B	D	I	Z	C
MSB	X						X	LSB

Ejemplo:

POSICIÓN	COD. MÁQUINA	LENG. ASSEMBLY
A000	A9 3F	LDA #$3F

	ANTES	DESPUÉS	
PSR	????????	0??????0	
A	??	3F	
		A9	$A000
		3F	$A001
Memoria datos			Memoria programas

LD A, — CARGAR EL ACUMULADOR — Z80

Inmediato —3E (2 bytes)

El contenido del byte siguiente al opcode es cargado en el acumulador.

EFECTO EN EL PSR

	S	Z	H	V	N	C
MSB						LSB

SIN EFECTO

Ejemplo:

POSICIÓN	COD. MÁQUINA	LENG. ASSEMBLY
A000	3E 9B	LD A,$9B

	ANTES	DESPUÉS	
PSR	????????	????????	
A	??	9B	
		3E	$A000
		9B	$A001
Memoria datos			Memoria programas

LDA — CARGAR EL ACUMULADOR — 6502

Desde la memoria —AD (3 bytes)

El contenido de la posición de memoria, cuya dirección va después del opcode, se carga en el acumulador.

EFECTO EN EL PSR

	S	V		B	D	I	Z	C
MSB	X						X	LSB

Ejemplo:

POSICIÓN	COD. MÁQUINA	LENG. ASSEMBLY
6F00	AD 07 B3	LDA $B307

	ANTES	DESPUÉS	
PSR	????????	1??????0	
A	??	D2	
$B307	D2	AD	$6F00
		07	$6F01
		B3	$6F02
Memoria datos			Memoria programas

LD A, — CARGAR EL ACUMULADOR — Z80

Desde la memoria —3A (3 bytes)

El contenido de la posición de memoria, cuya dirección va después del opcode, se carga en el acumulador.

EFECTO EN EL PSR

	S	Z	H	V	N	C
MSB						LSB

SIN EFECTO

Ejemplo:

POSICIÓN	COD. MÁQUINA	LENG. ASSEMBLY
6F00	3A E9 F4	LD A,($F4E9)

	ANTES	DESPUÉS	
PSR	????????	????????	
A	??	8C	
$F4E9	8C	3A	$6F00
		E9	$6F01
		F4	$6F02
Memoria datos			Memoria programas

STA — ALMACENAR ACUMULADOR — 6502

A la memoria —8D (3 bytes)

El contenido del acumulador se carga en la posición de memoria cuya dirección va después del opcode.

EFECTO EN EL PSR

	S	V		B	D	I	Z	C
MSB								LSB

SIN EFECTO

Ejemplo:

POSICIÓN	COD. MÁQUINA	LENG. ASSEMBLY
E532	8D A2 65	STA $65A2

	ANTES	DESPUÉS	
PSR	????????	????????	
A	68	68	
$65A2	68	8D	$E532
		A2	$E533
		65	$E534
Memoria datos			Memoria programas

LD( ),A — CARGAR EL ACUMULADOR — Z80

A la memoria —32 (3 bytes)

El contenido del acumulador se carga en la posición de memoria cuya dirección va después del opcode.

EFECTO EN EL PSR

	S	Z	H	V	N	C
MSB						LSB

SIN EFECTO

Ejemplo:

POSICIÓN	COD. MÁQUINA	LENG. ASSEMBLY
E532	32 E9 F4	LD (SF4E9),A

	ANTES	DESPUÉS	
PSR	????????	????????	
A	4B	4B	
$F4E9	4B	32	$E532
		E9	$E533
		F4	$E534
Memoria datos			Memoria programas

ADC — ADD CON ARRASTRE — 6502

Inmediato —69 (2 bytes)

El flag de arrastre más el contenido del byte siguiente al opcode son sumados el contenido del acumulador.

EFECTO EN EL PSR

	S	V		B	D	I	Z	C
MSB	X	X					X	X LSB

Ejemplo:

POSICIÓN	COD. MÁQUINA	LENG. ASSEMBLY
840B	69 A2	ADC #$A2

	ANTES	DESPUÉS	
PSR	00????01	11????00	
A	16	B9	
$840B	69	A2	$840C
Memoria datos			Memoria programas

ADC A, — ADD CON ARRASTRE — Z80

Inmediato —69 (2 bytes)

El flag de arrastre más el contenido del byte siguiente al opcode son sumados al contenido del acumulador.

EFECTO EN EL PSR

	S	Z	H	V	N	C
MSB	X	X	X	X		X LSB

Ejemplo:

POSICIÓN	COD. MÁQUINA	LENG. ASSEMBLY
840B	CE B9	ADC A,$B9

	ANTES	DESPUÉS	
PSR	00?0?001	10?0?1?0	
A	24	DE	
		CE	$840B
		B9	$840C
Memoria datos			Memoria programas

Informática pop

Virgin Games ha establecido significativos lazos entre el mercado discográfico y el de software de juegos

A principios de la década de los setenta, cuando el mercado de software para ordenadores personales era una incógnita, algunos jóvenes emprendedores tuvieron en Gran Bretaña muchas posibilidades de obtener beneficios de la demanda de software en cassette. Todo el que fuera capaz de escribir divertidos programas de juegos en BASIC podía conseguir un magnetófono de alta velocidad para grabar de cassette a cassette y ofrecer su venta por correspondencia en los anuncios del diario local.

Actualmente las diversas empresas de software crean sus productos de distintos modos. Por ejemplo, Imagine Software (véase p. 559) cuenta con muchos programadores de plantilla que codifican las ideas creativas de sus jefes.

Nick Alexander, el joven director-gerente de Virgin Games, afirma: "Con frecuencia, cuanto mejor es el programador, menos ideas tiene. Para ser un buen programador hace falta ser muy lógico, muy metódico, muy diligente, y no suelen ser éstas las cualidades de un individuo creativo". Por ese motivo ofrece un servicio técnico y creativo para corregir las deficiencias de los múltiples programas que cada semana les envían jóvenes promesas. Se ayuda a los programadores capaces a desarrollar ideas y a los creativos en la codificación.

La recompensa por ser editado por Virgin podría parecer inferior a las de otras empresas. Un juego publicado por Virgin obtiene para su autor un anticipo que media entre mil y tres mil libras esterlinas, además del 7,5 % en derechos de autor sobre el precio neto. Compare estas cifras con el 25 % de derechos de autor que muchos otros editores de software afirman ofrecer. Pero Alexander sostiene que debido a que cerca del 25 % de los ingresos netos de cualquier juego son reinvertidos en su promoción, las ventas son muy superiores.

La promoción de productos es, sin lugar a dudas, una actividad que Virgin conoce al dedillo. La marca Virgin se estableció gracias a sus empresas de éxito en la industria musical y las técnicas que aprendió en ese campo Richard Branson, jefe de Virgin de treinta y un años, han sido aplicadas su ramificación en la esfera del software. Los autores de juegos son promovidos como estrellas por derecho propio: la información que aparece en cada cassette no sólo pone el nombre del autor, sino también una foto y una breve biografía. Virgin Games, que comenzó en febrero de 1983 solicitando juegos en las revistas dedicadas a los ordenadores personales, recibió alrededor de 500 ofertas iniciales. Ahora en su lista figuran 46 títulos para ocho ordenadores personales. Sus mayores ventas corresponden al Spectrum; el BBC Micro ocupa el segundo lugar, seguido por el Commodore 64.

El más reciente escritor de software contratado por Virgin es Martin Wheeler, un chico de quince años que acaba de crear dos nuevos juegos para el Spectrum, titulados *Dr Franky and the Monsters* (El doctor Franky y los monstruos) y *Sorcerers* (Brujos). Wheeler ha ensamblado los programas en lenguaje máquina y desarrollado algunos elementos gráficos impresionantes. Alexander encuentra semejanza entre el programa de los ordenadores personales de hoy y el negocio musical de hace una década, y se muestra convencido de que la informática está en vías de desplazar a la música como actividad de tiempo libre de la juventud.

Nick Alexander

Richard Branson

Trabajo en la sombra

Los ordenadores cuentan con un programa incorporado que, en silencio, ejecuta las rutinas de funcionamiento de la máquina

Paul Chave

El "administrador"
Todos los ordenadores poseen alguna forma de sistema operativo, un programa que administra el funcionamiento del ordenador y controla todos los dispositivos conectados al sistema

El sistema operativo es un componente básico de todo ordenador, porque constituye un vínculo entre el hardware y el software. Sin embargo, como los sistemas operativos realizan la mayoría de sus tareas inadvertidamente, para muchas personas, en particular los usuarios de ordenadores personales, su labor es casi ignorada. Por este motivo es necesario presentar un panorama de ellos.

Los lenguajes de alto nivel permiten que el programador quede aislado del funcionamiento interno de la CPU y posibilitan una mayor portabilidad de programas entre un sistema y otro. Siempre y cuando un lenguaje esté razonablemente estandarizado, las instrucciones deberían funcionar en cualquier máquina que dispusiera del lenguaje. El intérprete o el compilador que procese el código fuente del lenguaje de alto nivel se encarga de los detalles de distribución de la memoria y demás. El intérprete o compilador de alto nivel también es un programa y se debe cargar en la memoria principal para que pueda así convertir el código fuente de alto nivel a instrucciones en código objeto listas para su ejecución. Los distintos BASIC en ROM ya están presentes, de forma permanente, en la memoria, y están listos para ser utilizados tan pronto como se conecte la máquina.

Sin embargo, para que un ordenador funcione no basta con disponer de un programa dentro de él que sea capaz de convertir código fuente en lenguaje máquina. Es necesario que haya aún otro programa funcionando en las sombras que se ocupe de la "administración doméstica". Como ejemplo de esta administración interna, consideremos el problema de conseguir que una letra digitada en el teclado aparezca en la pantalla. En algún lugar de la memoria ha de haber un programa que le diga constantemente a la CPU que verifique el teclado para ver si se ha pulsado alguna tecla. Si se hubiera pulsado alguna, el programa debería determinar de qué tecla se trata y luego instruir al sistema de circuitos de video para que produzca el patrón de puntos correcto, en la secuencia apropiada, para la salida a la pantalla. Se dice que las actividades como ésta son "transparentes" para el usuario.

Del mismo modo, cuando se emite una orden como CSAVE para guardar un archivo en cassette, el programador no se ha de preocupar por cómo se convierten los datos a la forma adecuada para el almacenamiento en cassette: todo forma parte del sistema operativo.

El sistema operativo es el *programa background*, es decir, de baja prioridad que se ejecuta constantemente y que supervisa todo lo demás. Cuando el ordenador en cuestión es un pequeño sistema informático basado en ROM con un BASIC incorporado, surge una pequeña fuente de confusión, porque con frecuencia el BASIC y el sistema operativo están retenidos en la misma ROM. En su forma más sen-

cilla, entonces, una ROM interna contendrá todo el software necesario para que el sistema funcione, aparte de los programas para aplicaciones (juegos, tratamiento de textos, etc.) cargados o escritos por el usuario. Parte de esta ROM contendrá el código necesario para convertir los programas de aplicaciones escritos en BASIC a lenguaje máquina (el *intérprete*); otra parte contendrá el código necesario para introducir y modificar los programas escritos por el usuario (el *editor*); y otra parte contendrá el software de administración interna necesario para cuidar del teclado, visualizar caracteres y gráficos, aceptar datos provenientes de cassettes y destinarlos a las partes pertinentes de la memoria, etc. (el *monitor*).

El término *monitor*, que no se debe confundir con un televisor o un monitor para visualización, es un sinónimo aproximado de "sistema operativo". En su forma más sencilla, el monitor puede hacer apenas poco más que aceptar instrucciones en lenguaje máquina, colocarlas en la posición de memoria adecuada y supervisar su ejecución. Cuando la administración interna se convierte en algo un poco más avanzado que esto, se tiende a aludir al propio monitor como al sistema operativo.

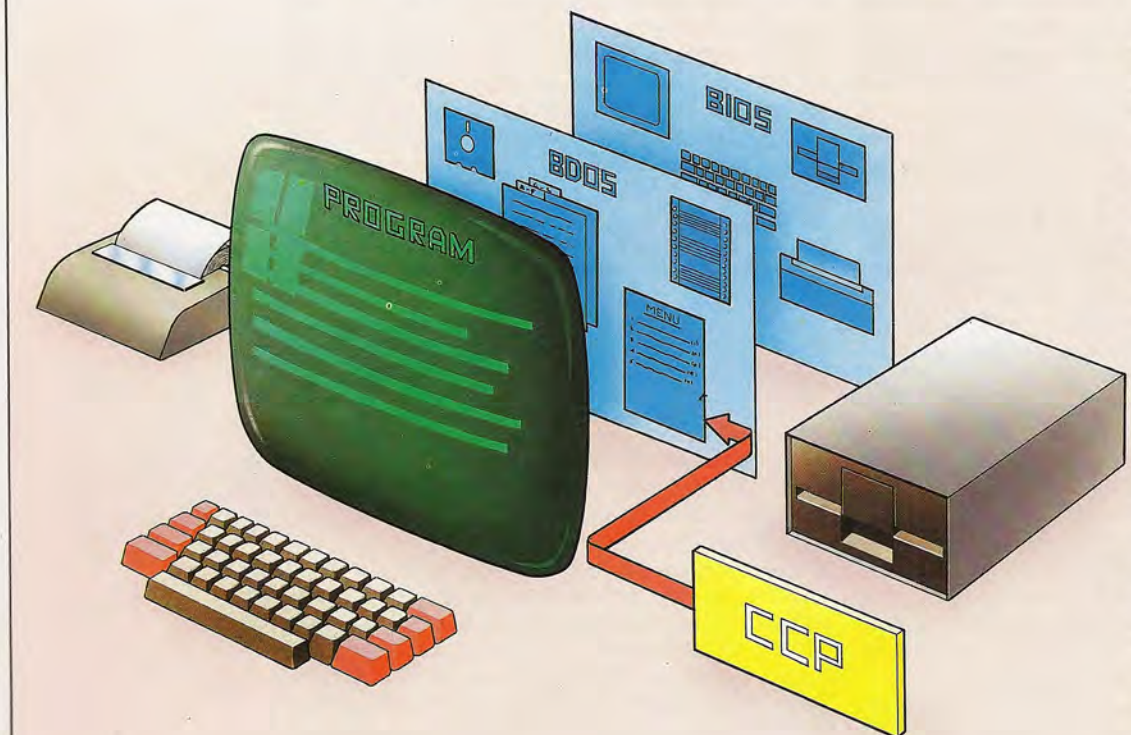
En el otro extremo están los ordenadores basados en disco, que a menudo se utilizan en oficinas como pequeños sistemas de gestión, y que poseen poderosos sistemas operativos. Antes de considerar ordenadores intermedios, como el Apple, vamos a analizar el tipo de sistema operativo que necesita un sistema sólo de disco.

Un sistema informático basado íntegramente en discos flexibles para su software, por lo general, tendrá muy poco almacenado en ROM con carácter permanente, aparte de un cargador *bootstrap* y algunas rutinas de administración interna. Cuando se conecta un ordenador de este tipo, el cargador *bootstrap* sólo contiene el código de lenguaje máquina suficiente para indicarle a la CPU cómo acceder a una unidad de disco y cargar el sistema operativo en la memoria principal.

El sistema operativo cargado en RAM ha de ser capaz de hacer más que los sistemas operativos de sistemas más convencionales basados en ROM. Se convierte en lo que denomina un *DOS* o *sistema operativo en disco*. Un sistema operativo como éste amplía sus funciones normales de administración interna mediante la adición de órdenes que actúan directamente en los archivos almacenados en el disco. Un sistema operativo se espera que incluya órdenes para listar los nombres de los archivos almacenados en el disco, borrar nombres de archivo o cambiarlos por otros, y copiar archivos de disco en la memoria principal o en otros discos.

Los sistemas operativos de sistemas más sencillos, basados en ROM, normalmente no poseen órdenes para manipulación de archivos tan sofisticadas y podrían no tener más que una orden simple para cargar un archivo con nombre desde cinta o para almacenar un archivo en cinta bajo un nombre determinado. Un sistema operativo sofisticado sabrá la dirección exacta del disco o cinta donde se encuentra almacenado cualquier archivo. Los sistemas operativos menos avanzados sólo son capaces de explorar en todos los archivos presentes hasta hallar el del nombre buscado, y después cargarlo: o escribir un archivo bajo un nombre de archivo dado en la cinta en el punto, cualquiera que fuera, en donde se encontrara la cinta en el momento de emitirse la orden.

Un buen ejemplo de sistema informático intermedio entre los dos anteriores es el BBC Micro, que es igualmente eficaz en la manipulación de archivos en cassette o en disco, utilizando en gran parte las mismas órdenes. El sistema operativo resi-



Controles de operación

El CP/M está dispuesto en capas para posibilitar que tanto él mismo como los programas para él escritos se trasladen fácilmente de un micro a otro. El programa del usuario funciona a través del BDOS (sistema operativo en disco básico), conjunto de rutinas que proporcionan funciones estándar y facilidades para el programa. El BDOS es el mismo en todas las máquinas, de modo que un programa que lo utilice funcionará en cualquier otro sistema CP/M. Cuando el BDOS necesita realizar una tarea determinada, llama al BIOS (sistema básico de entrada-salida). Éste se adapta especialmente a cada máquina y es el vínculo de software entre el BDOS y los componentes físicos del ordenador. El CP/M también incluye el CCP (*Command Control Program*: programa de control de órdenes), que permite que el usuario dé entrada a órdenes para ejecutar programas, cambiar el nombre de archivos y borrarlos, y, en general, controlar el sistema informático

Kevin Jones

de en una ROM, pero se trata de una ROM físicamente separada y se puede pensar en ella como un sistema operativo que se ha cargado en la memoria desde un dispositivo de memoria externo. Sus funciones incluyen órdenes para manipulación de archivos considerablemente más avanzadas que aquellas de que disponen otros ordenadores con ROM solamente, como el Spectrum.

El sistema operativo que utiliza un ordenador se puede, entonces, considerar como un programa que posee la función de situarse entre el usuario y el resto del sistema informático, incluyendo su CPU, su software de sistemas (como los lenguajes de programación) y el software de aplicaciones.

Compatibilidad

La capacidad de utilizar el software en más de un sistema de ordenador se conoce por compatibilidad. En términos amplios, ésta tiene dos aspectos. El primero de ellos es el hecho de que procesadores diferentes exigen juegos de instrucciones diferentes para realizar operaciones equivalentes. Por consiguiente, las instrucciones en lenguaje máquina para, supongamos, sumar los contenidos de dos posiciones de memoria, tendrían una forma si se escribieran para el 6502 (utilizado en el Apple) y otra forma totalmente distinta si se requiriera la misma operación en un ordenador Z80 como el Spectrum. El problema que entraña convertir código de alto nivel a código de lenguaje máquina idóneo es, no obstante, responsabilidad del intérprete o el compilador empleados. Para cada CPU diferente se han de escribir intérpretes y compiladores diferentes.

Existe, no obstante, un problema separado que afecta a la compatibilidad del software. Aun cuando se utilice la misma CPU, como en el Apple y el BBC, existen otras complicaciones. Se utilizan distintas posiciones de memoria para la memoria de video, se necesitan diferentes códigos para desplazar el cursor por la pantalla, se proporcionan distintos recursos de entrada y salida, y de este modo sucesivamente.

Para superar este problema se desarrollaron sistemas operativos en disco genéricos que permitieran que todo el software escrito, por ejemplo, para un ordenador Z80 basado en disco, se pudiera ejecutar en cualquier otro ordenador Z80 de similares características que tuviera el mismo sistema operativo. El más conocido de tales sistemas en disco es el CP/M (*Control Program/Microcomputers*: programa de control para microodenadores).

Los sistemas operativos en disco como el CP/M son esencialmente un desarrollo de los monitores y sistemas operativos más específicos para una sola máquina, pero representan un importantísimo avance en términos de compatibilidad de software. Cualquier programa escrito para funcionar con un sistema operativo genérico como el CP/M o el MS-DOS funcionará en cualquier otro ordenador con ese sistema operativo, siempre y cuando el software no intente hacer uso de ninguna configuración especial (como, p. ej., efectos sonoros) específicos de una máquina. El software del sistema operativo llega a manos del fabricante de ordenadores en forma estándar, suministrado directamente por quienes lo desarrollan. Todo lo que el fabricante de hardware ha de hacer es reescribir una pequeña porción, dependiente de la máquina, del programa.

Ian McKinnell

La clave del éxito
El Osborne 1 debe gran parte de su éxito al hecho de ser un ordenador basado en CP/M. Viene con algunos de los programas CP/M más populares, incluyendo el WordStar, SuperCalc y MBASIC

Los sistemas operativos en disco varían considerablemente en cuanto a complejidad y capacidades, pero los más simples, como el CP/M y el MS-DOS, constan esencialmente de tres partes: el procesador de órdenes, el sistema operativo en disco básico (*Basic Disk Operating System*: BDOS) y el sistema básico de entrada-salida (*Basic Input/Output System*: BIOS). De estos componentes, el único que tiene relación con la compatibilidad es el BIOS. Éste es una parte separada del programa que contiene todas las rutinas necesarias para manipular los periféricos, incluyendo la pantalla y el teclado, y su configuración debe ser especial para cada nuevo diseño de ordenador. Todo programa que se ejecute bajo el control de este sistema operativo se conectará en interface con el ordenador a través del BIOS. Éste, entonces, asumirá funciones como recoger los caracteres del teclado, dar salida a los caracteres hacia la pantalla o la impresora, direccionar los discos, etc.

El BDOS, por su parte, consta de los componentes del sistema operativo que no son específicos de los dispositivos (es decir, rutinas generalizadas para manipular la pantalla, la impresora, las unidades de disco, etc.).

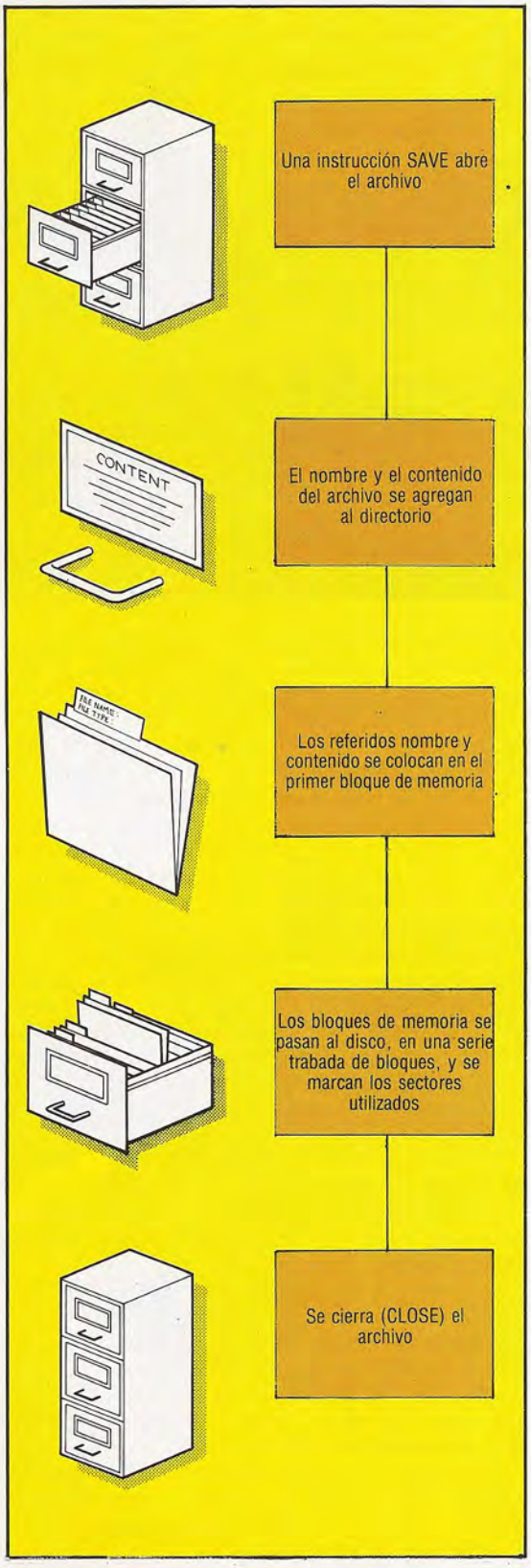
El procesador de órdenes es la parte del programa que manipula las órdenes para el sistema operativo digitadas en el teclado. Las órdenes usuales incluyen aquellas que cargan archivos desde el disco a la memoria principal, listan los nombres de archivo presentes en el disco y borran o cambian los nombres de los archivos del disco.

Dado que el sistema operativo lleva a cabo su trabajo en la sombra, a menudo se lo suele pasar por alto, cuando en realidad es una parte esencial de todo sistema informático.

Amos de la memoria

Iniciamos una serie dedicada al estudio de los archivos: qué son, cómo se emplean, de qué métodos se dispone para acceder a ellos

Cómo se guarda un archivo binario
Un archivo binario es simplemente la copia de un trozo de la memoria. Puede contener tanto un programa que encierra la memoria, como una imagen que aparece en pantalla, como cualquier otra cosa. El proceso archivador es inmediato: tras ser apuntada su entrada en el directorio, los datos de archivo se escriben como una cadena de sectores trabados entre sí. El DOS también llevará una lista de estos sectores, de modo que cuando haya que incorporar otro archivo no se escriba sobre los utilizados



Archivo es la palabra justa cuando se habla del almacenamiento informático, puesto que se pueden establecer analogías directas con el método de almacenar documentos e informes en un sistema de archivo tradicional. Teniendo esto presente, veamos primero por qué son necesarios los sistemas de archivo en los ordenadores personales.

Con el fin de "administrar" eficazmente nuestra vida cotidiana, es necesario mantener el registro más exacto posible de nuestras experiencias, transacciones monetarias, citas sociales, etc. La mayoría de las personas utiliza un diario-agenda y lleva el estado de una cuenta bancaria a través de los resguardos de un talonario. Esta necesidad de almacenar información y, lo más importante, de recuperarla fácil y simplemente, se amplía muchas veces cuando hay que tratar con una gran cantidad de información constantemente cambiante, típica de cualquier empresa o proyecto que abarque distintas personas, lugares, objetos y circunstancias. A medida que estas empresas van creciendo, el manejo de la información se va haciendo cada vez más complejo, y muchos de los problemas con que se encuentran las empresas de este tipo surgen por causa de su mala administración y una interpretación errónea de la información. Un sencillo método para almacenar, llevar un índice y recuperar los datos puede ser la clave de una buena administración. Los directores valoran la necesidad de buenas técnicas de archivo, y ajustan los métodos de almacenamiento de acuerdo con el tipo, el volumen y el porcentaje de variación de la información que tienen bajo su control.

Estos principios son válidos cuando los aplicamos a los sistemas de que dispone un ordenador para manipular y almacenar con precisión enormes cantidades de información a una velocidad increíblemente elevada. En el centro de un sistema de este tipo se halla el "administrador" del ordenador: el sistema operativo en disco (DOS). Éste trabaja de manera perfecta siempre que su "jefe" (es decir, el usuario o su programa) le proporcione la información correcta y le formule las preguntas adecuadas. Por tanto, los sistemas de almacenamiento por ordenador son eficaces si también lo son la organización de los datos (o sistema de archivo) adoptada por el DOS y la forma como los utiliza el DOS.

Los métodos estándar de tratamiento de archivos que utilizan los microordenadores en un principio se pensaron para ordenadores de unidad principal y miniordenadores. Se dividen en tres clases: archivos binarios, archivos secuenciales y archivos de acceso directo. Vamos a analizarlos por separado.

Archivos binarios

Un archivo binario es, simplemente, la copia de una porción de la memoria reservada al usuario. Un buen ejemplo sería un programa guardado (SAVE). Imaginemos esa zona de la RAM como un

Directorio del disco

En la pantalla se muestra un directorio del disco producido por la utilidad STAT del CP/M, con mucha más información que la mayoría de las visualizaciones de directorio

Regs
El número de registros del archivo puede variar; aquí los registros tienen capacidad para 128 bytes

Bytes
Longitud en Kbytes

Ext
La extensión es una medida alternativa del espacio del disco que ocupa el archivo

Acc
Acceso: un archivo puede servir para lectura y escritura (R/W) o sólo para lectura (R/O).

Nombre archivo
Incluye primero el nombre de la unidad (A: o B:), seguido del nombre del archivo (AUTOST, p. ej.), y después la extensión del archivo (.COM, p. ej.), que puede consignar información relativa al contenido del archivo

```
 Recs  Bytes  Ext Acc Filename
    0     0K    1 R/W B:ACNTLIST.DTA
   11     2K    1 R/W B:ANT
   10     2K    1 R/W B:ANT.BAK
   64     8K    1 R/W B:ASM.COM
   16     2K    1 R/O B:CODELIST.DTA
   16     2K    1 R/W B:AUTOST.COM
    1     1K    1 R/W B:COMPANY.DTA
    2     1K    1 R/W B:CONTROL.DTA
   34     5K    1 R/W B:COPY.COM
   40     5K    1 R/W B:DDT.COM
    4     1K    1 R/W B:DUMP.COM
  250    32K    2 R/W B:INSTALL.COM
    4     1K    1 R/W B:JUNK
   16     2K    1 R/W B:LOAD.COM
    6     1K    1 R/W B:MICROLIN
   40     5K    1 R/W B:ML.COM
   86    11K    1 R/W B:MOVCPM.COM
   58     8K    1 R/W B:PIP.COM
    2     1K    1 R/W B:SCREEN.ASM
    1     1K    1 R/W B:SCREEN.COM
    4     1K    1 R/W B:SCREEN.DOC
   42     6K    1 R/W B:STAT.COM
Bytes Remaining On B: 85K
```

sencillo bloc de notas. Si a usted le interesara conservar las notas principales o los apuntes más interesantes, arrancaría las páginas correspondientes y las conservaría en algún sitio donde las tuviera a mano. Los archivos binarios funcionan de la misma manera. Cuando se imparte una orden SAVE, el DOS almacena el NOMBRE ARCHIVO en disco, marcándolo de alguna forma especial como un archivo binario, y después copia el área correspondiente de la memoria byte por byte en el disco. El programa se almacena en bloques unidos (con los señaladores al final de cada bloque indicando dónde comienza el bloque siguiente). El último bloque termina con un señalador de final de archivo. Utilizando nuestra analogía, podríamos decir que nosotros le hemos puesto un nombre a una página de nuestro bloc de notas y a continuación hemos procedido a almacenarla en un cajón del archivador, añadiendo después el nombre de esa página al índice de contenidos del cajón.

Cuando se empieza (RETURN) o se acaba (ENTER) una línea de programa, ésta se comprime en una forma distintivada: el intérprete de BASIC toma las palabras clave y las codifica mediante números de un byte. Éstos se pueden manejar más fácilmente y decodificar cuando se reclama un listado del programa (LIST). Como un archivo binario es una imagen de la RAM, también puede almacenar códigos ASCII y datos en binario. La facilidad de guardar archivos en ASCII es útil, por ejemplo, para almacenar el contenido de la memoria de pantalla, de modo que las visualizaciones en pantalla se puedan guardar (SAVE) para volverlas luego a cargar (LOAD) en la misma zona de la memoria. Además, algunos sistemas operativos en disco y algunas versiones de BASIC permiten almacenar un programa en forma ASCII. Ello posibilita editar el programa no distintivado como un archivo de texto en las máquinas con programas de muy elaborada edición.

Los archivos binarios son muy sencillos de utilizar y administrar, pero están limitados por dos factores. El primero de ellos es que sólo se puede guardar la información correspondiente como un todo. Por consiguiente, la información se debe recuperar de la misma manera y, en consecuencia, los archivos binarios se deben cargar otra vez en la memoria en su totalidad. En segundo lugar, las dimensiones máximas de un archivo quedan en todo caso limitadas por la cantidad de RAM disponible para el usuario.

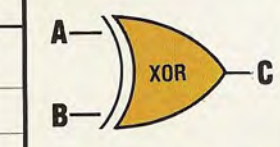
En el próximo capítulo de este curso analizaremos los archivos secuenciales, que permiten que un archivo sea tan largo como se necesite (dentro de los límites del espacio del disco) y los archivos de acceso directo, que emplean métodos diferentes para permitir que el DOS almacene los datos de tal forma que luego se puedan recuperar y actualizar libremente.

Pares y nones

Nuestro curso de lógica nos permite ahora diseñar circuitos bastante complejos. Ahora vamos a analizar otros dos, que figuran entre los más importantes de un ordenador

Antes de empezar a analizar el diseño de estas dos aplicaciones avanzadas, primero analizaremos con todo detalle otra puerta lógica importante: la puerta OR-exclusivo (XOR). Esta puerta ya la consideramos someramente (véase p. 527), pero todavía no habíamos ofrecido ni su símbolo de álgebra booleana ni el de su diagrama de circuitos:

Tabla de verdad			Símbolo de circuito
A	B	C	
0	0	0	
0	1	1	
1	0	1	
1	1	0	

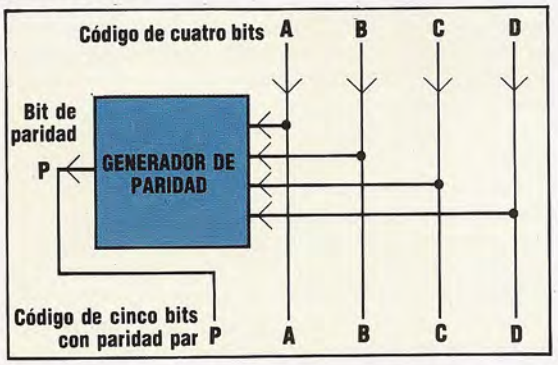


A partir de la tabla de verdad se puede ver que la salida C se puede expresar de dos formas:

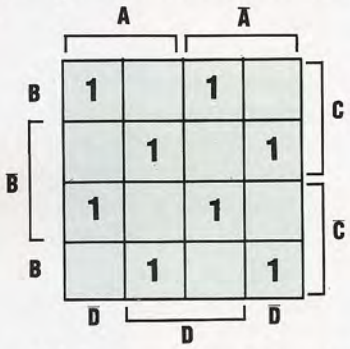
a) $C = A \oplus B = \overline{A}.B + A.\overline{B}$
b) $\overline{C} = A \oplus B = A.B + \overline{A}.\overline{B}$

La segunda expresión se forma considerando los casos en los que C no es uno (es decir, es cero). Esta puerta puede ser de particular utilidad para nuestra primera aplicación.

Un generador de bits de paridad



El de paridad es un concepto importante en el diseño de sistemas de transmisión de datos. El bit de paridad (véase el diagrama) se agrega a los restantes bits de un código en binario con el fin de que todos los códigos binarios transmitidos posean un número par de unos. (Otra convención podría ser que todos los códigos contengan un número impar de unos, lo que se conoce por *paridad impar*.) Un bit de paridad actúa como un sistema de control para verificar si se ha producido la transmisión correcta. El circuito que diseñaremos acepta códigos de cuatro bits y produce un bit de paridad adecuado. Con una pequeña modificación el circuito también podría actuar como un verificador de datos entrantes. La tabla de verdad para este circuito está consignada al margen. La representación de estos valores en diagrama de Karnaugh nos da:



Esta figura del diagrama K por desgracia no es simplificable a la manera habitual, porque no se ha formado ningún grupo. La expresión para P es:

$$P=\overline{A}.\overline{B}.\overline{C}.D+\overline{A}.\overline{B}.C.\overline{D}+\overline{A}.B.\overline{C}.\overline{D}+\overline{A}.B.C.D$$
$$+A.\overline{B}.\overline{C}.\overline{D}+A.\overline{B}.C.D+A.B.\overline{C}.D+A.B.C.\overline{D}$$

Si ahora procedemos a agrupar entre sí los términos rojos y los términos azules, nos es posible simplificar la expresión:

$$P=(\overline{A}.\overline{B}+A.B).(\overline{C}.D+C.\overline{D})$$
$$+(A.\overline{B}+\overline{A}.B).(\overline{C}.\overline{D}+C.D)$$

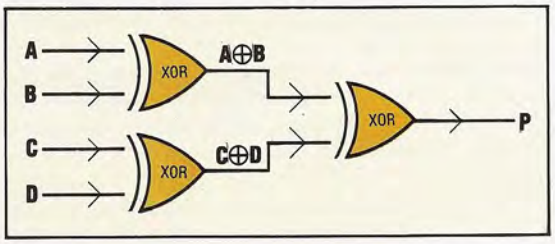
Ahora, remitiéndonos a las expresiones a) y b) para una puerta XOR que presentamos al comienzo de este artículo, podemos simplificar más la expresión para obtener:

$$P = (\overline{A \oplus B}).(C \oplus D) + (A \oplus B).(\overline{C \oplus D})$$

Considerando cada término entre paréntesis como una entrada para una puerta XOR, la expresión se podría reducir aún más:

$$P = (A \oplus B) + (C \oplus D)$$

y formar el circuito como una cascada de puertas XOR:

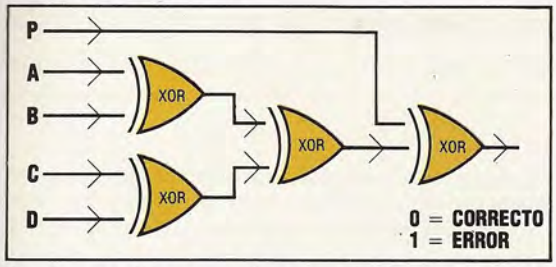


A	B	C	D	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Tabla de verdad de un GBP

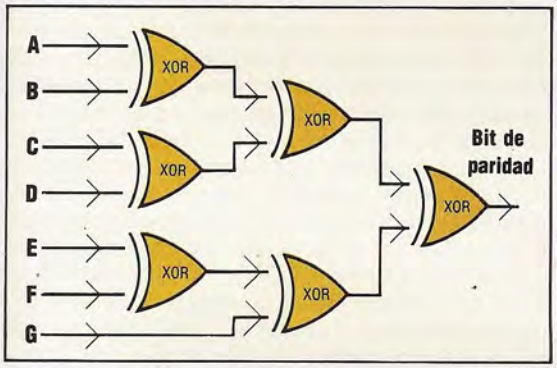
Este circuito se modifica para que actúe como control de paridad, añadiendo otra puerta XOR para comparar el bit de paridad recibido con uno generado por el circuito en el extremo de recepción.

Código recibido de cinco bits



En la práctica, para la transmisión de datos la mayoría de los ordenadores utilizan el código ASCII, que tiene ocho bits, siete de información y uno de paridad par. Es fácil concebir un generador de bits de paridad para códigos ASCII:

Código de información de siete bits



Un codificador de prioridad

Muchos ordenadores emplean sistemas "interruptores de prioridad" para controlar el flujo de datos hacia y desde dispositivos periféricos. En estos sistemas la operación de la CPU se interrumpe en virtud de una señal proveniente del periférico cuando éste necesita la atención de la CPU. Sin embargo, cuando dos o más periféricos interrumpen a la CPU al mismo tiempo, ésta ha de seguir un orden de prioridad para "atender" primero al dispositivo más importante. El codificador de prioridad que nosotros diseñaremos englobará cuatro dispositivos periféricos en un circuito que pueda identificar cuál es el periférico que está produciendo señales.

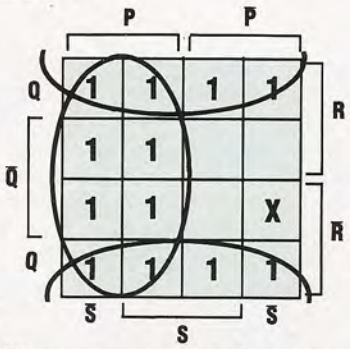
Para obtener información en uno de los cuatro dispositivos se necesitan dos líneas de salida, más una tercera línea de salida para indicar que se requiere una interrupción. Supongamos que los cuatro periféricos son P, Q, R y S, siendo P el de mayor prioridad y S el de menor prioridad. Las líneas de salida se denominarán A y B, para identificar al periférico, y Z para señalar una interrupción. La tabla de verdad para el codificador se puede realizar utilizando una X para aquellas situaciones ante las cuales el codificador "se muestra indiferente".

P	Q	R	S	A	B	Z
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Para ayudarle a comprender cómo se ha hecho esta tabla, observe la última fila. Aquí P está indicando una interrupción, y como P es el dispositivo de máxima prioridad, a nosotros no nos preocupa que los dispositivos de inferior prioridad señalen su prioridad o no.

Las tres salidas del circuito, A, B y Z, se deben analizar de forma independiente. Empezando por A, el diagrama K es:

Para A



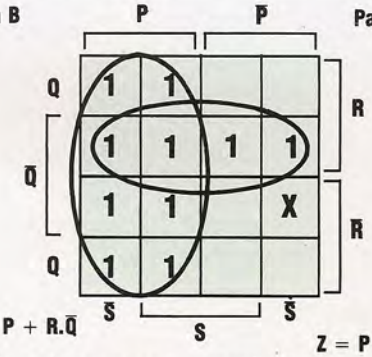
A = P + Q

El caso "indiferente" en la salida A está representado mediante una X (pero los casos "indiferentes" en las entradas se tratan de distinta forma). Tomemos el caso en el que P es uno y Q, R y S son "indiferentes". Aquí debemos rellenar todos los casilleros del diagrama K donde P sea uno; hay ocho en total. Del mismo diagrama sacamos la expresión simplificada:

$$A = P + Q$$

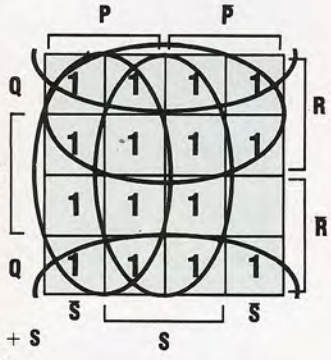
Del mismo modo, para B y Z los diagramas K son:

Para B



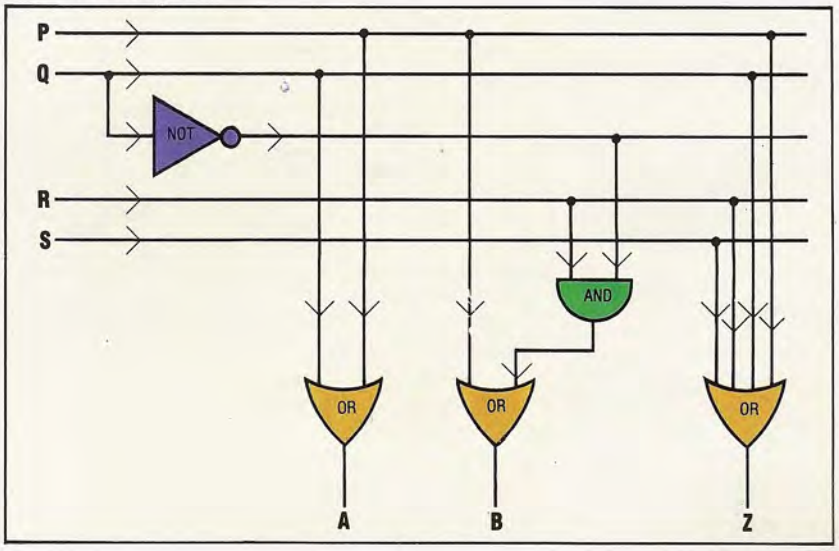
B = P + R.Q̄

Para Z



Z = P + Q + R + S

Utilizando estas tres expresiones, llegamos al diseño de este circuito:



Contar los pasos

Los contadores son registros que desempeñan un papel fundamental en las reiteraciones

Un contador es un registro que sirve para indicar las veces que se ha repetido el ciclo de operaciones de un bucle. Veamos su importancia:

La figura 1 muestra cómo tras la introducción de un número por teclado se visualiza el total resultante de sumarlo con los que se introdujeron antes que él. En este ejemplo, ningún registro nos indicará cuántos números se desea entrar. De aquí que se haga una pregunta tan vaga como la de que si quedan más números, para poder salir del bucle. Este planteamiento es válido para casos en que se desconoce el número exacto de cantidades a las que se va a dar entrada. Pero si ya se conoce de antemano de cuántas cifras se va a disponer, puede que resulte más cómodo contar cuántas veces habremos de repetir un mismo ciclo de operaciones, utilizando para ello un contador.

La figura 2 representa el mismo ejemplo anterior pero sabiendo que hay un total de 10 números a sumar. Se observa la utilización de un registro llamado contador, que se pone a 1 en un momento determinado del programa y, tras la operación de sumar el total actual más el número recién introducido por teclado, se incrementa en una unidad. Acto seguido se controla su valor. Caso de que el contador no haya rebasado el número 10 (cantidad de números que deben entrarse), el flujo retorna al principio del bucle. Sin embargo, al llegar al valor 11 se cumple la condición de que el contador es mayor de 10 y se baja en secuencia, abandonándose el bucle a la décima reiteración. Se ha conseguido detener un ciclo reiterativo mediante el control de las veces que debía repetirse, número de veces que coincide con el valor límite contenido en el contador.

El valor de un contador no necesariamente debe ser objeto de incremento, ya que también se puede partir de un valor inicial superior e ir decreciendo, también de uno en uno, con lo cual el valor final que deberá controlarse será 0, tal y como muestra la figura 3. También es claro que a un contador lo podemos hacer crecer o bien decrecer no solamente a "golpes" de unos. Con la expresión CONT = CONT + 3, por ejemplo, haríamos que el contador aumentara de tres en tres. Así, podría definirse el contador como un registro cuyo valor aumenta o decrece en cantidades constantes, de uno en uno, de dos en dos, de diez en diez, etc.

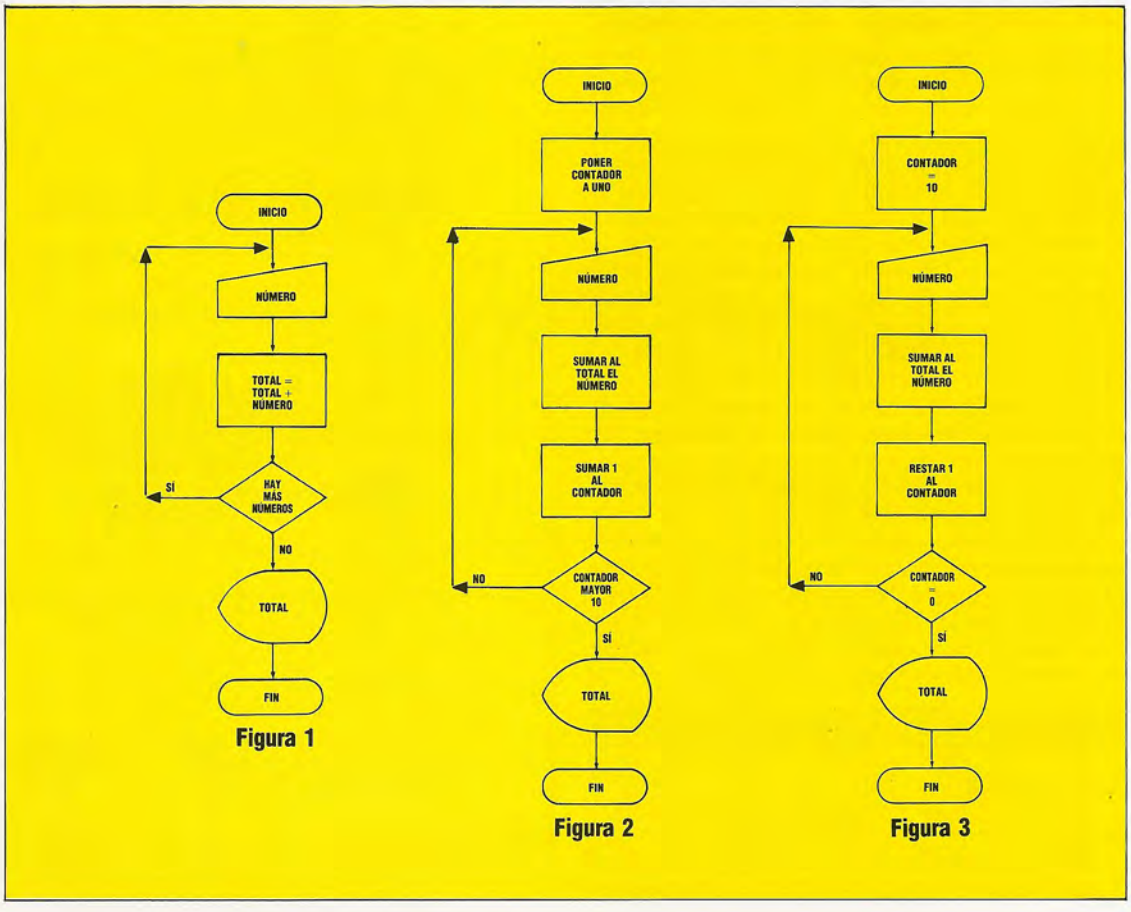


Figura 1

Figura 2

Figura 3

Atari se pone al día

Atari ha actualizado sus modelos 400 y 800, lanzando al mercado los ordenadores XL, más baratos y con atractivas configuraciones

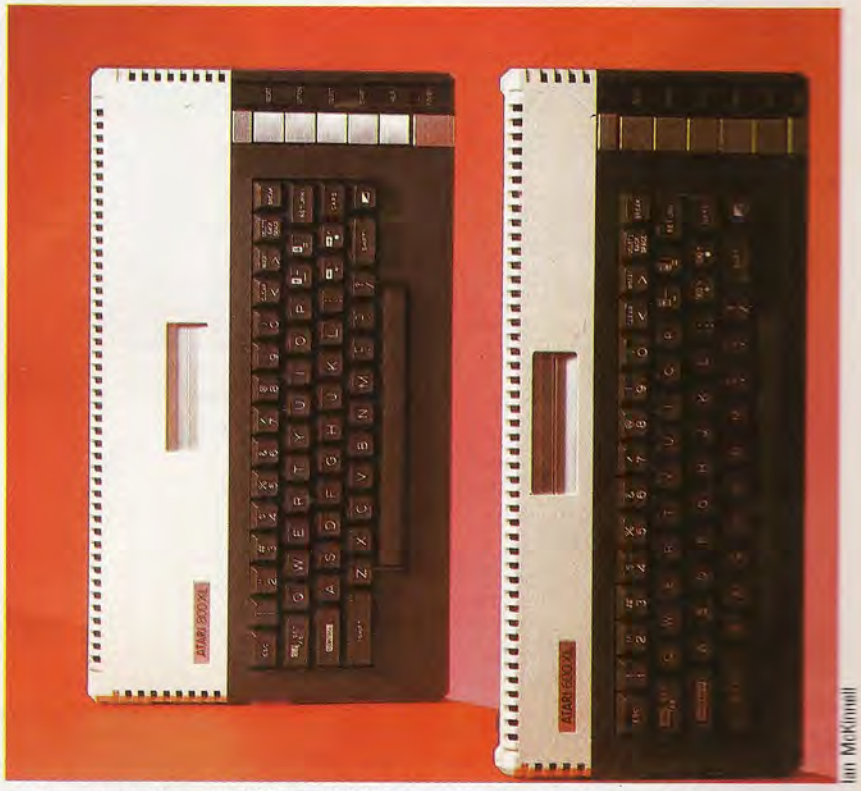
Enfrentada a la creciente competencia existente entre los fabricantes de ordenadores personales, Atari volvió a diseñar su línea de ordenadores y presentó el 600XL y el 800XL. Las nuevas máquinas pueden utilizar el software escrito para el 400 y el 800, que comprende una amplia gama de programas, desde los juegos más vendidos hasta paquetes de gestión.

Afortunadamente, ambas máquinas han adoptado el teclado del 800. El Atari 400 posee un teclado de tipo membrana, similar al del ZX81, con sus caracteres impresos sobre una membrana plástica plana. Debido al pequeño tamaño de las teclas y al grado de presión necesario para hacer contacto, la escritura al tacto es virtualmente imposible. Por el contrario, el 800 posee un teclado de tamaño natural, similar al de una máquina de escribir, que es uno de los mejores del mercado.

Los teclados de las nuevas máquinas son idénticos y poseen un total de 62 teclas. Éstas incluyen cuatro teclas de función: START, SELECT, OPTION y RESET, situadas a la derecha del teclado. También hay una tecla HELP, que trabaja con un software nuevo y proporciona útiles sugerencias en la pantalla. Hay 29 teclas para gráficos, y los de Atari poseen un juego de caracteres ASCII completo incorporado. Una original característica, heredada del 800, la constituye la manera como se utilizan las teclas del cursor. Las flechas están impresas como el tercer carácter de cuatro teclas separadas. Para desplazar el cursor se debe mantener deprimida la tecla CONTROL mientras se pulsa la tecla de flecha adecuada. Aparte de esta extraña configuración, los teclados están bien diseñados y es fácil digitar en ellos.

El Atari 400 viene con 16 Kbytes de memoria para el usuario, mientras que el 800 cuenta con 48 Kbytes. Ambos ordenadores poseen ranuras para ampliación en el tablero del sistema a las que se puede acceder quitando la tapa de la máquina. Los modelos XL no se deberían abrir. En cambio, las puertas para ampliación están incorporadas en la parte exterior de la carcasa. La única diferencia significativa entre ellos es la cantidad de memoria que viene como estándar. El 600XL viene con 16 Kbytes y se puede ampliar a 64 Kbytes enchufando un paquete de ampliación. El 800XL viene con 64 Kbytes.

Las dos máquinas XL poseen una interface incorporada que se denomina *Expander*. Ésta es una puerta para ampliación de tipo bus que se puede conectar a diversos periféricos y paquetes de ampliación. Asimismo, hay una ranura para cartuchos de ROM situada detrás del teclado, para juegos en cartucho y otro tipo de software. El Atari 800 original posee dos ranuras para cartuchos, pero virtualmente no se ha escrito software para hacer uso de la segunda ranura, de modo que en la línea XL ésta se ha suprimido. El 400 y el 800 poseen cuatro puertas para palanca de mandos, pero el software disponible no justifica este número, de modo que se han reducido a dos en las máquinas XL y se han desplazado desde delante de la carcasa hasta uno de los lados.



Ian McKinnell

Los gemelos Atari
Los ordenadores personales 600XL y 800XL de Atari son muy similares, pero el 600XL cuenta con 16 K de memoria, mientras que el 800XL dispone de 64 K. Las máquinas poseen un teclado de gran calidad y buenos gráficos en color. Puesto que se trata de versiones mejoradas de los Atari originales, disfrutan de una amplia gama de software

Luz y sonido

Todos los ordenadores Atari se pueden conectar directamente a un aparato de televisión y, además, excepto el 400, se pueden conectar a un monitor para ordenador. Conviene recalcar que la visualización en pantalla de Atari es de calidad, incluso en un televisor. El juego de caracteres generalmente es fácil de leer y el contraste entre texto y fondo es bastante bueno. Hay varias opciones de color disponibles, pero la visualización de textos normal se compone de letras blancas sobre un fondo de color azul oscuro. Los ordenadores Atari visualizan un máximo de 40 columnas por 24 líneas de texto. Existen cuatro modalidades para texto con visualizaciones distintas.

Los Atari fueron de los primeros ordenadores que proporcionaron gráficos sprite. La función sprite de los mismos se denomina gráficos *player-missile* y la controla un chip especial llamado chip GTIA. Los "jugadores" son objetos creados mediante pixels. Una vez se ha determinado la forma

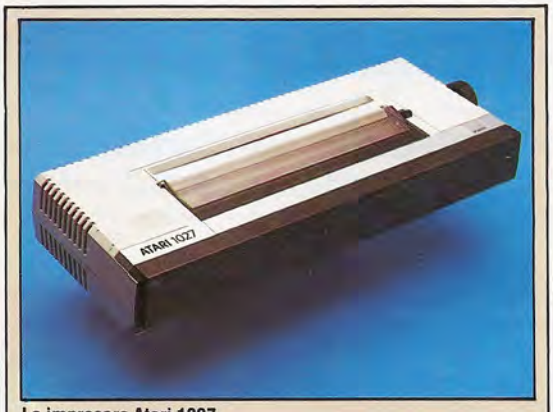
Unidad de disco
Una útil opción para el 800XL es la unidad de disco. Ésta proporciona 127 K de almacenamiento rápido. El 600XL estándar no posee memoria suficiente para utilizar la unidad de disco, pero se puede ampliar. Varios excelentes programas de juegos y de gestión se encuentran a la venta sólo en el formato de disco

de un jugador, los valores de cada pixel se colocan (POKE) en una zona de la memoria denominada *tabla de formas*. Uno puede crear hasta cuatro jugadores, cada uno de ellos con su correspondiente misil. Al jugador se le asignan uno o varios colores, y se lo manipula en la pantalla cambiando los colores de la tabla de formas. Aunque los gráficos *player-missile* de Atari no son fáciles de utilizar, facilitan notables visualizaciones en pantalla.

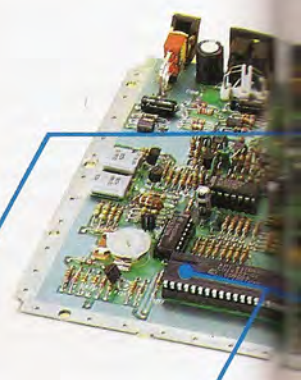
Las nuevas máquinas Atari poseen 11 modalidades para gráficos y hasta 256 colores (en realidad 16 colores, cada uno de ellos con 16 tonalidades distintas); sin embargo, debido a la cantidad de memoria que se requiere para la visualización en pantalla, el número de colores que se pueden mostrar varía según la resolución de la pantalla. Cuanto mayor sea la resolución, menor será la cantidad de colores que se puedan visualizar. En el 600XL y el 800XL la máxima resolución para gráficos es de 320 por 192 pixels.

Las configuraciones de sonido Atari también están controladas por un chip diseñado especialmente. Hay cuatro voces independientes, cada una de las cuales posee una escala de 3 ½ octavas. Las voces se pueden controlar a través de la orden SOUND en BASIC, o colocando (POKE) valores en los registros de la memoria que producen los diversos tonos. Los tonos también se pueden ajustar por oscilación, altura, distorsión y volumen. Cuando las voces se controlan mediante SOUND, sólo se puede emplear una voz cada vez. Ello significa que la armonización exige que se encienda cada voz por separado y la demora que hay es considerable. Este problema se puede superar empleando rutinas en lenguaje máquina o utilizando POKE en lugar de SOUND.

Los Atari 400 y 800 no tienen ningún lenguaje incorporado; se debe utilizar un cartucho separado. El 600XL y el 800XL poseen el BASIC Atari incorporado. Este BASIC no es el mejor, ya que carece de muchas de las configuraciones que hacen que el BASIC BBC y otros sean tan buenos. Por ejemplo, no tiene orden CIRCLE, ni la configuración PRINT@ ni PRINT USING, ni numeración o renumeración automática de líneas, ni provisión para variables de enteros. Sin embargo, se encuentran a la venta en cartuchos de ROM el BASIC Microsoft y el Extended BASIC.



La impresora Atari 1027
Posee una cabeza de impresión similar a la bola que usan algunas máquinas de escribir. Da una impresión tan buena como la de las máquinas eléctricas, pero es de operación más lenta. Existen otras dos impresoras Atari, una de las cuales usa lápices esferográficos para dibujar letras y líneas en cuatro colores; la otra es una impresora matricial que ofrece una impresión de inferior calidad, pero es rápida. Éstas son las únicas impresoras directamente utilizables con las máquinas XL, debido a la falta de una interface estándar para impresora



Conexión cartuchos
La gama XL posee una sola ranura para cartuchos de ROM

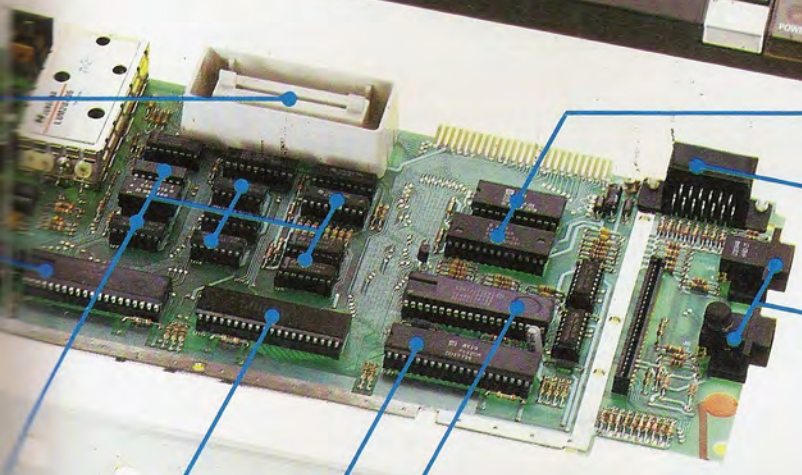
Chips de gráficos
Dos chips hechos a medida, llamados ANTIC y GTIA, proporcionan las espectaculares capacidades para gráficos de Atari

RAM
Numerosos chips componen los 16 K de RAM

Bola y palanca
Atari ofrece dos formas principales para controlar los juegos. El método convencional es mediante una palanca de mando, y la de Atari (a la derecha) se ha convertido en algo así como un estándar industrial. El mando de bola Atari (a la izquierda) controla a través de una bola que rota en su soporte

Para acompañar a la nueva línea de ordenadores, Atari ha vuelto a diseñar los periféricos existentes y ha incrementado el número de opciones disponibles para ampliación. Quizá el periférico más útil sea una caja de ampliación, que se enchufa en la interface *Expander*. La caja de ampliación proporciona ocho ranuras para ampliación, que pueden albergar fichas de interface para varios periféricos, dos puertas en serie RS232 y un bus en paralelo. Atari también posee un módulo CP/M con un microprocesador Z80, el sistema operativo CP/M 2.2 y una visualización conmutable de 40/80 columnas.

Con el 600XL y el 800XL Atari ha alcanzado cotas muy elevadas en cuanto a diseño, calidad de construcción y configuraciones. Ambos disponen de una vasta biblioteca de software en cartucho, cassette y disco que ha ido creciendo en los últimos años. Estas dos máquinas de Atari supondrán una adquisición muy atractiva para los entusiastas de los ordenadores personales.



ROM
Dos chips de ROM retienen el intérprete de BASIC

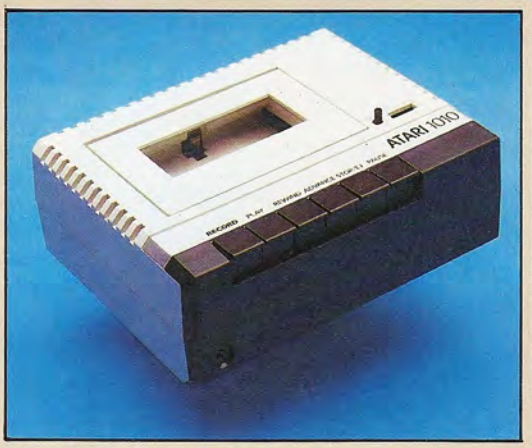
Conexiones periféricos
Se utiliza una puerta de 13 patillas para conectar los periféricos, incluyendo unidades de disco, impresoras y la grabadora de cassette exclusiva

Conexiones palancas de mando

Chip de sonido
Un chip construido a la medida y denominado POKEY se encarga de la generación de sonido

Chip de E/S
Un 6520 manipula las puertas de entrada y salida

CPU
El Atari se basa en un veloz chip 6502



La grabadora de cassette Atari
Los Atari sólo funcionan con su propia grabadora de cassette. Si bien ello encarece el precio del sistema, tiene sus ventajas. En primer lugar, al ser de Atari, la grabadora es más fiable. En segundo lugar, utiliza dos canales. Uno es para guardar programas en la forma normal; en el otro se puede grabar sonido. Esto permite que los programas para aprender idiomas reproduzcan fragmentos hablados en el momento preciso

ATARI 600XL/ 800XL

DIMENSIONES
600XL: 380 x 170 x 40 mm
800XL: 380 x 220 x 40 mm

CPU
6502, 2 MHz

MEMORIA
16-64 Kbytes de RAM, 24 K de ROM

PANTALLA
Hasta 24 filas de 40 columnas de texto, gráficos de hasta 320 x 192 con sprites y 16 colores con 16 niveles de brillo

INTERFACES
Palancas de mando (2), puerta para periféricos, puerta para ampliación, puerta para cartuchos

LENGUAJES DISPONIBLES
BASIC, FORTH, LOGO, PILOT, lenguaje Assembly 6502

TECLADO
Estilo máquina de escribir con 62 teclas, incluyendo teclas de cursor y de función exclusiva como SELECT y START para control de programas

DOCUMENTACION
Los manuales nunca han sido punto fuerte de Atari, ya que la empresa ha tendido a considerar sus ordenadores básicamente como máquinas para juegos y se ha saltado los detalles técnicos. No obstante, hay una enorme gama de soberbios manuales y revistas independientes, aunque éstos suponen un gasto extra para los usuarios de Atari

VENTAJAS
Los ordenadores Atari continúan ofreciendo lo mejor en cuanto a juegos y una amplia gama de software. Las nuevas máquinas también proporcionan excelentes opciones para ampliación

DESVENTAJAS
Los Atari pueden ser caros: se necesita una grabadora de cassette exclusiva y el precio del software suele ser elevado. Asimismo, la programación de gráficos y sonido es más difícil que en otras máquinas

El control del stock

La codificación es la base de aquellos sistemas de gestión que tratan con toda clase de detalles el almacén. Estudiaremos ahora las exigencias de diseño en un paquete concreto

Hemos estudiado con algún detalle las distintas formas de clasificar las existencias en el archivo de datos de almacén. Además de emplear sencillamente un número de código, el sistema debe permitir que el usuario grabe información en cada tipo de existencias.

Esta información necesitará clasificarse por categorías de datos relativamente constantes y la cantidad de datos se determinará en función de la capacidad de almacenamiento de su ordenador. El diseño de un sistema de este tipo deberá, por consiguiente, economizar detalles.

El *Stock recording system* (sistema de control de existencias) de Dragon Data para el Dragon 64 posee siete *campos* en los que grabar información para cada registro de existencias. Se reservan para el número del artículo, su descripción, nuevos pedidos, el precio de costo, precio de venta y la unidad de medida.

Los campos forman una estructura importante en el control de almacén. La diferencia entre el precio de coste y el precio de venta nos dará el beneficio bruto. Gracias al número los artículos también se pueden agrupar, lo que ayuda a analizar y resumir los informes. El campo de unidad de medida es esencial porque las mercancías se embalan o envasan de diversas maneras.

Tal vez el aspecto más interesante del diseño de un sistema de control de almacén sea el hecho de que gran parte de la información requerida se compone de datos *dinámicos* en vez de *estáticos*. Ya hemos visto que los registros del libro mayor suelen contener información relativamente constante acerca del cliente o de la cuenta (que ocupa la parte de encabezamiento del registro) e información relativa a las transacciones efectuadas en dicha cuenta.

En el caso del control de existencias, sin embargo, las líneas divisorias que existen entre la parte estática del registro y la parte dinámica o basada en las transacciones del registro, se vuelven menos definidas. El nombre de la existencia y la descripción de grupo así como la codificación son el equivalente de la información estática de los registros de clientes o proveedores en los sistemas de libro de compras o ventas. Pero en un paquete para control de existencias, los datos estáticos se han de complementar con una gran cantidad de información derivada de las transacciones de existencias.

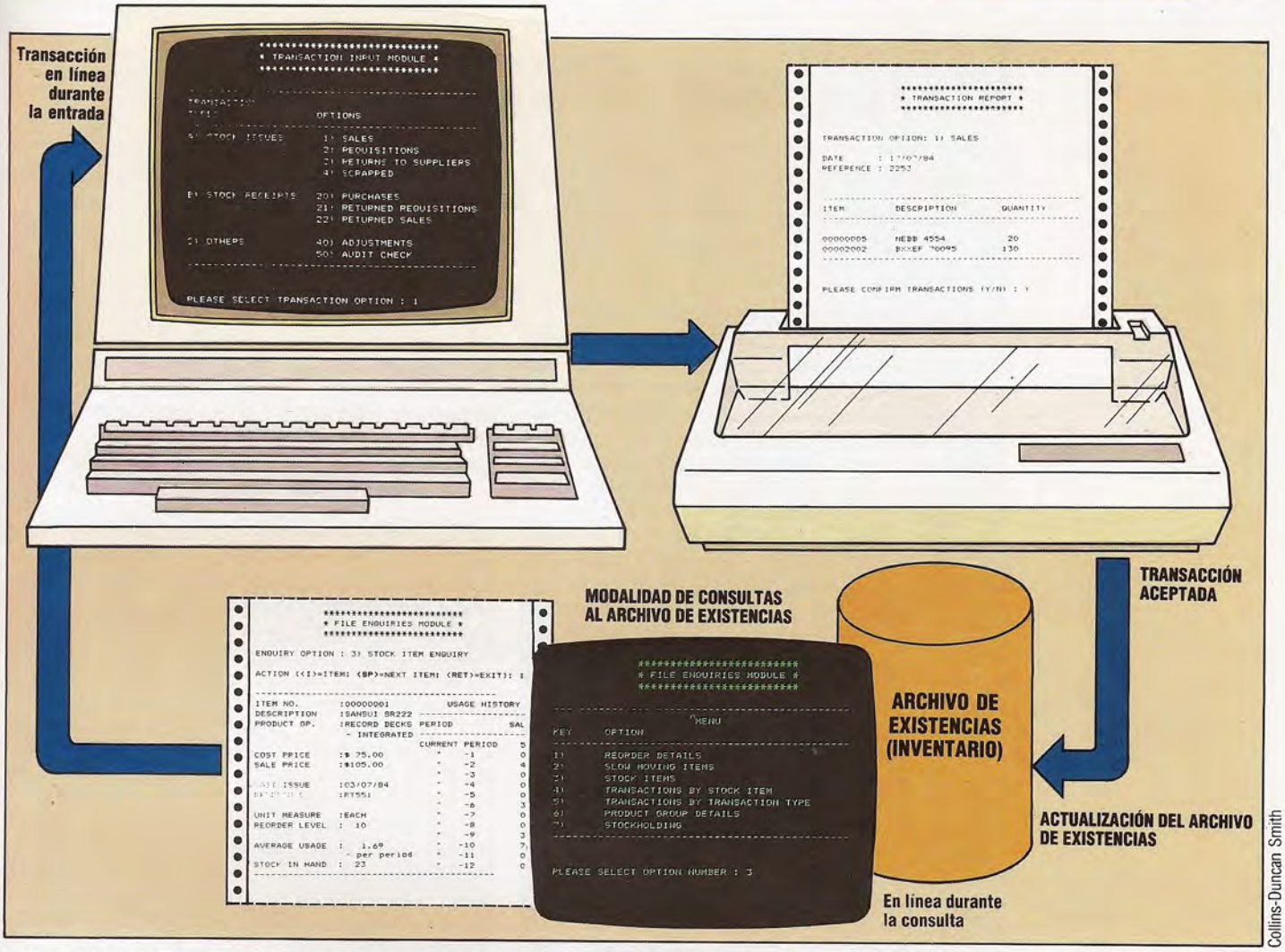
Por ejemplo, la dirección postal o el número de teléfono de los clientes en un archivo maestro del libro de ventas cambian con una frecuencia relativamente escasa. Por consiguiente, el programa tendrá una opción de mantenimiento del archivo maestro que permitirá que usted rectifique los registros de clientes particulares.

Pero si tomamos como ejemplo los campos de precio de venta y precio de costo del artículo del almacén, éstos pueden variar cada vez que la empresa vuelve a abastecerse de existencias. En este caso la solución lógica es que el programa tome esta información (junto con la fecha del cambio de precios y el volumen de existencias afectado por el mismo) desde una rutina de entrada de transacciones que registre los bienes recibidos, y no desde un programa para mantenimiento de archivos.

Para llegar a comprender cómo opera un programa para control de existencias es necesario que analicemos las rutinas de entrada y las facilidades para consultas e informes que las mismas ofrecen. En la ilustración gráfica hemos tomado como ejemplo el programa de Dragon señalando los diversos archivos. Los tres elementos importantes son: las rutinas de entrada de las operaciones de tráfico, detalles de las mismas y un archivo que informe sobre los artículos en existencia.

Un almacén de respuestas
El *Stock recording system* es uno de los muchos programas de aplicaciones para el Dragon 64 escritos para operar bajo el OS9. Se trata de un sistema operativo multiprogramación y multitarea que desarrolló Dragon a partir del sistema operativo UNIX

Ian McKinnell



Collins-Duncan Smith

En el sistema Dragon todas las visualizaciones de entrada de una operación tienen el mismo trazado. Los tipos de operaciones mercantiles se explican más o menos por sí solos. Por el momento nos ceñiremos a las ventas (es decir, a los movimientos hacia afuera del registro de existencias). Pero antes es interesante observar que todos los programas de gestión se diseñan para que resulten cómodos y fáciles de aplicar, y que le indicarán al usuario que dé entrada a toda la información necesaria. Esto nos lleva a dos exigencias de programación diferentes que se han de satisfacer para el logro perfecto de un sistema.

De una parte, el programa debe reconocer ciertos campos de datos y realizar sobre los datos ciertas manipulaciones, aritméticas, de clasificación, etc. Por otra parte, el programa ha de guiar al usuario indicándole los datos para los que no ha dado una entrada apropiada.

Una vez que el usuario ha dado entrada a los datos, al número de referencia para documentar la entrada (número que corresponderá al de la factura de la venta) y al número del artículo, el ordenador lee el archivo de existencias para ver si el artículo en cuestión existe. De hallar el número, visualiza automáticamente la descripción asignada a ese número de artículo. Tal visualización actúa como un verificador óptico por el cual el usuario entiende que puede dar entrada a la cantidad vendida.

A partir de esta información el ordenador está en condiciones de extraer una gran cantidad de detalles e informes. Por ejemplo, una de las opciones del menú principal, FILE ENQUIRIES (consultas al archivo), posee un submenú que consta de siete opciones que tratan los artículos en existencia, las transacciones y los artículos por grupos. He aquí las opciones: STOCK DETAILS (detalles de las existencias), SLOW MOVING ITEMS (artículos de movimiento lento), RE-ORDER DETAILS (detalles de nuevos pedidos), TRANSACTIONS (BY STOCK ITEMS) (transacciones por artículos de almacén), TRANSACTIONS (BY TRANSACTION TYPE) (operaciones según el tipo de transacción), PRODUCT GROUP DETAILS (detalles por grupos de productos) y STOCKHOLDING (conservación de existencias).

Las transacciones de venta afectan a todos estos informes. Si se solicita, por ejemplo, un informe sobre nuevos pedidos, el programa verificará si las existencias vendidas han hecho que la cantidad de las disponibles caiga por debajo del nivel especificado para volver a efectuar un pedido.

Todos los detalles introducidos en la visualización de una venta están correlacionados y se utilizan de una forma u otra. La visualización de consulta del programa Dragon proporciona una clara ilustración acerca de cuánta información de tipo administrativo se puede obtener a partir de una anotación de las ventas. En el tratamiento de históricos se ofrecen, inmediatamente calculados, el flujo y el volumen de la mercancía movida durante el año, el promedio de uso por período, y también se visualizan la fecha y referencia de la última operación.

Entrada documentada
El programa de control de existencias recibe información sobre una operación y comprueba la autoridad del usuario para dar entrada a información, así como que las transacciones estén correctamente anotadas. El archivo de existencias, que contiene los registros de artículos de existencias individuales, está en línea durante este proceso, y se actualiza. El programa también contiene un módulo de base de datos, que permite al usuario inspeccionar el archivo de control de existencias y crear informes relativos a diversos aspectos del inventario

Hora de examen

En esta ocasión nos detendremos para realizar un breve recuento del trabajo que hemos efectuado hasta ahora

El curso lo empezamos sugiriendo que para el trabajo resulta esencial disponer de las herramientas adecuadas (véase p. 524). Hay una gran variedad de equipos que resultan idóneos para las tareas de construcción, modificación o reparación. Usted *puede* utilizar un destornillador plano para un tornillo cruciforme, pero es bastante probable que si lo hace los estropee a ambos.

Cuando se trata de separar y unir cables ocurre lo mismo. Una buena juntura de soldadura durará más y funcionará mejor que una que se ha conectado torpemente. Aplique siempre la soldadura al cable y no al soldador, porque la soldadura irá corroyendo lentamente la punta de su soldador. Cuanto menos contacto haya entre ambos, mejor será. Deje que la soldadura fluya a lo largo de los dos cables que desea unir y sólo cuando la juntura esté cubierta totalmente quite el calor y enfríe la juntura soplando suavemente sobre ella. Siguiendo estas reglas se asegurará de que la juntura no esté seca. Desoldando usted puede quitar componentes de los tableros de circuitos con toda seguridad con el fin de repararlos o sustituirlos (véase p. 548).

Hemos analizado los conceptos fundamentales de la electrónica digital, tanto en el curso de *Bricolaje* como en el de *Ciencia informática*. Hemos presentado los componentes básicos y hemos visto cómo se interrelacionan. La resistencia es el más sencillo de todos los componentes, y el que se usa con más frecuencia. En el interior de su ordenador, verá más resistencias que circuitos integrados.

El condensador también es muy común. En los ordenadores, éstos se utilizan para filtrar el ruido que se acopla a una señal. Cada vez que una señal se modifica, se amplifica o se emplea de alguna otra forma, se degrada. De modo que es esencial algún medio de restaurar la señal a su forma incorrupta. El condensador es la forma más simple de eliminar los elementos que pudieran desvirtuar una señal.

Sin embargo, el componente más importante que hemos estudiado es el transistor. Éste es el componente esencial de todos los dispositivos que se utilizan para cambiar y manipular las señales en un ordenador. El transistor se puede emplear para amplificar una señal o para conmutar señales, encendiéndolas y apagándolas. Lo más importante, desde el punto de vista del ordenador, es que el transistor puede encender y apagar una señal de acuerdo a otra u otras señales. Esto es lo que sucede en el interior de una puerta lógica.

Hemos construido las tres puertas lógicas más sencillas, NOT, OR y AND, a partir de un puñado de componentes (aunque las puertas individuales del interior de los circuitos integrados del ordenador suelen estar construidas con otros tipos de transistores más complejos).

Las puertas lógicas no son particularmente útiles por sí mismas, pero se pueden combinar entre sí para formar circuitos lógicos que efectúen operaciones con los datos. En el capítulo anterior construimos un sumador incompleto, un circuito lógico sencillo para sumar dos números binarios, con las puertas lógicas de dos circuitos integrados.

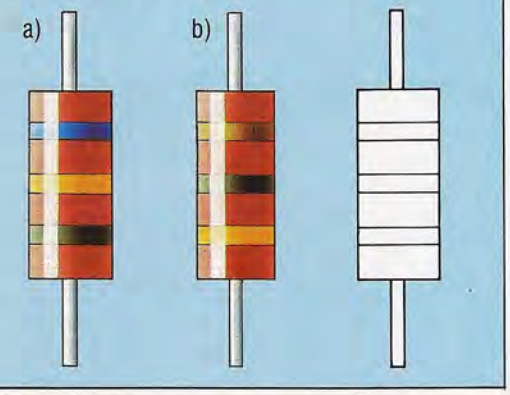
Los circuitos integrados son de los componentes más complejos que se emplean en electrónica. Los chips simples de lógica transistor-transistor (*Transistor-Transistor Logic*: TTL) que ha utilizado hasta ahora son paquetes de integración a pequeña escala (*Small Scale Integration*: SSI). En cada chip hay solamente unos pocos transistores. Éste fue el primer tipo de chips que se fabricó y los primeros ordenadores se basaban en ellos. Sin embargo, a medida que la técnica se fue perfeccionando, aumentó el número de transistores que se podían incluir en un único chip. La integración a media escala (*Medium Scale Integration*: MSI) permitió disponer de circuitos lógicos completos en un chip.

También hemos estudiado la integración a gran escala (*Large Scale Integration*: LSI) y la integración a muy gran escala (*Very Large Scale Integration*: VLSI). Una CPU completa en un único chip es un ejemplo de VLSI. Puesto que en el interior de un microprocesador hay miles de transistores, resulta difícil imaginar el circuito lógico de un chip de esta clase, pero esta complejidad hace que los chips individuales resulten muy eficaces y sencillos para el diseño de circuitos.

Una vez que dominemos estos fundamentos, podremos seguir adelante para abordar conceptos más complejos, lo que le brindará la posibilidad de construir algunas adiciones útiles para su micro. Primero, no obstante, ponga a prueba su destreza con los proyectos de la página contigua.

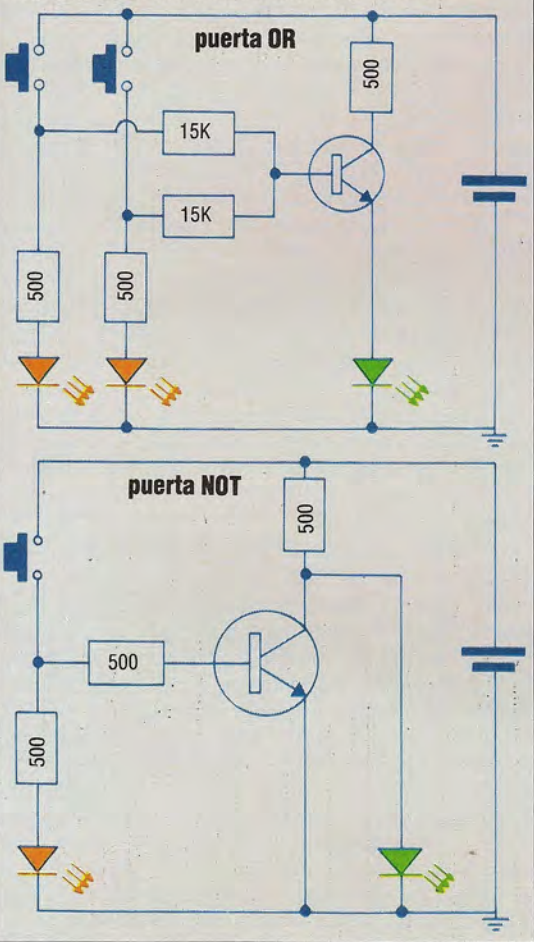
1) Resistencias

Las resistencias poseen una serie de franjas de colores para saber su valor. ¿Qué valores tienen estas resistencias? ¿Qué franjas tendría una resistencia de 150 ohmios?



2) Puerta NOR

En la página 625 construimos puertas NOT, OR y AND empleando transistores. Este primer ejercicio práctico consiste en construir una puerta NOR utilizando un circuito similar a aquellos para NOT, OR y AND. Como ayuda, abajo le proporcionamos los circuitos para OR y NOT. Este problema se puede enfocar de dos maneras. Podría usar sus conocimientos sobre circuitos lógicos para construir una puerta NOR combinando una OR y una NOT. O bien, hallar un procedimiento más corto si estudiara el circuito de la puerta NOT



3) Conversor de decimal a binario

Construya un circuito que convierta de decimal a binario. Con el fin de que el circuito sea sencillo, restringiremos este ejercicio a números binarios de dos bits, es decir, los números decimales del cero al tres. Su circuito habrá de tener cuatro interruptores de entrada etiquetados cero, uno, dos y tres. Cuando pulse uno de estos interruptores, en un par de LED se deberán reflejar los dos bits binarios correspondientes. Una tabla de verdad parcial para el conversor sería así:

BOTÓN CERO	BOTÓN UNO	BOTÓN DOS	BOTÓN TRES	BIT HI	BIT LO
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

4) BCD, ¿sí o no?

En el curso sobre *Ciencia informática* hemos estudiado el sistema numérico decimal codificado en binario (BCD). Éste es un sistema de numeración a medio camino entre el decimal y el binario; cada dígito decimal se convierte a su equivalente binario y los grupos de cuatro bits resultantes se agrupan entre sí para formar el número BCD. Por ejemplo, 53 se codifica como 01010011, 5 se representa mediante 0101 y tres se representa mediante 0011. Esto significa que todo dígito BCD válido es un grupo de bits comprendido entre 0000 y 1001, correspondiente a entre 0 y 9 en decimal. En BCD son no válidos todos los códigos que no estén comprendidos dentro de la escala de 1010 a 1111.

Construya un circuito para comprobar si una entrada dada de cuatro bits es un dígito legal. Su circuito ha de tener cuatro interruptores: B0, B1, B2 y B3, que representan al número que se está probando. Deberá haber dos salidas: un LED verde si el número es un dígito BCD legal y, en caso contrario, un LED rojo. La tabla de verdad sería así:

Equivalente decimal	B3	B2	B1	B0	BCD válido	BCD no válido
0	0	0	0	0	1	0
1	0	0	0	1	1	0
2	0	0	1	0	1	0
3	0	0	1	1	1	0
4	0	1	0	0	1	0
5	0	1	0	1	1	0
6	0	1	1	0	1	0
7	0	1	1	1	1	0
8	1	0	0	0	1	0
9	1	0	0	1	1	0
10	1	0	1	0	0	1
11	1	0	1	1	0	1
12	1	1	0	0	0	1
13	1	1	0	1	0	1
14	1	1	1	0	0	1
15	1	1	1	1	0	1

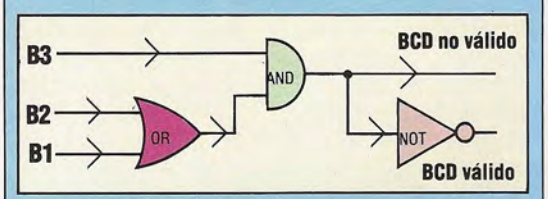
A partir de esta tabla de verdad podemos ver que la señal BCD viene dada por $\overline{B3} + \overline{B2}.\overline{B1}$. La señal BCD no válida es el NOT de ésta:

$$\overline{\overline{B3} + \overline{B2}.\overline{B1}}$$

que se puede simplificar así:

$$\overline{\overline{B3}} \cdot \overline{\overline{B2}.\overline{B1}}$$
$$B3.(B2+B1)$$

De modo que el circuito para probar un dígito BCD no válido es bastante sencillo y sólo implica a tres de las entradas. Un diagrama lógico sugerido para el circuito convalidador BCD es:



Las respuestas, en el próximo capítulo de *Bricolaje*.

Recurso esencial

Ahora examinaremos un aspecto básico en la programación en lenguaje máquina: las modalidades de direccionamiento

Toda instrucción en lenguaje Assembly, explícita o implícitamente tiene que ver con los contenidos de la memoria, y puesto que un byte sólo se puede distinguir de otro por su dirección, toda instrucción en lenguaje Assembly debe relacionarse al menos con una dirección. El modo como se alude a una dirección puede ser directo y evidente, como en LDA $E349, que significa "cargar en el acumulador el contenido de la dirección $E349". En este caso, tanto el acumulador (un byte con nombre más que el número de su dirección) como la dirección $E349 se mencionan sin ningún tipo de ambigüedad y la naturaleza de la relación entre ambos está clara.

Pero la referencia a una dirección puede ser mucho menos obvia: RET, por ejemplo, que significa "retornar desde una subrutina". Ésta no parece referirse a ninguna dirección en concreto, mientras no entendamos que "la dirección de la siguiente instrucción a ejecutar se localiza allá donde se efectuó la llamada a la subrutina". Aquí, la dirección cuyo contenido se ha de cambiar (o sea, el contador del programa: el registro que retiene la dirección de la próxima instrucción a ejecutar) no se menciona, ni tampoco la dirección en donde se pueden encontrar sus nuevos contenidos (es decir, la nueva dirección de posición). Estas dos instrucciones se pueden considerar como ejemplos de *modalidades de direccionamiento* que contrastan entre sí.

Hasta el momento, en este curso hemos visto en las instrucciones dos tipos de modalidades de direccionamiento: *modalidad inmediata*, como en LDA A,$45 o ADC #$31, y *modalidad directa absoluta*, como en STA $58A7 o LD ($696C),A. Éstas podrían parecer las modalidades de direccionamiento "naturales", cubriendo todos los casos posibles, salvo las modalidades implícitas como RTS o RET, pero hay otras y vamos a analizarlas por separado.

Direccionamiento de página cero

El *direccionamiento de página cero* (también llamado *direccionamiento corto*) se utiliza siempre que una instrucción pida una dirección comprendida entre $0000 y $00FF. Todas las direcciones de este tramo poseen un byte *hi* (alto) de $00 y, por consiguiente, residen en la página cero de la memoria. Tales instrucciones sólo necesitan dos bytes: un byte para el opcode y otro para el byte *lo* (bajo) de la dirección. Cuando la CPU detecta una dirección de un solo byte en un punto donde espera que haya dos bytes, da por sentado un byte *hi* de $00 y, por tanto, entiende automáticamente la página cero. La ventaja de esta modalidad de página cero es la velocidad de ejecución: a los datos de la página cero se accede con una rapidez apreciablemente mayor que a los datos de cualquier otra página, precisamente porque requiere una dirección de un único byte.

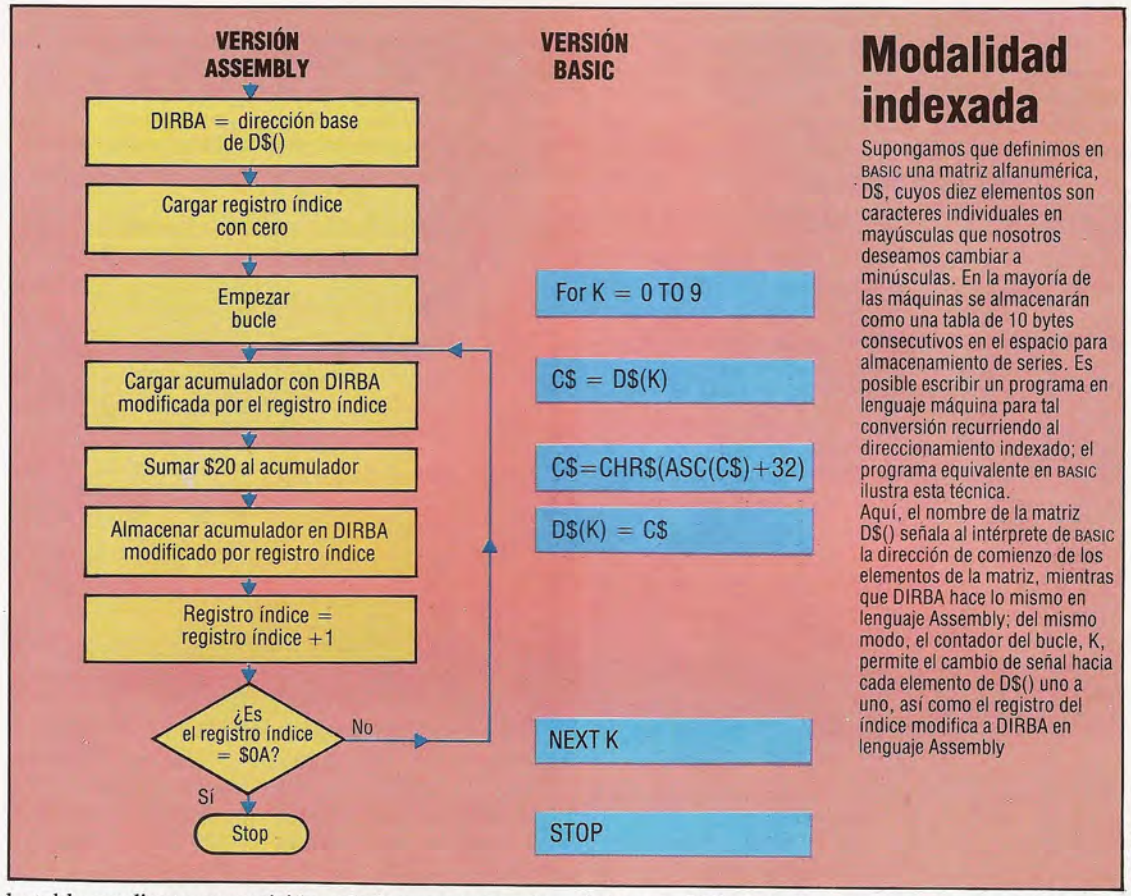
Los microprocesadores Z80 y 6502 se diferencian sobre todo por la forma como usan esta modalidad de página cero. En el Assembly del 6502, toda instrucción con dirección RAM (como LDA) se puede utilizar en la modalidad de página cero, y todos los 256 bytes de la página cero están a disposición de la CPU. En el Assembly del Z80, sólo una instrucción (RST, la instrucción *reset*, "recomenzar" o "restablecer") configura la modalidad de página cero, y sólo puede direccionar ocho posiciones específicas de esta página. Dado que la instrucción RST es tan específica en su direccionamiento, requiere solamente un opcode (la dirección forma parte del opcode propiamente dicho), lo que le confiere una gran rapidez de ejecución. Avanzado el curso de lenguaje máquina, hablaremos más detenidamente de la instrucción RST del Z80, por los usos especiales a los que se destina.

Podría parecer ridículo que estemos comparando la velocidad de las instrucciones en lenguaje Assembly cuando sabemos que la instrucción más lenta se mide en microsegundos; pero no es difícil llegar a escribir programas en lenguaje Assembly en los cuales ahorrar un microsegundo por cada instrucción ejecutada podría decantar hacia el éxito o el fracaso. Los programas para juegos que configuran gráficos en color tridimensionales de alta resolución, por ejemplo, exigen millones de instrucciones por cada operación de pantalla, y deben ejecutar estas órdenes con la mayor rapidez posible para producir una animación uniforme. Descontar un microsegundo de una operación sí que importa cuando uno coloca esa operación en un bucle y la ejecuta 64 000 veces de forma consecutiva.

Direccionamiento indexado

La *modalidad indexada* es fundamental para la programación en lenguaje Assembly, porque permite la construcción de estructuras de datos encadenados. Sin este tipo de estructuras los programas se limitan a dirigirse a posiciones de memoria individualmente por dirección: esto es lo que hemos venido haciendo hasta ahora en todos los programas explicados. Al disponer de la indexación, sin embargo, se pueden manipular más datos y se conceden mayores poderes al programador.

Los elementos esenciales de la modalidad indexada son una *dirección base* y un *índice*. Supongamos que queremos disponer de una tabla de datos (p. ej., los códigos de caracteres ASCII) en bytes consecutivos. La "direción base" de la tabla es la dirección de su primer byte. A partir de éste podemos aludir a cada uno de los bytes subsiguientes de



Modalidad indexada

Supongamos que definimos en BASIC una matriz alfanumérica, D$, cuyos diez elementos son caracteres individuales en mayúsculas que nosotros deseamos cambiar a minúsculas. En la mayoría de las máquinas se almacenarán como una tabla de 10 bytes consecutivos en el espacio para almacenamiento de series. Es posible escribir un programa en lenguaje máquina para tal conversión recurriendo al direccionamiento indexado; el programa equivalente en BASIC ilustra esta técnica.
Aquí, el nombre de la matriz D$() señala al intérprete de BASIC la dirección de comienzo de los elementos de la matriz, mientras que DIRBA hace lo mismo en lenguaje Assembly; del mismo modo, el contador del bucle, K, permite el cambio de señal hacia cada elemento de D$() uno a uno, así como el registro del índice modifica a DIRBA en lenguaje Assembly

Liz Dixon

la tabla mediante su posición relativa respecto a la dirección base, de modo tal que el primer byte está en la posición cero, el segundo byte está en la posición uno, el tercero en la posición dos, y así sucesivamente. La posición relativa de un byte respecto a la dirección base de la tabla es su "índice", y la dirección absoluta de cualquier byte de la tabla se calcula a partir de la suma de la dirección base y el índice del byte. Si pudiéramos construir un bucle de programas en lenguaje Assembly y utilizáramos el contador del bucle como un índice para la dirección base de la tabla, entonces obtendríamos la dirección de cada byte de la tabla en secuencia, de manera semejante a como accederíamos a los elementos de una matriz en BASIC mediante un bucle FOR ...NEXT.

Nuevamente, los lenguajes Assembly Z80 y 6502 tratan el direccionamiento indexado de forma diferente. El chip 6502 emplea dos registros de un solo byte, llamados X e Y, cada uno de los cuales puede contener un índice que modifique a una dirección base. Esto limita la longitud de una tabla a 256 bytes (el número máximo posible en un solo byte). El chip Z80 contiene dos registros de dos bytes, IX e IY, que pueden retener la propia dirección base y que pueden luego aumentar o disminuir para señalar a sucesivos bytes de la tabla. Puesto que son registros de dos bytes, IX e IY pueden direccionar tantos bytes como pueda direccionar la CPU.

Direccionamiento indirecto

El direccionamiento indirecto implica el uso de direcciones de señaladores, un concepto que ya habíamos introducido con anterioridad en el curso, en relación con las fronteras flotantes de la memoria (véase p. 538). Imaginemos que un grupo de personas forma un cine club y que están de acuerdo en ir una vez por semana a ver la película que recomienda el presidente del club. La película escogida puede que se esté exhibiendo en cualquiera de los doce cines distintos de la localidad, de modo que el presidente escribe semanalmente en una cartulina el título de la película, hora y lugar, y luego la pega en el escaparate convenido de una tienda del centro de la ciudad. Los socios del club no saben en qué sala se exhibirá la película semana tras semana, pero sí saben dónde se halla la tienda, y la tienda les "señala" el cine buscado. La dirección de la tienda es, indirectamente, la dirección de la sala de cine.

Ian McKinnell

Indicador indirecto
En nuestra vida cotidiana no son frecuentes los casos de direccionamiento indirecto. Sin embargo, en esta fotografía el panel de destino y número de andén de donde salen los trenes contiene la información que desean los viajeros, de modo que el rótulo colgante que les indica dónde hallar ese panel direcciona indirectamente los datos. En una instrucción Assembly, direccionamiento indirecto significa que en la dirección que indica el operando se encuentra la dirección del byte donde están almacenados los datos; la dirección del operando es un indicador

En el direccionamiento indirecto se pueden escribir instrucciones que contengan la dirección de un señalador y afecten al contenido de la posición que indique el señalador (no al contenido del señalador, por supuesto). Las ventajas que ofrece esta modalidad de direccionamiento son considerables, especialmente cuando se combina con la modalidad indexada. Supongamos, por ejemplo, que usted escribe una rutina en lenguaje Assembly para buscar una tabla de datos para un carácter determinado, y retorna con la posición del índice del carácter. Supongamos, además, que usted desea manejar varias tablas de este tipo situadas en distintos lugares de la memoria, y que desea emplear la misma rutina para buscar cualquiera de ellas. Si la rutina se escribiera de modo tal que encontrara la dirección base de la tabla de búsqueda indirectamente, a través de la posición del señalador escogido, entonces se podría utilizar cualquier tabla, siempre y cuando el contenido de la posición del señalador se ajuste convenientemente antes de llamar a la rutina.

En general, los programas requieren mezclas de estas modalidades y no ejemplos puros de modalidades únicas. La instrucción LDA del 6502, por ejemplo, se puede utilizar en las modalidades que ofrecemos en el cuadro.

Opcode	Operando	Modalidad
LDA	#$34	Modalidad inmediata
LDA	$A2	Directa página cero
LDA	$967F	Directa absoluta
LDA	$A2,X	Página cero, indexada por X
LDA	$967F,X	Absoluta, indexada con X
LDA	$967F,Y	Absoluta, indexada con Y
LDA	($A2,X)	Indirecta, preindexada con X
LDA	($A2),Y	Indirecta, postindexada con Y

En este ejemplo convenía utilizar una instrucción del 6502 porque ilustra con toda claridad las combinaciones de modalidades de direccionamiento. Observe que las dos versiones indirectas de la instrucción están en página cero, así como las modalidades indirectas e indexadas. Una tabla como ésta podría resultar confusa a primera vista, pero cuando se practican las diversas modalidades su significado queda claro enseguida. Hasta el momento usted ya conoce y emplea tanto LDA como ADC en dos modalidades (inmediata y absoluta) sin que haya ninguna confusión.

La tabla, además, responde a la pregunta que planteábamos en el capítulo anterior: ¿cómo distinguir la modalidad de direccionamiento de una instrucción cuando la mnemotécnica es la misma en todos los casos? Pues bien, se puede ver que el formato del operando es diferente en cada modalidad, y que la única ambigüedad posible es si una instrucción como LDA SYMB1 está en modalidad de página cero o absoluta. Un programa ensamblador le resolverá este problema, pero si es usted quien ensambla el programa "a mano", tendrá que verificar si SYMB1 se ha definido como una cantidad de un único byte o de dos bytes.

En general, una vez que se empieza a utilizar un ensamblador podemos olvidarnos de aquellas cosas como opcodes o número de bytes por instrucción, y concentrarnos en aprender las técnicas de programación del lenguaje Assembly. Es importante entender la mecánica del lenguaje máquina, pero el lenguaje Assembly utilizado con un buen ensamblador simbólico constituye una forma mucho mejor de programar, combinando el poder del lenguaje máquina con muchas de las facilidades de los lenguajes de alto nivel.

Respuestas al "Ejercicio" de p. 658

El programa monitor de la página 598 se escribió en módulos teniendo presente la idea de la ampliación, de modo que agregar una opción de menú es relativamente sencillo:

Versión para el Spectrum

1) Ajuste la inicialización editando o agregando las siguientes líneas:

```
1050 LET LT = 5:DIM C$(LT):DIM 0$(LT,24):DIM
     X$(16)
1150 LET C$ = "ADGQB":LET C1 = -48:LET
     C2 = 10-CODE(C$(1))
1280 LET 0$(5) = "  B-VISUALIZACIÓN EN
     BINARIO"
```

2) La rutina de entrada de la línea 2000 ya ha producido la dirección de comienzo, la ha normalizado como un número hexa de cuatro dígitos en A$, y la ha convertido en un número decimal en DN, y la subrutina para visualización en binario sería:

```
7000 REM**S/R VISUAL HEXA&BIN**
7020 FOR P = DN TO (DN + 15)
7040 LET NM = P:GOSUB 3100:PRINT H$,
7060 LET N = PEEK(P):LET NM = N
7080 GOSUB 3000: PRINT B$;"  ";
7100 GOSUB 7300: PRINT B$
7120 IF P = 65535 THEN LET P = DN + 15
7140 NEXT P
7200 RETURN
7300 REM **S/R BYTE BINARIO**
7310 LET B$ = ""
7320 FOR D = 8 TO 1 STEP-1
7330 LET N1 = INT(N/2)
7340 LET R = N-2*N1
7350 LET B$ = STR$(R) + B$
7360 LET N = N1
7370 NEXT D
7380 RETURN
```

Versión para el BBC y el Commodore

Copiar la versión del Spectrum, con las siguientes correcciones:

1) Cambiar la inicialización de LT y 0$() como en la versión para el Spectrum, y agregar C$(5) = "B" en la línea 1150.

2) La línea 600 transfiere el control a la rutina de órdenes, por ello en el Commodore 64 y el BBC Micro debe cambiarse por:

```
600 ON CM GOSUB 5000,5500,6000,6500,7000
```

3) En el BBC, cambiar la línea 7060 de arriba por:

```
7060 N = ?(P):NM = N
```

4) En el Commodore 64 cambiar la línea 7350 anterior por:

```
7350 B$ = MID$(STR$(R),2) + B$
```

En constante avance

Hace cincuenta años era una pequeña firma proveedora de artículos eléctricos; hoy en día Tandy Corporation es una empresa multinacional sólida y de prestigio

Tandy Corporation, a través de sus departamentos de ventas al por menor, Tandy y la cadena norteamericana Radio Shack, posee 392 centros de ordenadores y más de 5 500 puntos de venta al detalle a lo largo de 76 países. La empresa cuenta con 29 fábricas que proveen equipos para la venta bajo las marcas comerciales Tandy y Radio Shack.

Tandy no comenzó su andadura como detallista de artículos electrónicos. La fundaron Norton Hinckley y David Tandy en 1927 como la Hinckley-Tandy Leather Company, una proveedora de cuero para las tiendas de reparación de calzado en Beaumont (Texas). El primer paso que la llevaría a convertirse en un gigante de la electrónica se dio en 1963, cuando Charles, el hijo de David Tandy, decidió ampliar el negocio y compró participaciones en una empresa con sede en Boston que se estaba tambaleando, denominada Radio Shack. Esta firma había venido operando desde los años veinte como un proveedor a pequeña escala de componentes eléctricos para radioaficionados y otros entusiastas de la electrónica. Aunque los negocios de la empresa se realizaban mayormente por pedidos por correspondencia y tenía un total de nueve tiendas en la zona de Boston, estaba experimentando enormes pérdidas. Para 1967 Charles Tandy había conseguido transformar los 4 millones de dólares de pérdidas en beneficios que ascendían a 20 millones.

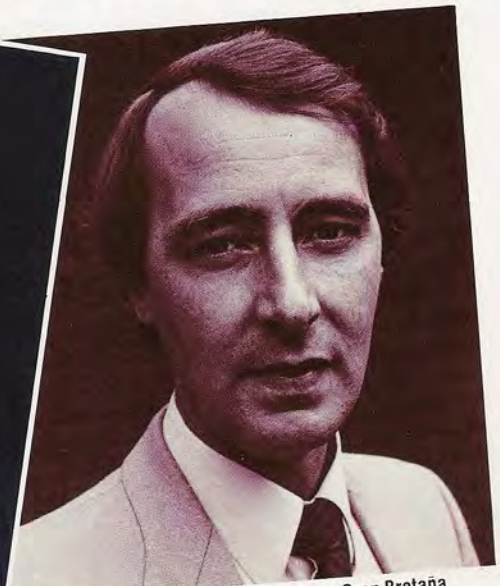
El siguiente paso importante se dio en 1970, cuando Tandy se hizo cargo de una cadena de grandes almacenes, Leonards, que le dieron acceso al mercado de artículos eléctricos para el consumidor. Ahora esta área se ha desarrollado hasta el punto de que en el catálogo Tandy de 1984 se incluyen 2 625 artículos no relacionados con los ordenadores, desde resistencias a equipos de alta fidelidad y sintetizadores, y 396 productos de carácter informático.

Tandy llegó a Gran Bretaña en 1973 y se estableció rápidamente en los principales centros comerciales como detallista de artículos eléctricos. En 1978, cuando se lanzó en dicho país el microordenador TRS-80 Modelo I, Tandy contaba con 120 tiendas. Para 1983 el número se había elevado a 227 comercios detallistas extendidos a lo largo y a lo ancho del país. Veintiséis de éstos eran tiendas de informática, especializadas exclusivamente en ordenadores, software y periféricos.

El Modelo I hizo de Tandy uno de los más importantes fabricantes de ordenadores. Se trata de una máquina de teclado individual con un microprocesador Z80, al menos cuatro Kbytes de RAM y una pantalla en blanco y negro con gráficos de baja resolución. Se encuentran a la venta unidades de disco y los usuarios hasta pueden ejecutar el sistema operativo de gestión CP/M en su máquina.

Desde entonces la línea de Tandy siempre se ha mantenido actualizada, aunque nunca ha vuelto a reconquistar el dominio de mercado que le proporcionó el Modelo I. La empresa se introdujo rápidamente en el mercado del micro de oficina a través del Modelo II, y en la actualidad su gama incluye los Modelo 12 y Modelo 16, ambas máquinas de 16 bits, así como el nuevo Modelo 2000, compatible con el IBM-PC. La informática del Modelo I escaló posiciones en el mercado con el Modelo III, que se puede considerar como un ordenador personal caro o bien como una económica máquina de oficina. Tandy ha sustituido el III por el Modelo 4 (y su versión portátil, el 4P). Éstos poseen mejores facilidades de gestión, pero mantienen la compatibilidad con los programas para el Modelo I y el III: un

John Sayers, director de marketing en Gran Bretaña



Vince Moore, director de ventas en Cran Bretaña

Ian McKinnell

Por todo lo alto
La tienda más céntrica de Tandy realza el punto más fuerte de la empresa: un solo proveedor local de equipos, servicios y asesoramiento. En Estados Unidos el nombre Radio Shack de la empresa es igualmente conocido

logro poco habitual en la microinformática. El resultado de ello es que los modelos TRS-80 disponen de una amplia gama de software de apoyo.

Tandy intentó volver a establecerse en el mercado personal con el Tandy Color Computer, un micro basado en el 6809 que comparte muchas de las configuraciones del Dragon. Aunque en Gran Bretaña no ha alcanzado nunca un éxito notable, el "CoCo" se ha vendido bien en Estados Unidos. Pero Tandy no ha renunciado a esta área del mercado, como indica la existencia de una miniversión mejorada: el Micro Color Computer.

Quizá la línea más interesante haya sido la de los ordenadores portátiles. Tandy se dedicó intensamente a ella, vendiendo una serie de ordenadores de bolsillo basados en la gama Sharp. Más recientemente, un acuerdo con una empresa japonesa, Kyocera, y la firma de software norteamericana Microsoft, dio como resultado el Modelo 100, un portátil a pilas de reducidas proporciones (tiene el tamaño de un libro), pero muy completo, pues incorpora BASIC, procesador de textos, software para comunicaciones y un diario.

En 1984, varias de las grandes firmas de la industria del microordenador estaban empezando a experimentar cuantiosas pérdidas, pero los negocios de Tandy parecen saludables, con un flujo continuo de nuevos productos provenientes de sus centros de diseño de Fort Worth (Texas), donde la empresa tiene ahora su sede, y de su subsidiaria TC Electronics Corporation, instalada en Tokio. Con las ventajas que ofrecen un gran departamento de fabricación y su amplia red de comercialización al por menor, Tandy Corporation parece estar en condiciones inmejorables para sacar partido de todos los desarrollos, cualesquiera que sean, que pudieran producirse en el campo de la microelectrónica.

La familia Tandy

PERSONAL

Color Computer
Después del éxito del Modelo I, Tandy lanzó un económico ordenador en color

Micro Color Computer
El MC10 es un ordenador en color para principiantes

Color Computer 2
Al Color Computer se le ha agregado un teclado completo

DE OFICINA

Modelo I
El Tandy TRS-80 original

Modelo II
El primer Tandy de oficina

Modelo III
Compatible con el Modelo I, el III es una máquina más al estilo de las de oficina

Modelo 4
Un Modelo III mejorado con una pantalla de 80 columnas

Modelo 12
Excelente ordenador de oficina para usuario único

Modelo 16
Micro multiusuario basado en el 68000 que funciona bajo el sistema operativo Xenix

Modelo 2000
Una máquina nueva, compatible con el IBM Personal Computer

PORTÁTILES

Modelo 4P
Versión portátil del Modelo 4

Ordenadores de bolsillo
Tandy ha vendido, bajo su nombre, cuatro de los ordenadores-calculadoras de bolsillo de Sharp

Modelo 100
Portátil muy popular, a pilas, del tamaño de un libro, con software incorporado

El poder de la luz

Cortesía de Teletape Video Ltd

Ian McKinnell

Combinación de vanguardia
Los aparatos reproductores de discos láser pronto serán tan asequibles como los ordenadores personales y será habitual encontrar a ambos conectados en el hogar. Comprando los discos y el software apropiados, el usuario podrá tener acceso a enormes bases de datos ilustradas y a sofisticados programas de entrenamiento, así como a entretenidos juegos de aventuras

Los discos láser son revolucionarios dispositivos a los que el usuario de microordenadores, por el descenso de los precios, ya puede tener acceso

A muchas personas les sorprenderá saber que los aparatos reproductores de discos láser, que hasta hace unos años se consideraban artículos de lujo por su elevado precio, ahora cuestan menos que los aparatos de video y ofrecen una reproducción visual muy superior. Una imagen de televisión es una imagen dinámica que se puede grabar en una cinta de video como una secuencia ininterrumpida. La única manera de localizar una determinada parte de la secuencia consiste en bobinar la cinta hacia adelante, ya sea a velocidad *play* (reproducción) o bien utilizando *fast forward* (avance rápido), con el contador de cinta como guía. Los discos almacenan las imágenes como fotogramas separados, lo que le permite al usuario acceder directamente, con precisión y rapidez, a cualquiera de ellos. La posición en disco de un fotograma se puede describir en términos de pista y sector, y un microprocesador puede llevar un catálogo actualizado de las posiciones.

Las tareas del microprocesador de activar una platina portadiscos a velocidades constantes y elevadas y posicionar una cabeza de reproducción exactamente sobre ciertas posiciones de la superficie, no son los principales logros de la tecnología del disco. El mayor desafío fue el de facilitar la inmensa densidad de almacenamiento de los discos. Posibilitar el almacenamiento de millares de fotogramas en un disco del tamaño de un disco fonográfico de larga duración de 12″ constituyó una labor especialmente audaz y ardua.

Si usted ha utilizado alguna vez gráficos de alta resolución en su micro, entonces sabrá que una imagen de televisión se compone de puntos individuales (pixels) de luz (cuanto mayor sea el número de puntos en la pantalla, mejor será la imagen) y que el almacenamiento de visualizaciones en pantalla de alta resolución exige emplear muchísima memoria. El BBC Micro, por ejemplo, tiene una reso-

Ian McKinnell

Una nueva memoria
Los discos láser de doce pulgadas y los discos compactos más pequeños se pueden utilizar para almacenar información tanto de video como de audio, así como los datos digitales almacenados en cassettes y discos flexibles de ordenador. El inconveniente es que en los discos no se puede escribir y que técnicamente son sólo ROM. Esto significa que el usuario no puede guardar sus propios programas e información en un disco láser.

lución máxima de 640 × 256 pixels por visualización en pantalla, que ocupa exactamente 20 Kbytes de memoria. Si un fotograma de televisión tuviera una resolución no más refinada que ésta, entonces almacenar un segundo de video (a 25 fotogramas por segundo) ¡requeriría 25 × 20 Kbytes de almacenamiento! Un minuto de programa de televisión ocuparía 30 Megabytes (30 millones de bytes) y un episodio de una serie de televisión necesitaría más de un Gigabyte (1 000 Megabytes) de memoria; y si decidiéramos emplear discos flexibles de densidad simple y de una sola cara, que utilizan la mayoría de los ordenadores personales, ¡exigiría más de 6 000 discos y una semana de trabajo!

A la luz de estas cifras, el almacenamiento en cinta de video se vuelve bastante más atractivo: trasladar media hora de programas de televisión a un disco parece plantear problemas insuperables.

La respuesta a este dilema radica en escribir los datos muy pequeños. Una grabadora de discos láser graba los datos con un haz láser delgado como un cabello sobre una placa metálica revestida con una envoltura translúcida resistente. Para leer los datos del disco se utiliza un haz de bajo poder. Para este disco se emplean láseres como aguja de lectura-escritura por ser dispositivos de fina resolución y baja tolerancia. Ninguna otra técnica podría leer y escribir tal cantidad de datos en un espacio tan pequeño.

El formato es una combinación de las técnicas utilizadas en alta fidelidad y en las unidades de disco. Los surcos de los discos fonográficos forman una espiral continua y las paredes de los surcos llevan una representación grabada de las formas de onda del sonido grabado. La mayoría de los usuarios de micros saben que las pistas de una unidad de disco son círculos concéntricos, y que en el disco la información se escribe magnéticamente y se almacena digitalmente como patrones de unos y ceros. El formato del disco láser utiliza pistas y no surcos, pero éstas forman una espiral. En el disco la información se graba ópticamente en patrones de unos y ceros, pero no se puede borrar. Los unos y los ceros (que representan los patrones de puntos de que consta la imagen de televisión) se escriben en la superficie del disco mediante el láser, que quema diminutos orificios en la película metálica para representar a los unos, dejándola intacta para representar los ceros. Cada agujero tiene una anchura de medio micrómetro (0,0005 mm) y una profundidad de un décimo de micrómetro (0,0001 mm). Así, un centímetro cuadrado de superficie de disco podría albergar 400 millones de estos orificios.

Esta increíble miniaturización apenas si es suficiente para hacer frente a las demandas de almacenamiento de video. Un disco de 35 cm de diámetro retiene 54 000 fotogramas de televisión por cada cara, o aproximadamente 36 minutos de tiempo de reproducción. Los cálculos del tamaño de los fotogramas que realizamos antes en este artículo se basaban en la resolución de gráficos en ordenador en blanco y negro, mientras que el disco láser ha de almacenar información en color para cada punto de la imagen de televisión y llevar, asimismo, un canal de audio. Un fotograma en color, con su correspondiente canal de sonido, podría, por consiguiente, requerir 100 Kbytes de espacio de almacenamiento. 54 000 de estos fotogramas utilizarían hasta 5 400 000 Kbytes, o 5,4 Gbytes.

Habiendo superado el problema de la limitación de almacenamiento, los discos láser ofrecen amplias posibilidades de aplicaciones. Uno de los beneficios fundamentales es el de eliminar uno de los mayores obstáculos del procesamiento de datos: la recogida y entrada de datos. La información por lo general está disponible fácilmente, pero todavía alguien debe sentarse frente a un terminal y digitar representaciones codificadas de los datos en el sistema: un procedimiento tedioso, caro y que ocupa mucho tiempo. Si, en cambio, uno puede dirigir una cámara a los datos y dejar que la misma almacene la información visual en disco mientras uno sólo da entrada a detalles de indexación de las imágenes grabadas, entonces el volumen de trabajo disminuye sustancialmente.

Los aparatos reproductores de discos láser varían mucho en cuanto a su sofisticación. Se encuentran a la venta aparatos económicos para uso personal. Éstos se pueden utilizar para visionar películas como un aparato de video normal, pero con la ventaja adicional de que proporcionan una notable calidad de imagen, tanto en la reproducción normal como en congelación de imagen y cámara lenta.

Las verdaderas posibilidades no empiezan hasta que uno puede hacer que los fotogramas individuales los seleccione un programa para ordenador. Los aparatos reproductores que pueden hacer esto son de marcas como Pioneer y Philips, pero por el momento son caras y están destinadas sólo a la utilización profesional. El sistema más sencillo consiste en usar una interface IEEE o RS232 de modo que

el ordenador pueda seleccionar un fotograma determinado por su número.

Philips ha llevado la idea un paso más adelante y ha incorporado en sus modelos más recientes un microordenador sencillo. Éste puede cargar un programa apropiado para un disco determinado desde un cartucho EPROM enchufado en la máquina, o bien desde el propio disco láser. Cada disco láser almacena dos canales de audio y una pista de video. Esto permite, por ejemplo, que un mismo disco contenga una pista de sonido en dos idiomas. Sin embargo, si no se necesita la segunda pista de audio, se la puede utilizar para almacenar un programa de ordenador.

De modo que contamos con un banco de 54 000 imágenes de calidad bajo el control del ordenador. La etapa final consiste en mezclar las imágenes del disco láser con el texto proveniente del ordenador. Esto se podría hacer con dos monitores separados, o mezclando dos entradas de video, o utilizando un monitor con su propio generador de teletexto. Esta fase final constituye un medio totalmente nuevo: el video interactivo. El usuario y el software pueden guiar la visualización en televisión, tanto de secuencias de acción como de fotogramas fijos, mediante la lectura del disco por cualquier orden.

La aplicación más obvia es para una base de datos ilustrada. El usuario podría formular preguntas al ordenador y éste recuperaría la información pertinente de una base de datos e instruiría al aparato reproductor para que visualizara un fotograma de video adecuado. Se podría utilizar como referencia en bibliotecas y en escuelas, para todo tipo de fines, desde identificar diversas flores hasta seleccionar artículos de un catálogo.

El video interactivo puede progresar aún hasta otra etapa al implicar al usuario en las secuencias mostradas en la pantalla. Un programa de entrenamiento podría explicar algo en un breve trozo de película, recapitulando de ser necesario o entrando en más detalles, y así sucesivamente. Se podría incluso producir películas en las que el usuario asumiera la tarea de dirección tomando sus propias decisiones en nombre de los personajes de la historia. El desarrollo de la película, como su final, serían distintos según cómo uno la "jugara". Los entusiastas de los juegos de aventuras para micros personales encontrarán apasionantes estas posibilidades.

En el desarrollo de este mercado existen, por supuesto, ciertos problemas. Aparte del costo de los equipos y de la fabricación de discos láser, se han de desarrollar nuevas capacidades en cuanto al diseño y la producción de discos interactivos, tanto desde el punto de vista del software para ordenadores como del guión y filmación de imágenes. Muchas empresas pequeñas están empezando a afrontar estos desafíos; es así como una multinacional denominada Computer Assisted Televideo (CAT) ya se ha establecido como una fuerza dominante en el negocio. La empresa le proporciona al cliente un servicio completo: elección e instalación de equipos, diseño y producción de discos y escritura del software correspondiente. CAT tiene planificado abordar casi todas las aplicaciones del video interactivo, pero los elevados costos actuales limitan la mayor parte de su trabajo a la producción de programas de entrenamiento para grandes empresas.

No obstante, las posibilidades para el usuario de un micro personal no se deben ignorar. A pesar de que el costo de producción de los discos es elevado, no lo es más que los costos relacionados con películas y grabadoras. Los discos láser ya están a la venta, de modo que los discos interactivos, con software, podrán adquirirse pronto, aunque, eso sí, a un precio bastante mayor. Considerando la calidad de los programas de entretenimiento y educativos que posibilita el video interactivo, el precio a pagar puede considerarse razonable.

El video interactivo
Muchos aparatos reproductores de discos láser poseen una interface IEEE o RS232 que les permite ser controlados por microordenadores. Una instalación típica podría ser un ordenador con una base de datos relacionada con imágenes del disco láser, almacenadas en discos flexibles. El usuario puede seleccionar ítems de interés de la base de datos, el ordenador instruye luego al aparato reproductor de discos láser para que busque y visualice la imagen correspondiente en virtud de un número de fotograma. El sistema aquí ilustrado adopta un enfoque popular al combinar la salida del ordenador y las imágenes del disco láser en una sola pantalla. Un sistema más sencillo utilizaría dos pantallas separadas

Steve Cross

Desfile ordenado

Los archivos secuenciales son herencia de un tiempo en que el procesamiento de datos se basaba en cinta. Veamos cómo se crean y se accede a ellos desde un programa en BASIC

El archivo binario (o de programas) es un caso simplificado de archivo secuencial. Cuando usted digita SAVE "nombreprog", el sistema operativo realiza las diversas operaciones necesarias en la creación de un archivo: primero, abre comunicación con la cinta o la unidad de disco y escribe una etiqueta de encabezamiento que contiene el nombre del archivo y algo de información acerca de su contenido. A continuación, el sistema operativo escribe en el archivo el bloque de memoria que contiene el programa corriente. Por último, escribe en el archivo un indicador de final de archivo y cierra la comunicación entre el ordenador y la cinta o unidad de disco.

Las órdenes de BASIC que se utilizan para crear un archivo secuencial varían de un ordenador a otro, pero todas ellas deben llevar a cabo las mismas tareas. Tomando como ejemplo al Commodore 64:

```
100 R$ = "Éste es un registro del archivo"
150 OPEN 5,1,2,"ARCHDATO,SEQ,WRITE"
200 PRINT #5,R$
250 CLOSE 5
```

Se empieza con la orden OPEN 5,1,2 para que el sistema operativo establezca un canal de comunicación, denominado canal 5, con el dispositivo número 1, la unidad de cinta. El sistema operativo puede crear archivos secuenciales en un número de dispositivos distintos, y puede acceder a varios archivos diferentes al mismo tiempo, de modo que el canal del ordenador al dispositivo y, por tanto, a un archivo determinado, debe estar etiquetado. (En cuanto al último número, 2, es específico del Commodore y no importa por ahora.)

Después de la orden OPEN viene el nombre del archivo. En el Commodore, éste consiste en un nombre como ARCHDATO" seguido por el tipo de archivo (SEQ, por archivo secuencial) y el método de acceso (WRITE, que indica que el archivo se abre para escribir). Los archivos secuenciales, en efecto, se pueden abrir (OPEN) ya sea para escribir, ya sea para leer, y no para ambas cosas al mismo tiempo. Esta orden activa la cabeza de lectura-escritura del dispositivo, y escribe un registro de "encabezamiento" compuesto por el nombre del archivo y alguna información del sistema para indicar el comienzo del archivo.

La orden PRINT #5 de la línea 200 envía datos al archivo. PRINT mantiene su significado habitual, pero el signo "#" indica que se han de enviar datos al canal especificado y no a la pantalla, que es el canal de salida en caso de que no haya indicación alguna. El contenido de R$, por consiguiente, se envía por el canal 5 al archivo secuencial denominado "ARCHDATO". Ahora el archivo contiene un registro: la serie "Éste es un registro de archivo". Por último, CLOSE 5 cierra el canal 5 para toda otra comunicación más, y escribe en el archivo un indicador de final de archivo.

La información de un archivo puede leerse en una segunda etapa de este modo:

```
500 OPEN 3,1,2,"ARCHDATO,SEQ,READ"
550 INPUT #3,A$
600 CLOSE 3
650 PRINT A$
```

Tanto este fragmento de programa como el anterior utilizan la orden OPEN, que aquí significa "hallar el comienzo del archivo denominado ARCHDATO y prepararse para leerlo", mientras que previamente significaba "crear un archivo denominado ARCHDATO y prepararse para escribir en él". Los números de canal son distintos, pero éstos simplemente son etiquetas (en ambos casos se podría haber escogido un número diferente sin que se afectara la operación); la dirección del dispositivo, no obstante, es la misma en ambos casos porque el archivo está almacenado en el dispositivo 1, la unidad de cinta. El nombre de archivo y el tipo de archivo son los mismos porque identifican el archivo a acceder, pero el método de acceso es diferente: READ en lugar de WRITE.

Este programa tiene INPUT #3,A$, que significa "entrada desde canal 3" en vez de desde el teclado, el canal de entrada implícito cuando no se especifica otro. El primer registro completo del archivo se lee y se envía desde el archivo a través del canal 3 a la variable A$, de la misma manera como se envió, a través del canal 5 desde la variable R$ el archivo mediante PRINT #5,R$.

De este modo se evidencian algunas de las limitaciones y la mayoría de las ventajas de los archivos secuenciales: pueden almacenar todo cuanto pueda retener una variable y no existen limitaciones en cuanto a las dimensiones del archivo o su estructura. Pero su contenido ha de leerse empezando por el principio del archivo, registro a registro; no hay ninguna forma de abrir el archivo por un registro determinado, ni saltar, releer, borrar, insertar ni añadir nada en él.

Bob Hall

Dentro de un orden

Los archivos secuenciales se desarrollaron teniendo presente el almacenamiento en cinta; pero el orden en el que se envían los registros al archivo debe ser respetado. La única forma de clasificar o editar un archivo secuencial es mediante la creación de una nueva versión del mismo en algún otro lugar de la cinta, así como una cassette de música se puede editar sólo regrabándola o cortando la cinta

Archivar y hallar

```
199 REM ****************** CMB C64 ******************
200 REM*                  ESCRIBIR ARCHIVOS                *
201 REM ****************** CBM C64 ******************
220 GOSUB 1500
240 FOR K = 65 TO 90
260 Z$ = CHR$(K)+X$:C$ = D$+"ESCRIBIENDO  " + CHR$(K)
280 GOSUB 2000
300 NEXT K
399 REM ************************************************
400 REM*                  LEER ARCHIVOS                *
401 REM ************************************************
420 FOR L = 0 TO 1 STEP 0:FOR M = 1 TO 1
440 PRINT D$;"ACCEDER A REGISTROS"
460 INPUT"BUSCAR SERIE (* = ABANDONAR  )";N$
480 L$ = LEFT$(N$,1):IF L$ = "*" THEN GOTO 5000
500 IF L$ < "A" OR L$ > "Z" THEN M = 0
520 NEXT M
540 Z$ = L$ + Y$
560 GOSUB 3000
580 PRINT TAB(5) "REGISTRO  ";N;"  (PULSAR CUALQUIER
    TECLA)"
600 GET GT$:IF GT$ = "" THEN 600
620 NEXT L
999 END
1499 REM ************************************************
1500 REM ************ S/R INICIALIZAR ************
1501 REM ************************************************
1520 D$ = CHR$(147):PRINT D$,CHR$(8);CHR$(142)
1540 X$ = ",S,W":Y$ = ",S,R"
1600 RETURN
1999 REM ************************************************
2000 REM ********* S/R ESCRIBIR UN ARCHIVO *********
2001 REM ************************************************
2020 PRINT C$:INPUT"CUANTOS REGISTROS";R
2040 OPEN 8,8,2,Z$
2060 IF R = 0 THEN PRINT #8,"*":CLOSE8:RETURN
2080 FOR I = 1 TO R
2100 PRINT C$:PRINT "REGISTRO #";I
2120 INPUT "TEXTO.....";R$
2140 PRINT #8,R$
2160 NEXT I
2180 PRINT #8,"*":CLOSE8
2499 RETURN
2999 REM ************************************************
3000 REM ************ S/R LEER UN ARCHIVO ************
3001 REM ************************************************
3020 PRINT D$;"BUSCANDO  ";L$;"  A  ";N$
3040 OPEN 8,8,2,Z$
3060 FOR I = 1 TO 100000
3080 INPUT #8,R$
3100 PRINT R$
3120 IF R$ = "*" THEN N = 0:I = 100000
3140 IF R$ = N$ THEN N = I:I = 100000
3160 NEXT I:CLOSE8:N$ = ""
3180 RETURN
5000 REM ************ CERRAR PROGRAMA ************
5020 PRINT CHR$(9);"FIN DEL PROGRAMA":STOP
```

220: Inicialización

260-280: Crea archivos de la "A" a la "Z"

420-540: Introduce registro de búsqueda, halla la letra inicial e identifica el archivo que la contiene

560: Busca ese archivo

580-600: Informa del resultado de la búsqueda

1520: Limpia la pantalla y establece la modalidad mayúsculas

2040: Abre (OPEN) un archivo para escribir (WRITE)

2060: Verifica un archivo vacío, escribe "*", cierra (CLOSE) el archivo

2120-2140: Introduce el texto del registro y lo escribe en el archivo

2180: Escribe "*" señal del último registro, y cierra (CLOSE) el archivo

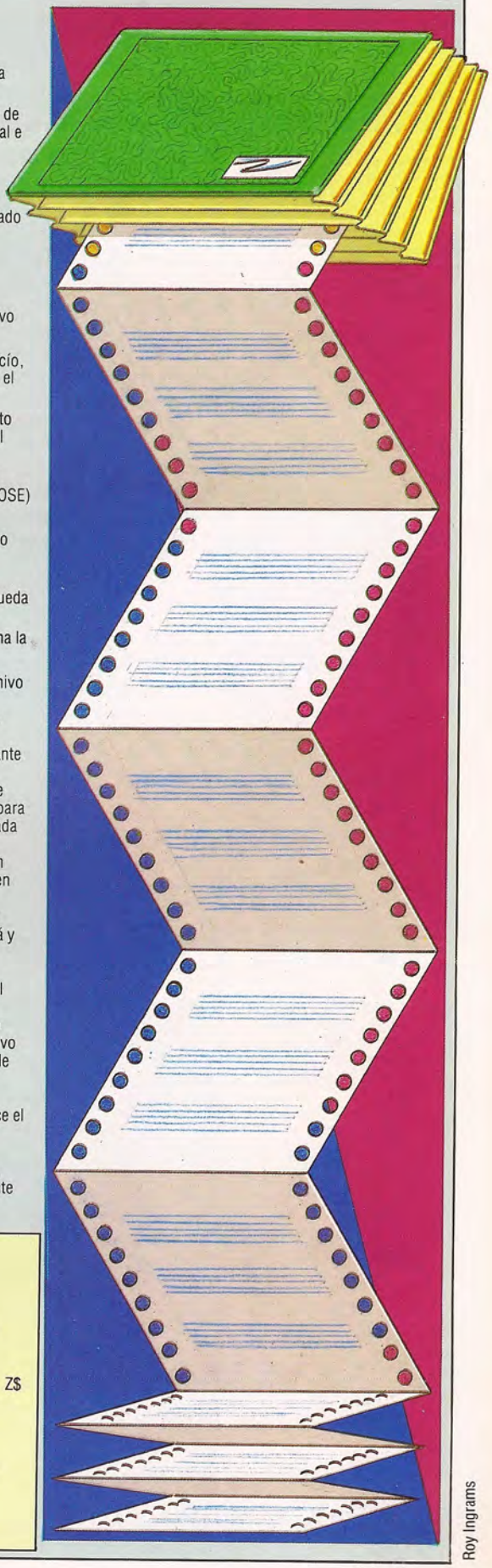
3040: Abre (OPEN) el archivo para leer (READ)

3120: Comprueba el último registro y abandona la búsqueda

3140: Comprueba si se ha hallado el registro y abandona la búsqueda

3160: Cierra (CLOSE) el archivo

Este programa demuestra la utilización de los archivos secuenciales en disco mediante la creación de un libro con índice alfabético sencillo que consta de 26 archivos, uno para cada letra del alfabeto. En cada archivo se pueden digitar registros que comiencen con esa letra, o ningún registro en absoluto. Luego se puede buscar cualquier registro. El archivo adecuado se buscará y se visualizará hasta que se encuentre el registro; de no encontrarse, se visualizará el mensaje "REGISTRO 0". El programa utiliza el DOS para ganar acceso directo al archivo correspondiente al registro de búsqueda; pero el archivo propiamente dicho se lee secuencialmente. Esto reduce el tiempo de búsqueda a una longitud razonable; si los archivos estuvieran en cinta, buscarlos sería excesivamente lento



Roy Ingrams

Complementos al BASIC

BBC Micro

Tenga en cuenta las variaciones para el Spectrum de las líneas 260 y 540. Sustituya PRINT #8, INPUT #8 y CLOSE8 por PRINT #C8, INPUT #8 y CLOSE #8

```
600 GT$ = GET$
1520 *DISK
1530 MODE 7
1540 D$ = CHR$(12):PRINT D$"UTILIZAR
     MAYUSCULAS"
1550 PRINT"—PULSAR CUALQUIER
     TECLA—":GT$ = GET$
2040 C8 = OPENOUT(Z$)
3040 C8 = OPENIN(Z$)
5020 PRINT"FIN DEL PROGRAMA"
```

Spectrum Microdrive

Ponga LET donde sea necesario.
Sustituya PRINT D$;.. por CLS PRINT..
Sustituya PRINT C$ por CLS:PRINT:C$
Sustituya OPEN 8,8,2Z$ por OPEN #8;"m";1;Z$
Sustituya CLOSE8 por CLOSE #8
Borre la línea 1540

```
260 Z$ = CHR$(K):LET C$ = "ESCRIBIENDO" + Z$
480 LET L$ = N$(1):IF L$ = "*" THEN GOTO
    5000
540 LET Z$ = L$
600 PAUSE 0
1520 CLS:LET F2 = PEEK 23658:POKE 23658,8
3080 INPUT #8;R$
5020 POKE 23658,F2:PRINT "FIN DEL
     PROGRAMA":STOP
```

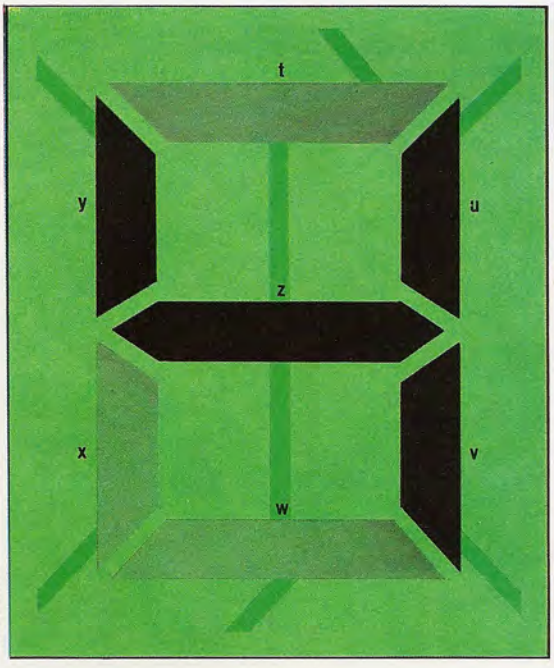
Éste es su número

Con siete luminosos segmentos pueden calculadoras, relojes y ordenadores portátiles representar cualquier número decimal

En nuestro diseño de circuito daremos por sentado que las representaciones binarias de los números decimales están en un código conocido como decimal codificado en binario, o BCD *(Binary Coded Decimal)*. Consiste en que cada dígito decimal tiene su equivalente en un grupo de cuatro dígitos binarios, tal como vemos en la tabla.

Una visualización en siete segmentos representa a los números decimales iluminando ciertos diodos emisores de luz, en el caso de una visualización LED (*Light-Emitting Diode*: diodo emisor de luz), o cambiando la polaridad de ciertas barras, como en las visualizaciones en cristal líquido (LCD: *Liquid Crystal Display*). La figura ilustra el nombre que hemos dado a cada uno de los siete segmentos (al segmento superior lo llamaremos *t*, al segundo inferior *w*, etc.). También indicamos qué segmentos se usan cada vez para formar los dígitos del 0 al 9.

Decimal codificado en binario	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010 a 1111	no se utilizan



La pantalla LCD comprende siete segmentos conmutables individualmente. Los designamos con letras de la *t* a la *z*

Kevin Jones

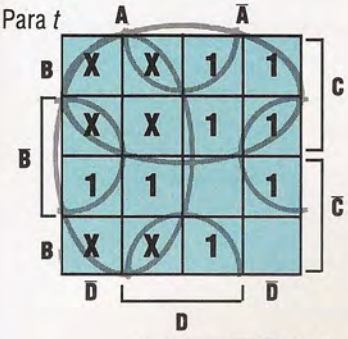
Ésta es toda la información que necesitamos para construir una tabla de verdad para nuestro conversor. A partir de la ilustración de la página contigua podemos decir qué segmentos se necesitan activar cuando el circuito recibe cada entrada individual. Por ejemplo, si la entrada binaria es 0100 (4 en decimal), el circuito activará los segmentos etiquetados *u, v, y* y *z*. Del mismo modo, la entrada 1000 (en notación decimal 8) debe hacer que se iluminen todos los segmentos. La tabla de verdad del circuito será:

Decimal	Entradas				Salidas						
	A	B	C	D	t	u	v	w	x	y	z
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	1	0	1	0	X	X	X	X	X	X	X
	1	0	1	1	X	X	X	X	X	X	X
	1	1	0	0	X	X	X	X	X	X	X
	1	1	0	1	X	X	X	X	X	X	X
	1	1	1	0	X	X	X	X	X	X	X
	1	1	1	1	X	X	X	X	X	X	X

Por desgracia, no existe ninguna forma sencilla de analizar esta tabla de verdad, y, por consiguiente, cada salida (*t, u, v, w, X, y* y *z*) se ha de simplificar por separado. Podemos construir un diagrama de Karnaugh para cada salida, colocando una X en cada diagrama para las zonas "indiferentes". Éstas podrían ayudar a la simplificación en algunos casos. Le hemos trazado aquí los siete diagramas k, uno para cada línea. Encerrando los grupos en óvalos podemos elaborar una expresión simplificada para cada línea de salida. A veces es posible incluso otra simplificación por medio del factor común.

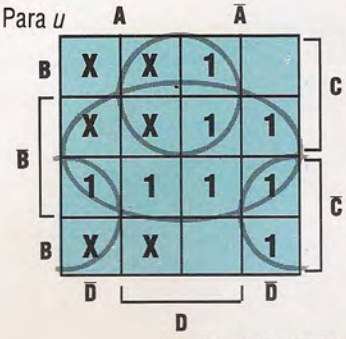
El primer diagrama k, por ejemplo, trata de las entradas que activarán el segmento *t* (que se usa en todos los números salvo dos). Simplificadas las expresiones para todas las líneas de salida, se dibuja el circuito final.

Para *t*



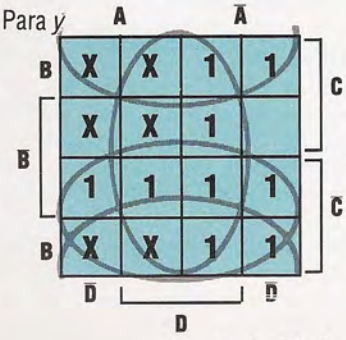
$t=C+A+\overline{B}.\overline{D}+B.D$

Para *u*



$u=\overline{B}+C.D+\overline{C}.\overline{D}$

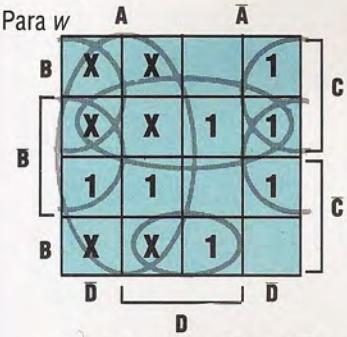
Para *y*



$v=D+\overline{C}+B$

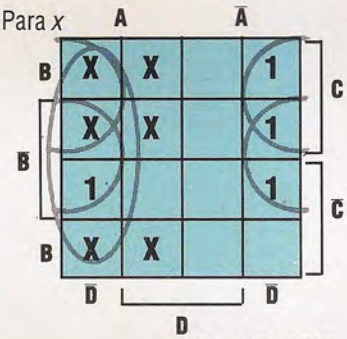
Codificación de los segmentos
A partir de estos diagramas podemos determinar qué segmento es necesario activar para formar cada dígito. El dígito simple, "1", sólo necesita que se enciendan los segmentos *u* y *v*. La tabla de verdad de la página contigua nos proporciona las salidas necesarias para cada uno de los dígitos

Para *w*



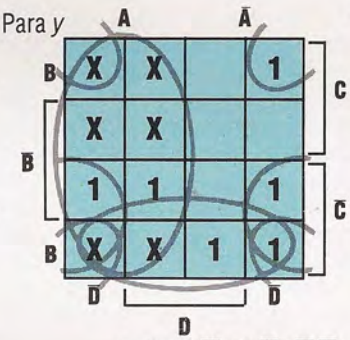
$w=A+C.\overline{D}+\overline{B}.C+\overline{B}.\overline{D}+B.\overline{C}.D$

Para *x*



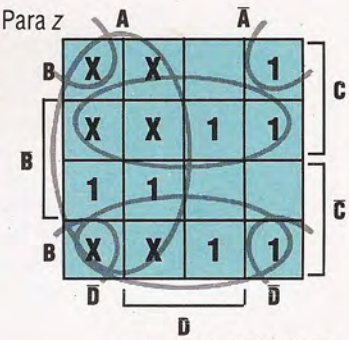
$x=C.\overline{D}+A.\overline{D}+\overline{B}.\overline{D}$
$=(A+\overline{B}+C).\overline{D}$

Para *y*

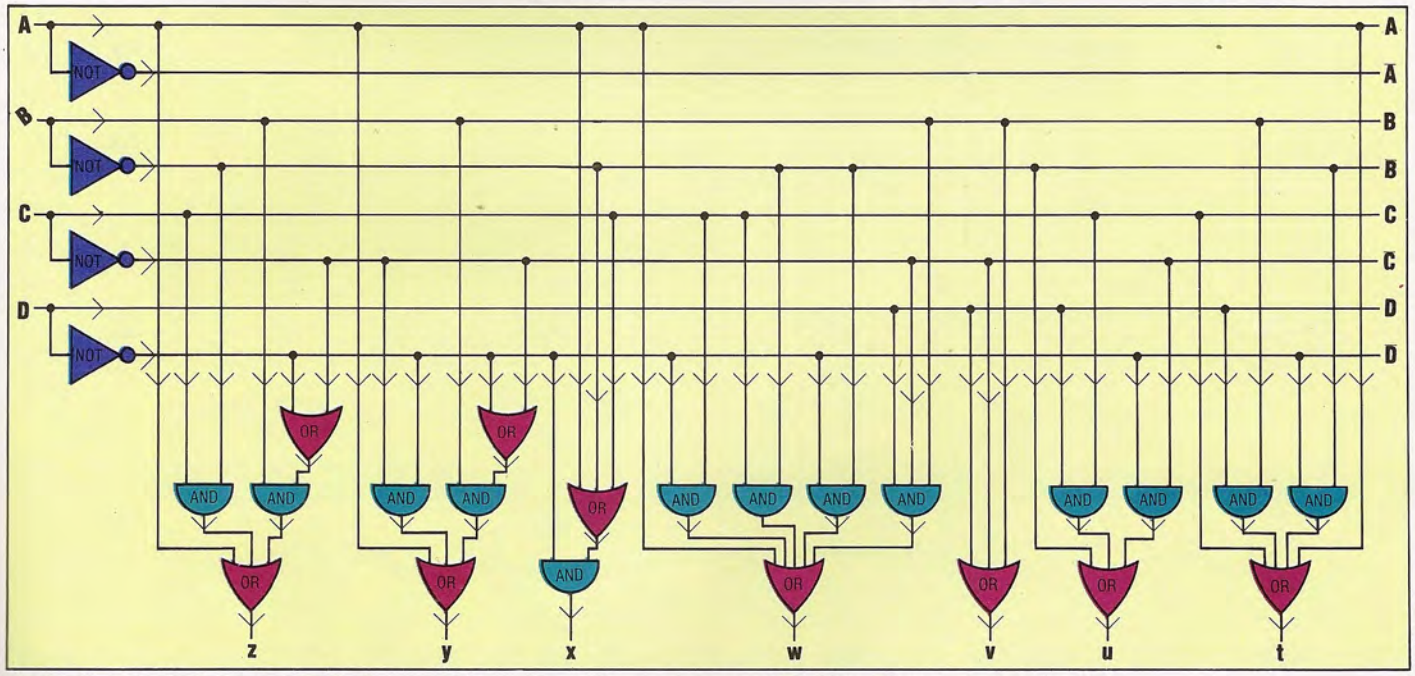


$y=A+B.\overline{C}+B.\overline{D}+\overline{C}.\overline{D}$
$=A+B.(\overline{C}+\overline{D})+\overline{C}.\overline{D}$

Para *z*



$z=A+\overline{B}.C+B.\overline{C}+B.\overline{D}$
$=A+\overline{B}.C+B.(\overline{C}+\overline{D})$



Liz Dixon

El circuito que hemos diseñado sirve tan sólo para una celda de la pantalla. Dado que la mayoría de los circuitos de este tipo poseen 8 o incluso 10 unidades, parecería que fuera necesario duplicar el circuito para cada celda. Sin embargo, se puede conseguir que todas las visualizaciones compartan un mismo conversor en virtud de un proceso que se conoce como *multiplexión*. Consiste en conmutar cada unidad de la visualización encendiéndola y apagándola en secuencia de modo que en cualquier momento dado sólo haya una unidad aceptando información proveniente del circuito del conversor. Debido a que esto sucede a una extremada velocidad, la visualización parece constante.

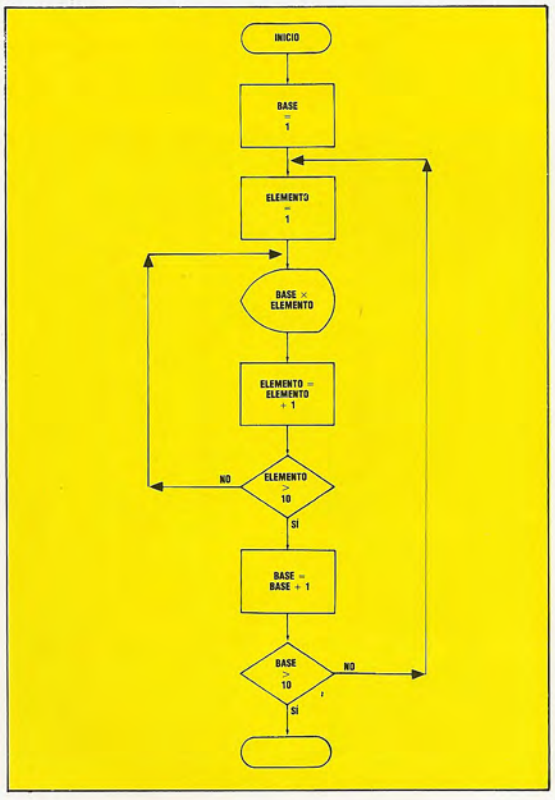
Contarlo todo

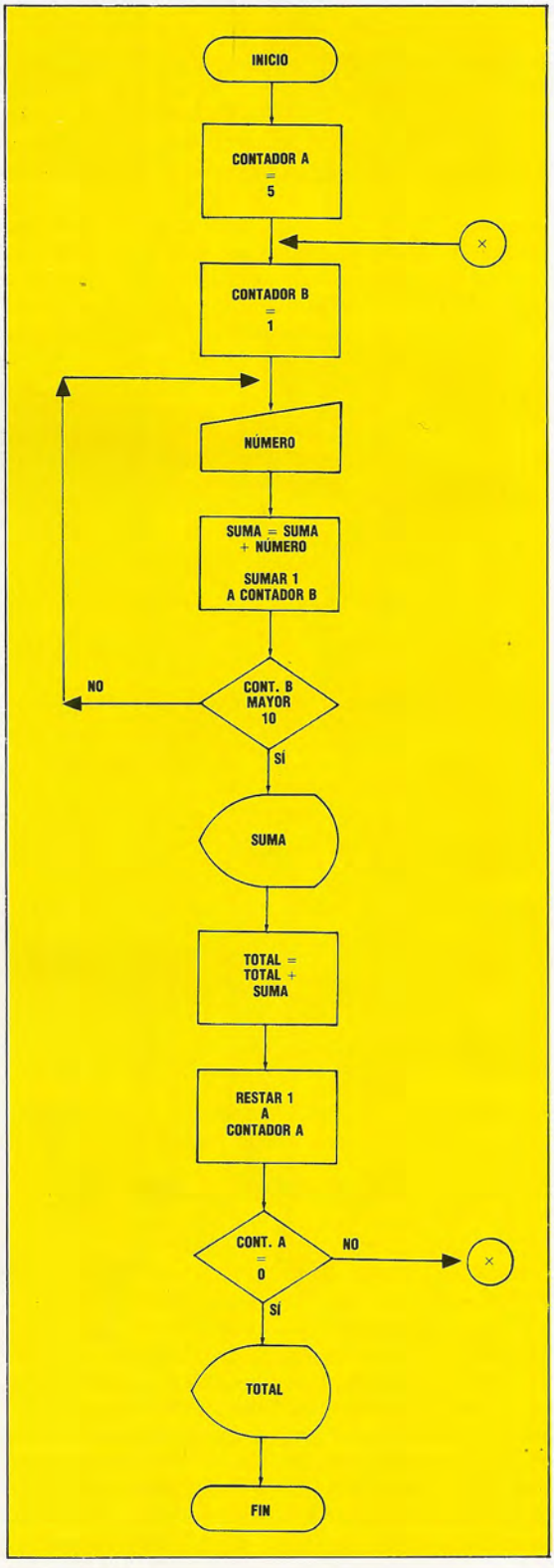
Completamos la lección iniciada en el capítulo anterior estudiando los contadores múltiples

Al igual que en el caso de los bucles, un mismo programa puede contener un número indeterminado de contadores. Así, tomando nuevamente el ejemplo del capítulo anterior, efectuaremos algunas variaciones.

Se dispone ahora de cinco grupos de diez cantidades cada uno y, además de visualizarse las sumas parciales (por cada grupo), se debe dar, asimismo, la suma total de ellos. Para conseguirlo, esta vez va a ser necesario utilizar dos contadores: el A, que será el de los lotes de números y que reiterará cinco veces el ciclo general, y el B, que se encargará de repetir diez veces las operaciones necesarias para conseguir los totales parciales.

Para completar este tema le ofrecemos otro ejemplo de programa con más de un contador. Es una versión del conocido programa de la tabla de multiplicar al que se ha incluido alguna variación. Se trata, como siempre, de visualizar las tablas de los diez primeros elementos, del uno al diez; pero esta vez con una solución más elegante. Al igual que en el anterior ejemplo, van a emplearse dos contadores: el del número base, cuya tabla se va a visualizar, y el de elementos, que, a diferencia del anterior, cada vez que rebase su valor máximo de 10, volverá a ponerse con su valor inicial de uno, para ser utilizado con la próxima base, es decir, el contador de tablas incrementado.





Prueba

He aquí las respuestas a los ejercicios planteados en el capítulo anterior

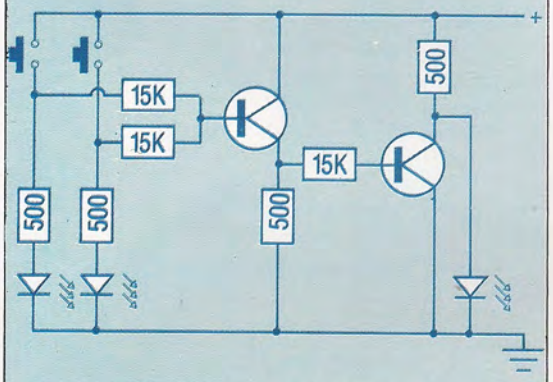
En el último capítulo de *Bricolaje* le ofrecimos algunos ejercicios sencillos para poner a prueba sus conocimientos de los temas que hemos tratado. Aquí le presentamos las respuestas. Puede que éstas no se correspondan exactamente con las suyas, ya que existen muchas maneras de diseñar un circuito para que realice una tarea determinada. No obstante, usted podrá probar si sus circuitos funcionan cotejándolos con las tablas de verdad dadas con el problema.

1) Resistencias

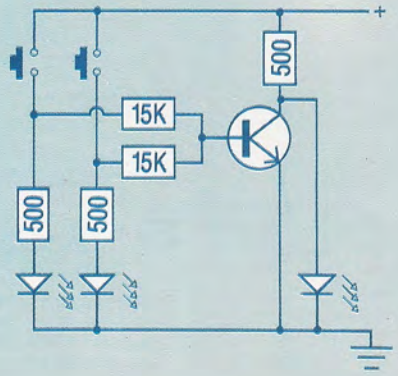
Los valores de las resistencias ilustradas son: a) 6 400 K-ohmios y b) 150 K-ohmios. Ésta tiene franjas marrón-verde-marrón (leyéndolas en dirección a una franja dorada o plateada).

2) Puerta NOR

El método más obvio para construir una puerta NOR consiste en combinar los dos circuitos para OR y NOT tal como vemos abajo:

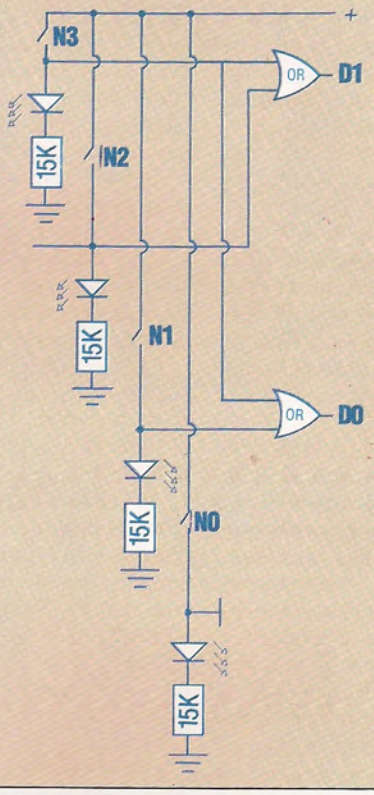


Sin embargo, el procedimiento más corto consiste en utilizar el circuito para una puerta OR y tomar la señal de salida desde el colector del transistor en vez de su emisor, de modo similar al circuito de la puerta NOT:



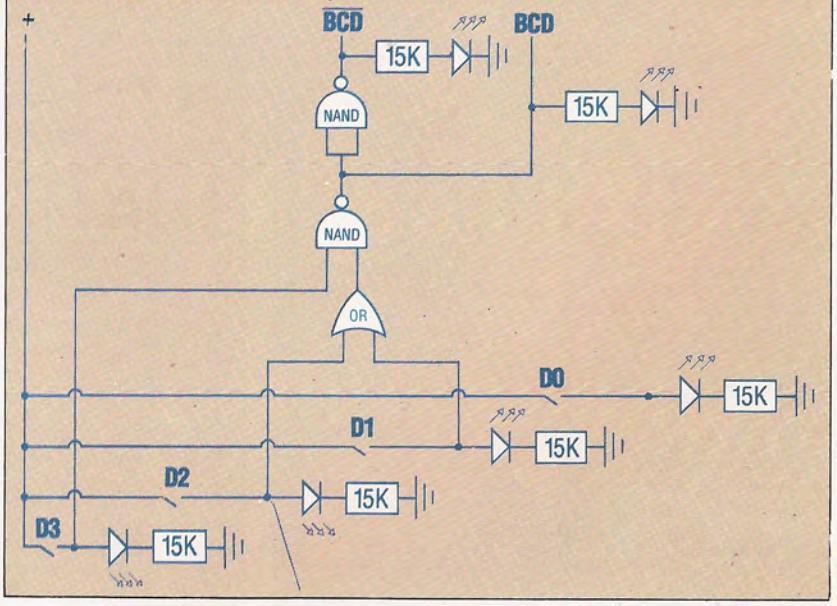
3) De decimal a binario

El convertidor de decimal a binario sólo requiere un circuito integrado: un chip TTL con cuatro puertas OR.



4) Convalidador BCD

Los códigos BCD válidos están comprendidos en la escala entre 0000 y 1001 y los códigos no válidos entre 1010 y 1111. El diseño del circuito necesario sería el siguiente:



Liz Dixon

Un portátil poderoso

El innovador micro Sharp PC-5000 cuenta con una visualización en cristal líquido incorporada y una memoria de burbuja

En cuanto a aspecto externo, el MS-DOS portátil de Sharp parece una pequeña máquina de escribir eléctrica. Pero bajo la tapa del PC-5000 se ocultan, entre otras configuraciones, una visualización en cristal líquido y una memoria de burbuja. Por otra parte, la carcasa del ordenador ha sido diseñada para dar cabida a su propia impresora opcional.

Los ordenadores portátiles asumen muchas formas, pero recientemente el estilo más popular ha sido el del portátil de regazo, como el TRS-80 Modelo 100 y el Epson HX-20. Estas máquinas incorporan una ligera pantalla LCD y pueden funcionar con bastante eficacia alimentadas solamente con pilas. El Sharp PC-5000 es la más cara de las máquinas de esta clase. Responde a esta tendencia de diseño y se adentra en un nuevo territorio al basarse en la memoria de burbuja en vez de en el almacenamiento en cassette.

Cuando hace algunos años se introdujo la idea de la memoria de burbuja, produjo una conmoción en la industria; pero la idea no cuajó en la medida que cabía esperar porque son pocas las compañías que han sido capaces de superar problemas relacionados con la velocidad y la fiabilidad. Sharp parece haber resuelto los problemas, porque su máquina de prueba demostró ser muy rápida y fiable. La memoria de burbuja requiere muy poca potencia y permite el almacenamiento de grandes cantidades de datos en un espacio diminuto. El principio implica almacenar datos en burbujas codificadas magnéticamente.

El PC-5000 tiene la apariencia de una máquina de escribir eléctrica pequeña y portátil. Al abrir la tapa queda a la vista un teclado esculpido tipo máquina de escribir con ocho teclas de función definida por el usuario y cuatro flechas de cursor colocadas elegantemente a través de la parte superior. La tapa está unida por medio de goznes y contiene la pantalla LCD. A pesar de ser bastante pesada, la tapa se mantiene en su sitio mediante un ensamblaje de trinquete, que permite moverla en varias posiciones para mayor comodidad. Arriba del teclado hay un panel con tres luces LED de aviso (potencia, poca pila, burbuja) y una ranura para un paquete de memoria burbuja.

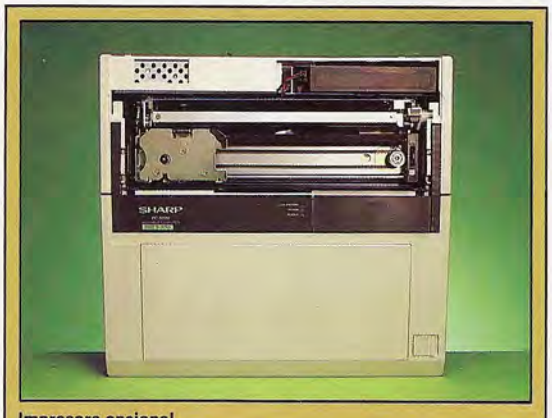
Detrás de este panel hay una bandeja donde se sitúa la impresora opcional. Ésta es una caja rectangular que se inserta limpiamente en la bandeja. Es una impresora térmica que produce 37 caracteres por segundo. Sobre papel sensible al calor, la impresión ofrece muy buen aspecto, si bien el papel desluce en cierto sentido la calidad del documento.

El Sharp PC-5000 tiene 128 Kbytes de memoria para el usuario y 192 Kbytes de ROM, incluyendo BASIC Microsfot GW, la misma versión utilizada en el IBM PC. La CPU es el 8088 de Intel, nuevamente el mismo que el del PC, y el Sharp posee el MS-DOS como sistema operativo. El ordenador direcciona la burbuja como si ésta fuera las unidades de disco A y B. Sharp ofrece unidades de disco flexible gemelas que se pueden enchufar en la parte posterior del PC-5000. Estas unidades se direccionan como C y D y no pueden funcionar con pilas.

El Sharp viene con varios paquetes de software de Sorcim, incluyendo SuperCalc, SuperWriter y SuperComm, un programa de telecomunicaciones. Los programas vienen en un paquete burbuja y se seleccionan del menú del sistema pulsando una de las teclas de función. Los valores de las teclas de función pueden aparecer como etiquetas en la última línea de la pantalla. A Sharp se la conoce por su calidad en cuanto a diseño e ingeniería, y con el PC-5000 ha conseguido colocar un poderoso ordenador en un pequeño paquete.

Ian McKinnell

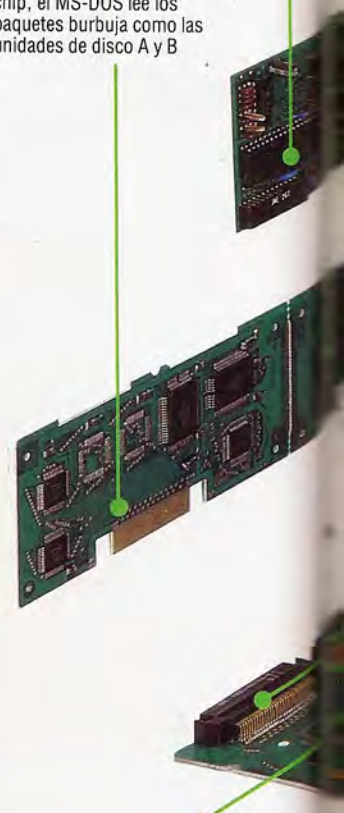
Unidades gemelas de disco flexible
Esta unidad contiene dos unidades de disco de 320 K bajo MS-DOS. Se conecta en la parte posterior del PC-5000, pero no puede funcionar dependiendo de las pilas de la máquina. Al software escrito para otras máquinas MS-DOS se les debe formatear para la visualización en 8 líneas



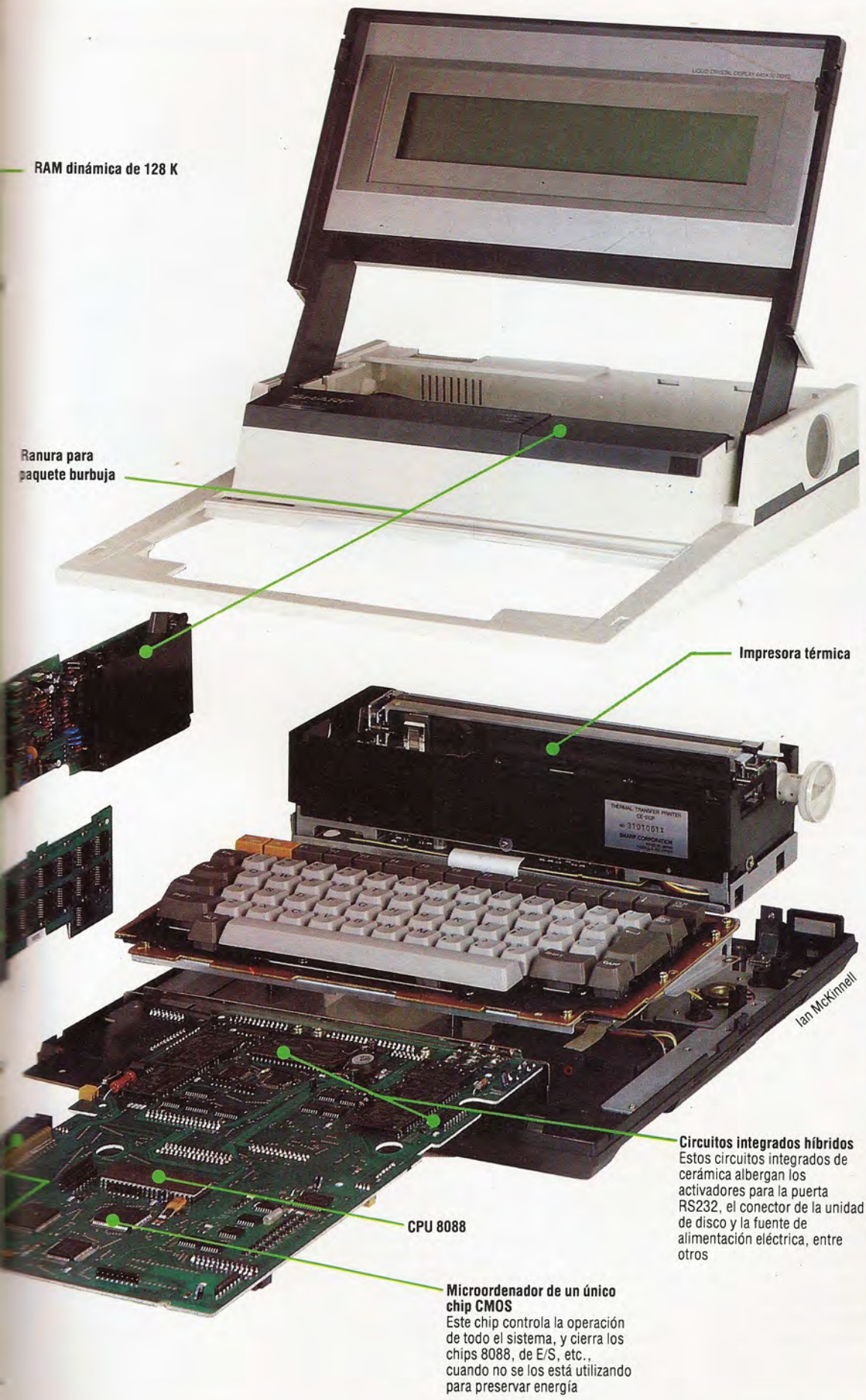
Impresora opcional
Mirando el PC-5000 desde arriba se puede ver la impresora térmica opcional anidada en la carcasa. La impresora encaja exactamente en una pequeña bandeja y se conecta al tablero del sistema mediante un cable plano. Con sus 37 caracteres por segundo la impresora es veloz, y produce una salida de gran calidad sobre papel sensible al calor

Controlador de memoria de burbuja
Esta ficha actúa como un controlador de unidad de disco, excepto en que controla la entrada y la salida desde la ranura del paquete burbuja

Tablero ROM
Este tablero posee 128 K de ROM incluyendo MS-DOS, generadores de caracteres para la visualización y la impresora, algunas comunicaciones y E/S del sistema y el PISCS. El PISCS es un chip que traduce las señales del paquete burbuja a un lenguaje que el MS-DOS entiende. Con la ayuda de este chip, el MS-DOS lee los paquetes burbuja como las unidades de disco A y B



Ranuras para ampliación
Estas ranuras retienen los paquetes de RAM o el software retenido en cartuchos ROM



SHARP PC-5000

DIMENSIONES
300 x 318 x 90 mm

PESO
5,7 kg incluyendo la impresora

CPU
8088 de Intel de 16 bits y CMOS de 8 bits

MEMORIA
128 K de RAM
192 K de ROM

PANTALLA
LCD de 80 caracteres por 8 líneas
Resolución de gráficos de 640 x 80
Caracteres internacionales y para gráficos incorporados

INTERFACES
Cassette, unidad de disco externa, salida para modem, RS232, adaptador para corriente CA, puertas para ROM-RAM

LENGUAJES DISPONIBLES
BASIC Microsoft GW

TECLADO
Estilo máquina de escribir, estándar, de 57 teclas, con 8 teclas de función, teclas para control del cursor, otras tres teclas especiales. Los valores de las teclas de función se definen mediante software y pueden aparecer en la última línea de la pantalla

DOCUMENTACION
Los manuales que se proporcionan incluyen información relativa al montaje del PC-5000 y a la utilización del MS-DOS, y son bastante buenos. El manual de BASIC lo ha escrito Microsoft y es excesivamente técnico para el usuario recién iniciado

VENTAJAS
El 5000 ofrece la mayoría de las facilidades de una máquina de tamaño natural. La impresora incorporada y los cartuchos burbuja, en particular, la colocan por delante de sus rivales

DESVENTAJAS
Existen pocos puntos débiles si uno considera los desarrollos que Sharp ha llevado a cabo. No obstante, el 5000 es pesado para sus dimensiones y más caro que otras máquinas de regazo. Si usted está acostumbrado a los discos flexibles, encontrará un tanto lentas las memorias de burbuja

Último informe

Completamos nuestra serie de capítulos sobre software de gestión exponiendo otros métodos para analizar los datos contables

Ian McKinnell

Los responsables del control de almacén han de contar con información suficiente no sólo de un artículo en particular. Necesitan conocer todos los aspectos del stock actual y de los movimientos de existencias. Hay dos maneras de lograrlo: por medio de consultas en pantalla o a través de informes impresos. El menú de consultas del programa Dragon contiene varias opciones con el tipo de datos disponible y cómo se presentarán.

Una facilidad esencial de consulta o informe es la de los artículos con escaso movimiento de venta. Dado que todas las operaciones son fechadas cuando se digitan en el sistema, el programa dispone de antemano de la información necesaria para generar un informe sobre artículos de movimiento lento. Le bastará con indagar en el archivo que contiene el historial de todas las operaciones y comparar datos. Como cada empresa tiene su propio concepto de "lentitud" (lo que para una es movimiento lento podría ser un ritmo excelente para otra), el informe ha de permitir que el usuario defina los términos.

El sistema Dragon satisface ampliamente este punto. Digitando una fecha determinada (p. ej., 150484, por 15 de abril de 1984), el usuario da al sistema una clave para comenzar la búsqueda. El programa lee y luego imprime todos los artículos sin movimiento de ventas. Esto proporciona una facilidad para informes muy útil, porque se puede generar cualquier número de informes de movimiento lento simplemente proporcionándole al programa una fecha diferente. La única precaución que ha de tomar el usuario es que la fecha esté comprendida dentro de los límites del historial de transacciones del archivo. De no haber ningún artículo de movimiento lento que caiga dentro de la fecha especificada, el mensaje en pantalla del programa Dragon dirá: "NINGUN ARTICULO SEGUN CRITERIO DE SELECCION".

Existe una limitación para este tipo de informes. Sólo detecta artículos de las existencias que no hayan sido objeto de absolutamente ninguna transacción. Pero está claro que en algunos casos un usuario podría muy bien pensar que una determinada línea de existencias del cual se han producido dos ventas en seis meses se debería considerar como de movimiento lento.

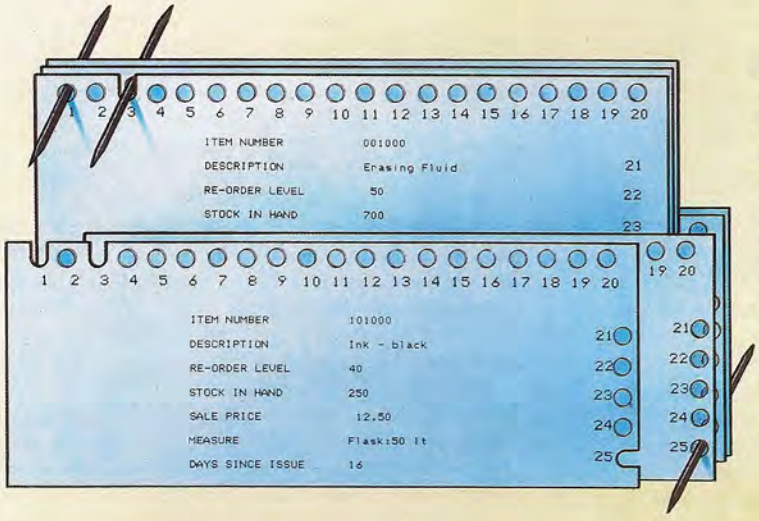
Un sistema más sofisticado proporcionaría una mayor flexibilidad. Hay dos formas de conseguir esto. El sistema podría ofrecer un criterio de selección adicional distinto del de la fecha, como especificar el número de transacciones por debajo del cual un artículo se imprimirá en el informe de movimiento lento. O bien, hacer que el mismo sistema lea y compare los movimientos de todas las partículas, listando, pongamos por caso, los 50 artículos más lentos, luego los siguientes 50 artículos más lentos y así sucesivamente.

Visualización en pantalla

La pantalla visualiza cada artículo por separado. Por consiguiente, si hay un gran número de artículos en existencias de movimiento lento, tener que pasar las páginas de la lista nos llevaría mucho tiempo. En ese caso, por lo tanto, un informe impreso podría muy bien ser una alternativa mejor que una visualización en pantalla, porque es más rápido y más fácil de explorar. El diseñador de sistemas ha de tener esto en consideración cuando se compone o "especifica" un programa de gestión como un paquete de control de existencias.

La visualización del movimiento de existencias contiene una gran parte de información adicional que es importante para el usuario. Refleja el valor de las existencias, lo que le indica al usuario cuánto capital hay comprometido en un almacén o en un

Perforaciones



Por el borde
El sistema de fichas de bordes perforados es una forma simple de base de datos. Los agujeros alrededor de la ficha se tratan como dígitos binarios: un agujero equivale a cero, una ranura es un uno. Aquí las fichas representan un archivo de existencias de una empresa: los agujeros del 1 al 3 representan al grupo de productos, del 4 al 6 son el número de catálogo, de modo que los agujeros del 21 al 25 indican la cantidad de días desde que el artículo salió del almacén por última vez, y el agujero 25 significa 16 días

De movimiento lento
El *Stock control system* de Dragon incluye un módulo de base de datos que genera una variedad de informes acerca del estado del inventario. Mantener grandes existencias de artículos de movimiento lento cuesta dinero, de modo que la firma está buscando el grupo de productos 101 para los artículos que no salieron durante 16 días o más; la búsqueda refleja que las existencias de tinta para impresora son muy grandes y que las ventas son muy bajas, de modo que las existencias se deben reducir rápidamente

Kevin Jones

depósito. Muestra el uso promedio por mes, la cantidad en existencia y la fecha de la última distribución. A partir de esta información, el usuario podrá elaborar una política para renovar las existencias, ofreciéndolas quizá con grandes descuentos.

El principio de elaborar informes de acuerdo a un criterio de selección (la fecha, p. ej.) es fundamental cuando se trata de analizar datos de transacciones e informar sobre ellos. Obviamente, el sistema necesita ser capaz de listar todas las transacciones de todos los artículos. Pero también debería indicar qué ha sucedido con secciones de las existencias entre fechas especificadas.

El programa Dragon lo consigue pidiéndole al usuario que dé entrada a los valores superiores e inferiores de una gama de artículos de las existencias y las fechas de delimitación (p. ej., entre las fechas 010184 y 010484).

Además de contemplar las transacciones habidas con una gama específica de artículos, al usuario podría interesarle, asimismo, una descomposición, con los artículos agrupados de acuerdo al tipo de transacción, o cifras de ventas, o ajustes de existencias, etc. El programa Dragon poseen un menú con tres opciones para las fechas seleccionadas (a saber: sólo transacciones del período actual; todas las transacciones; y transacciones dentro de un período especificado), y luego otro submenú para descomponer la selección por tipos de transacciones. Este submenú es similar al menú de tipo de transacciones que describimos en la página 672, pero también incluye una nueva opción, 99 ALL TYPES (todo tipo), que permite una salida impresa general de todas las transacciones dentro de los parámetros de datos especificados.

El programa permite a los usuarios incluir los artículos individuales a grupos determinados. Así, los usuarios podrán revisar los grupos de productos y sus descripciones. También pueden saber cuáles son las existencias disponibles para cada grupo determinado de productos. Los informes basados en pantalla son útiles, pero no son informes permanentes. Con el fin de proporcionar un registro de "salida impresa", la facilidad de informes duplica en muchos sentidos la facilidad de consultas que ofrece este programa.

Hay una cantidad de razones por las cuales el programa Dragon es un sistema de control de existencias bastante restringido, a pesar de sus detalladas configuraciones analíticas. El sistema está diseñado como un sistema de existencias "en solitario". No puede conectarse con otros programas para aplicaciones de gestión que podrían utilizar la información de los archivos de este programa. No hay ninguna facilidad para señalar las existencias según pedidos. Una de las preguntas más importantes que a los usuarios les agradaría hacer: "¿Tengo suficientes artículos en existencia para atender tales pedidos?" Cuando a usted se le amontonan varias órdenes de clientes por un número de artículos diferentes, es muy difícil llevar de forma manual el registro de las demandas sobre las existencias.

En toda esta serie de capítulos nos hemos concentrado en la utilización de los ordenadores personales para el procesamiento de datos en pequeñas empresas. A estas alturas debería estar claro que a pesar de que estos paquetes podrían ofrecer supuestamente sistemas integrados, con frecuencia se concentran en un aspecto del negocio (p. ej., el movimiento de caja) en detrimento de otros, como, por ejemplo, procesamiento de pedidos, inventario, conformidad. Quizá la mejor oferta de estos paquetes es la de proporcionarles información suplementaria a los directores de empresas.

A pesar de estos inconvenientes, los paquetes de gestión para ordenadores personales pueden mejorar sustancialmente la eficacia de la contabilidad y la organización de una pequeña empresa. Elegidos y empleados con cuidado, dichos paquetes pueden resultar verdaderamente útiles.

Acción en alta mar

Comenzaremos a estudiar la capacidad de gráficos del Commodore 64. Esto nos permitirá construir y analizar un juego muy atractivo

Una de las configuraciones más atractivas del Commodore 64 es su capacidad para escribir juegos de estilo recreativo en BASIC. Eso es posible gracias a la capacidad que tiene la máquina para manipular sprites: formas de alta resolución que el usuario puede definir y manipular fácilmente en la pantalla. En la memoria del Commodore hay registros especiales que controlan los atributos de los sprites, como su color, su tamaño y su posición en la pantalla. Colocando (POKE) números en estos registros, el programador puede controlar la acción con gran facilidad. El juego *Subhunter* que vamos a diseñar como proyecto central de esta serie hará uso de cuatro de los ocho sprites disponibles de la máquina. Estos sprites representarán al barco, al submarino, a las cargas de profundidad disparadas desde el barco y a una explosión.

De modo que el juego se puede ir elaborando a lo largo de sucesivos capítulos, escribiendo cada una de las secciones del programa como breves subrutinas que serán llamadas para su utilización dentro del bucle principal del programa. Así es más sencillo rastrear errores y ampliar el programa.

El concepto de este juego resultará familiar a muchos lectores. El jugador tiene bajo su control un barco que se encuentra a la caza de submarinos. Éstos atraviesan la pantalla a una profundidad y velocidad variables y el barco les arroja cargas de profundidad. El marcador del jugador aumenta cada vez que toca a un submarino, calculándose el valor de cada uno de éstos a tenor de su profundidad y su velocidad. Naturalmente, las marcas más elevadas se consiguen alcanzando a submarinos que se mueven con mayor velocidad y a mayor profundidad. No obstante, si el submarino escapa se resta del marcador del jugador el valor de aquél. El barco se controla desde el teclado, utilizando las teclas Z y X para el movimiento horizontal hacia izquierda y derecha. Las cargas de profundidad se disparan mediante la tecla M. En la pantalla se visualiza un reloj, que permite jugar sólo durante tres minutos. Transcurrido este tiempo se le pregunta al jugador si desea jugar otra vez y se le ofrece un registro de la marca más alta obtenida.

La regla de oro a observar cuando se diseña un juego de acción en BASIC es que el bucle principal del programa sea lo más corto posible. El programa *Subhunter* se vale de las subrutinas para llevar a cabo la mayoría de las funciones del juego. Las únicas funciones que se controlan directamente desde dentro del bucle principal del programa son: actualización del reloj, aceptación de una entrada desde el teclado, desplazamiento del barco y desplazamiento del submarino.

Este diseño de programa es lo suficientemente general como para poderse aplicar a cualquier marca de ordenador, pero la programación detallada variará en función de las características individuales del que se esté utilizando. En esta sección del proyecto analizaremos la rutina que crea una visualización en pantalla.

Visualización en pantalla

En el Commodore 64 hay dos formas de imprimir los caracteres en la pantalla. Una consiste en emplear la sentencia PRINT y la otra en utilizar POKE para colocar números en las zonas de la memoria que retienen información relativa a la visualización. Utilizaremos ambos métodos para la creación del telón de fondo de nuestro juego.

La pantalla Commodore se compone de 25 filas y 40 columnas o espacios para caracteres. En otras palabras, en la pantalla hay 1 000 lugares donde se puede colocar un carácter. Cada posición de la pantalla tiene dos números relacionados con ella. El primero es un número de código de pantalla que le dice al ordenador qué carácter visualizar en esa posición. El segundo es un número de color que le indica al ordenador de qué color debe ser el carácter visualizado. Hay dos bloques de memoria, cada uno de los cuales consta de 1 000 posiciones: uno para retener información relativa al código de pantalla y otro para retener información relativa al color de cada posición de la pantalla. La zona que retiene los códigos de pantalla se denomina *memoria de pantalla* y va desde la posición 1024 a la 2023. La memoria de color va desde la 55296 a la 56295.

Cada carácter posee su propio código de pantalla y éstos están listados en el manual para el usuario. En el Commodore 64 hay 16 colores. Los códigos de color son:

0	negro	8	naranja
1	blanco	9	marrón
2	rojo	10	rojo claro
3	cyan	11	gris 1
4	púrpura	12	gris 2
5	verde	13	verde claro
6	azul	14	celeste
7	amarillo	15	gris 3

Además de estas zonas de la memoria, hay otras dos posiciones que son de especial interés. La posición 53280 controla el color del borde y la 53281 controla el color de la pantalla. Para diseñar el paisaje marino que necesitamos para nuestro juego, las seis filas superiores de la pantalla serán de color celeste para representar el cielo, mientras que el resto de la pantalla y el borde serán azul oscuro para evocar el mar (excepto las dos filas inferiores, que representarán el lecho marino).

Para establecer el color de la pantalla en celeste y el color del borde en azul oscuro, se requiere el siguiente par de órdenes POKE:

```
POKE53281,14:POKE53280,6
```

La séptima fila de la pantalla comienza en la posición 1264. La segunda fila por la parte inferior comienza en la posición 1944. Podemos colorear el mar colocando (POKE) el código de pantalla para un espacio invertido (un carácter espacio bloqueado) en las posiciones 1264 a 1943, al tiempo que se coloca (POKE) 6 en las correspondientes posiciones de memoria de color. Existe una forma sencilla de conectar una posición de memoria de color con una correspondiente posición de memoria de pantalla: sumándole simplemente 54272 a la posición de memoria de pantalla.

El código de pantalla para un carácter espacio es 32. El código de pantalla para un carácter invertido se puede calcular sumando 128 al código de pantalla normal, de modo que el código para un espacio invertido es 32 + 128 = 160. Las siguientes líneas del programa se valen de un bucle FOR...NEXT simple para colorear el mar:

```
FOR I = 1264 TO 1943
POKE I,160:POKE I + 54272,6
NEXT I
```

El lecho marino se compone de dos filas de un espacio cuadrado con un código de pantalla de 102, de color marrón. Nuevamente, un simple bucle FOR...NEXT se encargará de esto:

```
FOR I = 1944 TO 2023
POKE I,102:POKE I + 54272,9
NEXT I
```

La sentencia PRINT también es un método eficaz para producir visualizaciones en pantalla. El color y el posicionamiento del cursor se pueden controlar desde ella mediante los caracteres de control especiales de Commodore o bien utilizando códigos CHR$. Utilizaremos este último método, porque es más fácil de leer en los listados de programas. En un apéndice del manual del usuario hay una relación completa de los códigos CHR$. A nosotros nos interesan aquellos que se refieren al color y al posicionamiento del cursor:

CHR$(5)	color blanco
CHR$(144)	color negro
CHR$(147)	limpia la pantalla y posiciona el cursor en el ángulo superior izquierdo
CHR$(19)	posiciona el cursor en el ángulo superior izquierdo
CHR$(17)	desplaza el cursor un lugar hacia ABAJO
CHR$(145)	desplaza el cursor un lugar hacia ARRIBA
CHR$(157)	desplaza el cursor un lugar hacia la IZQUIERDA
CHR$(29)	desplaza el cursor un lugar hacia la DERECHA

Como parte de nuestra rutina de organización de la pantalla se ha de imprimir MARCADOR y MARCADOR MAX en la línea superior de la pantalla. CHR$(19) asegurará que el cursor esté al comienzo de la línea superior. La siguiente orden imprime el marcador inicial en negro:

```
PRINT CHR$(19);CHR$(144);"MARCADOR 000"
```

El Subhunter en acción
El programa que estamos desarrollando en esta serie es el clásico juego, con gráficos por ordenador, de caza de submarinos con cargas de profundidad. Se utilizan los gráficos sprite del Commodore 64 para dar una animación continua al submarino y al barco

Ian McKinnell

MARCADOR MAX también se posiciona en la línea superior, pero sobre el lado derecho. La función SPC permite insertar un número de espacios. La orden PRINT ahora se puede alterar para que incluya MARCADOR MAX:

```
PRINT CHR$(19);CHR$(144);
"MARCADOR 000";SPC(16);"MARCADOR MAX
  000"
```

La rutina de preparación de la pantalla formará una subrutina para el programa principal, empezando en la línea 1000. También se incluye una orden POKE que hace que todas las teclas se repitan cuando se pulsen. Esto se utilizará cuando analicemos la rutina de control del teclado. Esta subrutina se puede comprobar mediante las siguientes líneas:

```
10  GOSUB 1000: REM PREPARACION PANTALLA
20  END
1000  REM ****PREPARACION PANTALLA****
1010  PRINT CHR$(147):REM LIMPIAR PANTALLA
1020:
1030  REM**COLOR MAR**
1040  POKE53281,14:POKE53280,6
1050  FOR I = 1264T01943
1060  POKE I,160:POKE I + 54272,6
1070  NEXT
1080:
1090  REM**FONDO MAR**
1100  FOR I = 1944T02023
1110  POKE I,102:POKE I + 54272,9
1120  NEXT
1130  POKE 650,128:REM REPETIR TECLAS
1140:
1150  REM**MARCADOR**
1160  PRINTCHR$(19);CHR$(144);"MARCADOR
      000";SPC(16);"MARCADOR MAX 000"
1170  RETURN
1180:
1190:
```

Después de que se ha dado entrada a la rutina, es una buena idea guardarla en cinta o en disco antes de ejecutarla.

Instrucciones para el contador

Aprender a utilizar de manera adecuada etiquetas y bucles es fundamental para todo aquel que se proponga programar eficientemente en lenguaje máquina

Los bucles y las bifurcaciones condicionadas se ejecutan en lenguaje assembly utilizando los flags del registro indicador de estado con objeto de probar la condición del acumulador, y las instrucciones del salto relativo para modificar el flujo de control del programa. En la creación de tablas de datos se combinan a un tiempo estas estructuras y la modalidad de direccionamiento indexada.

Antes de que podamos utilizar provechosamente las diversas modalidades de direccionamiento de la CPU (en especial las direcciones indexadas), debemos primero ser capaces de escribir un bucle. Sin esta estructura fundamental estamos en una situación bastante similar a la de un programador de BASIC que sabe mucho de matrices pero que ignora la orden FOR...NEXT. En lenguaje assembly no hay estructuras automáticas como FOR...NEXT (aunque hay una instrucción para el Z80 que se aproxima mucho a la misma), pero podemos construir bucles de tipo IF...THEN GOTO... Éstos requieren instrucciones que tomen decisiones o expresen condiciones y, efectivamente, cambien el orden por el cual se obedecen las órdenes en el programa.

La toma de decisiones en lenguaje assembly se centra en los flags del indicador de estado del procesador. Tales flags, o banderas, muestran los efectos en el acumulador de la última instrucción ejecutada y en algunas ocasiones se denominan *flags de condición*. Todos ellos se pueden utilizar en la toma de decisiones, pero por el momento necesitamos considerar sólo dos de los mismos: el flag de cero (Z) y el de arrastre (C).

El estado de estos flags se utiliza para decidir si el procesador ejecuta la siguiente instrucción del programa, o si salta a alguna otra instrucción de éste. El procesador decide si continuar o saltar, bien cambiando o bien aceptando la dirección que contiene su contador del programa. Este registro siempre contiene la dirección de la siguiente instrucción en lenguaje máquina a obedecer. Cuando el procesador empieza a ejecutar una instrucción, carga en el contador del programa el opcode de la instrucción contenido en el byte cuya dirección es indicada por el contador. La dirección del registro se incrementa en razón del número de bytes de la instrucción, de modo que el contador del programa señala entonces el opcode de la siguiente instrucción. Si la que está en curso hace que el contador del programa señale a una dirección de algún otro lugar del programa, entonces se genera efectivamente un salto.

En el 6502, la instrucción BEQ hace que el contador del programa cambie si el flag de cero está activado. BCS es la instrucción equivalente en el caso de que el flag de arrastre esté activado. En el Z80, estas instrucciones son JR Z y JR C, respectivamente. Estos cuatro opcodes se denominan *instrucciones de bifurcación*, porque representan un punto de bifurcación en el flujo de control del programa. Su operando es un número de un único byte, que se le suma a la dirección del contador del programa con objeto de indicar una nueva dirección. Veamos a continuación lo que sucede cuando se ejecuta el siguiente programa:

ORG $5E00			
6502		Z80	
5E00	ADC #$34	ADC	A,$34
5E02	BEQ $03	JR	Z,$03
5E04	STA $5E20	LD	($5E20),A
5E07	RTS	RET	

Si la instrucción ADC en $5E00 produce en el acumulador un resultado cero (poco probable, pero posible, como ahora veremos), entonces las instrucciones BEQ y JR Z en $5E02 harán que se le sume $03 al contenido del contador del programa. La siguiente instrucción a ejecutar, por lo tanto, será RTS, la instrucción de retorno en $5E07, saltándose la instrucción en $5E04.

A primera vista esto podría parecer erróneo. Después de todo, si la instrucción en $5E02 hace que se le sume $03 al contador del programa, la dirección almacenada será sin duda $5E05. En este punto resulta imprescindible tener siempre presente que el contador del programa lo que señala es la *siguiente* instrucción a ejecutar y no la instrucción que se está realizando en cada momento. Por consiguiente, cuando comience la ejecución de la instrucción en $5E02, el contador del programa contendrá la dirección $5E04: la posición de la siguiente instrucción. Si a $5E04 se le suma $03, el resultado será $5E07, la dirección de la instrucción siguiente.

Vale la pena recalcar que el procesador no sabe verificar si las direcciones que se le indican son las correctas. Si, por descuido, cambiáramos el desplazamiento de la instrucción a $02, entonces el contador del programa se incrementará (si el acumulador contiene cero) en $02, y el procesador considerará $5E06 como la dirección del opcode de la siguiente instrucción. En nuestro programa correcto, $5E06 contiene el valor $5E, que es el byte *hi* del operando de la instrucción contenida en $5E04. El procesador no puede evaluar si es la instrucción correcta o no. Para él, $5E es un opcode válido y procederá a eje-

cutarlo, tomando los bytes que siguen a $5E06 como los operandos de la instrucción. A resultas de lo cual el programa desbarrará. Calcular mal, como aquí, los desplazamientos es uno de los errores más comunes que se producen en la programación máquina.

Sin embargo, en la programación con assembly calcular los desplazamientos de salto no constituye ningún problema, porque el programa ensamblador lo puede hacer por nosotros. Y así, en vez de escribir un byte hexa como operando de la instrucción de bifurcación, damos la dirección simbólica de la instrucción a la cual saltar. Con esto resulta mucho más fácil seguir el programa en lenguaje assembly. El ensamblador decodifica la dirección simbólica otorgándole una dirección absoluta, calcula el desplazamiento necesario para llegar a la dirección y escribe ese desplazamiento en la instrucción en lenguaje máquina. La dirección simbólica se denomina *etiqueta* y es análoga a un número de línea de un programa en BASIC.

Analicemos con más detalle el empleo de las etiquetas. Una etiqueta es una serie alfanumérica escrita al principio de una instrucción en lenguaje assembly. El programa ensamblador la trata como un símbolo de dos bytes que representa la dirección del primer byte de la instrucción. Veamos ahora de qué forma quedaría el programa dado aplicando esta nueva técnica:

ORG $5E00		
	6502	Z80
5E00	ADC #$34	ADC A,$34
5E02	BEQ EXIT	JR Z,EXIT
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

Ahora la instrucción en $5E02 es como si dijera: "si (IF) el valor del acumulador es cero, entonces ir (THEN GOTO) a la dirección representada por la etiqueta EXIT". Vemos que en comparación con la versión previa, hemos mejorado la legibilidad del programa y hace que disminuyan en buena parte las posibilidades de calcular mal el destino del salto.

Ahora ya estamos en disposición de poder utilizar las etiquetas y las instrucciones de bifurcación para crear un bucle:

ORG $5E00		
	6502	Z80
5E00 START	ADC #$34	ADC A,$34
5E02	BNE START	JR NZ,START
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

Observe aquí la utilización de la nueva etiqueta, START, así como las nuevas instrucciones de bifurcación: BNE, que significa "bifurcar si el acumulador no es igual a cero" (*Branch if Not Equal*); y JR NZ, que significa "saltar si el acumulador no es igual a cero" (*Jump if Not Zero*). Consideremos el efecto que tendrá este código. El programa primero sumará $34 al acumulador. Si el resultado no es igual a cero, entonces el programa se bifurca hacia atrás, hasta $5E00 (la dirección donde se colocó la etiqueta START). $34 se le volverá a sumar al acumulador, y el resultado decidirá si se produce de nuevo ese retroceso. El bucle se repetirá una y otra vez mientras se satisfaga la condición negativa de la bifurcación. Cuando el contenido del acumulador sea igual a cero como resultado de una instrucción ADC, entonces no se producirá la bifurcación en $5E02 y se ejecutará seguidamente la instrucción contenida en $5E04.

Esto es exactamente como un bucle IF...THEN GOTO... de BASIC. Sólo que quizá usted no acabe de creer cómo el acumulador puede convertirse alguna vez en cero. Porque, lo que sí está claro es que ¡se está incrementando en razón de $34 cada vez que se ejecuta el bucle! ¿Cómo es posible que dé cero alguna vez? La respuesta se basa en el hecho de que el acumulador es en definitiva un registro de un único byte, y si la suma da como resultado un número de dos bytes, entonces se activará el flag de arrastre en el indicador de estado del procesador y el acumulador sólo retendrá el byte *lo* del resultado. Si en un determinado momento el acumulador contiene $CC, por ejemplo, entonces al sumarle $34 se obtendrá el número de dos bytes $0100. Se activará el flag de arrastre y el acumulador retendrá el byte *lo* de este resultado: $00. He aquí cómo el contenido del acumulador será cero a resultas de lo cual el flag de cero se activará.

Teniendo en mente este resultado, podríamos reescribir el programa para utilizar una condición de bifurcación diferente, incorporando el estado del flag de arrastre en lugar del estado del flag de cero.

ORG $5E00		
	6502	Z80
5E00 START	ADC #$34	ADC A,$34
5E02	BCC START	JR NC,START
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

En esta versión, la instrucción en $5E02 significa: "si el flag de arrastre se activa, bifurcar a START". Enseguida que el resultado de sumarle $34 al acumulador sea mayor que $FF, el flag de arrastre quedará activado y dejará de producirse la bifurcación hacia atrás hasta la dirección START.

Contadores de bucles

Puede que le parezca bastante limitada esta posibilidad de bifurcación según el estado del flag de arrastre o de cero, pero nos permite un amplio margen de maniobra, tal como veremos enseguida. Lo que sí que deberá estar echando en falta es un *contador de bucle*. Es posible que necesite, por ejemplo, contar el número de veces que se efectúa un bucle antes de que se produzca la condición de salida, o que desee realizar una salida del bucle al cabo de un número dado de iteraciones. Lo primero se alcanza fácilmente con un registro de índice de la CPU que guarde el contador, más una instrucción de incremento para actualizarlo:

6502		
0000	ORG	$5DFD
5DFD	LDX	#$00
5DFF START	INX	
5E00	ADC	#$34
5E02	BCC	START
5E04	STX	$5E20
5E07 EXIT	RTS	

Z80		
0000	ORG	$5DFA
5DFA	LD	IX,$0000
5DFE START	INC	IX
5E00	ADC	A,$34
5E02	JR	NC,START
5E04	LD	($5E20),IX
5E08 EXIT	RET	

El programa presenta ahora las novedades que pasamos a explicarle. En primer lugar, las instrucciones que escribíamos al principio del programa requieren una nueva dirección ORG. El efecto de tales instrucciones sabemos que es muy similar tanto en el 6502 como en el Z80, pero sus longitudes son diferentes, de modo que las direcciones de su posición ya no son las mismas en las dos versiones del programa.

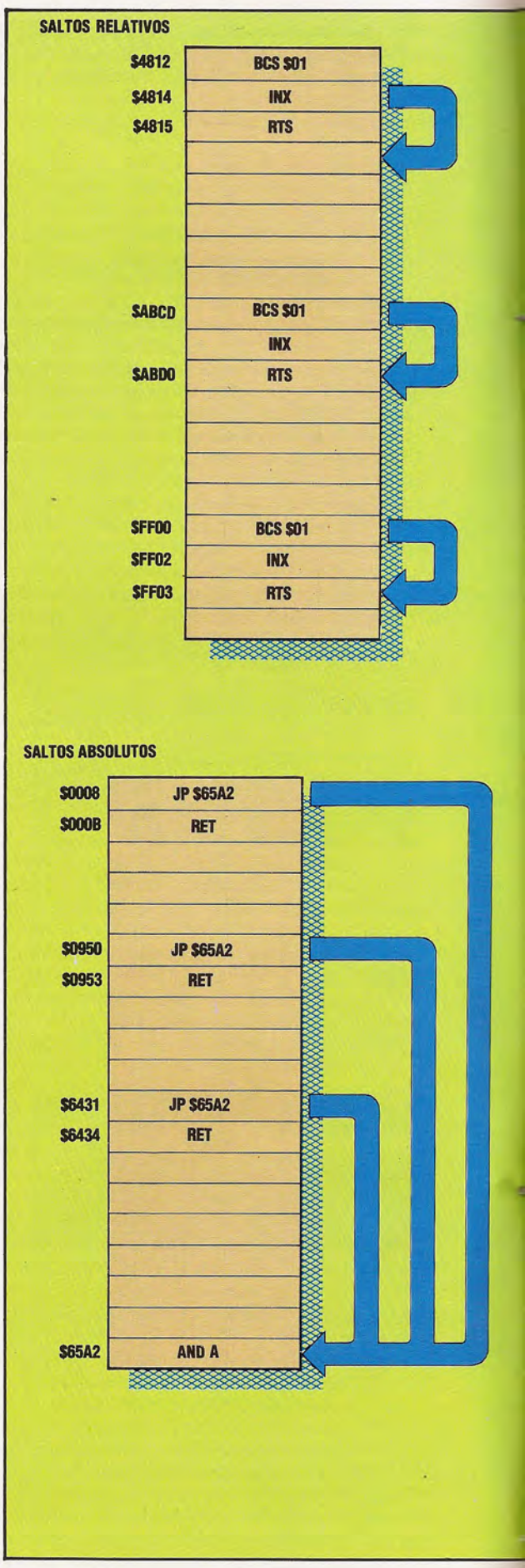
En segundo lugar, se ha utilizado una nueva versión de las instrucciones cargar (LDX, LD IX) y almacenar (STX, LD(), IX) para colocar en el registro índice de la CPU un valor inicial de $00. El registro X del 6502 es un registro de un solo byte, pero el registro IX del Z80 es de dos bytes. Los registros índice poseen funciones especiales, pero esencialmente son RAM de la CPU, al igual que el acumulador, y aquí los utilizamos como acumuladores suplementarios donde colocaremos el contador de bucle. Cuando se produce la salida del bucle, el contenido del registro X del 6502 se almacenará en $5E20. En la versión Z80, el byte *lo* del registro (de dos bytes) IX se almacenará en $5E20 y el byte *hi* en $5E21.

En tercer lugar, una instrucción completamente nueva ha ocupado el lugar de la instrucción ADC como comienzo (START) del bucle: tanto INX como INC IX son instrucciones de incremento, para que el contenido del registro índice aumente (o se INCremente) en razón de $01. Con ésta actualizamos el valor del contador del bucle cada vez que este último se ejecuta.

Podemos "traducir" así el programa: "poner el contador del bucle a cero, empezar el bucle incrementando el contador, agregar $34 al acumulador y bifurcar hacia atrás hasta el comienzo del bucle si el flag de arrastre se ha activado; de lo contrario almacenar el contenido del contador del bucle en $5E20". Una pequeña y ulterior modificación del programa aumentará enormemente su utilidad y su alcance:

6502		
0000	ORG	$5DFA
5DFA	LDX	#$00
5DFC START	STA	$5E22,X
5DFF	INX	
5E00	ADC	#$34
5E02	BCC	START
5E04	STX	$5E20
5E07 EXIT	RTS	

Z80		
0000	ORG	$5DF7
5DF7	LD	IX,$5E00
5DFB START	LD	(IX+$22),A
5DFE	INC	IX
5E00	ADC	A,$34
5E02	JR	NC,START
5E04	LD	($5E20),IX
5E08 EXIT	RET	



Saltos relativos
La mayoría de las instrucciones de bifurcación, como BCS ("bifurcar si el flag de arrastre está activado") y JR NZ ("saltar —*jump*— si el acumulador no es cero"), actúan en función de la situación del indicador de estado del procesador y utilizan la modalidad de salto relativo para reorientar el flujo de control a través del programa. La alternativa es el salto absoluto. En el ejemplo, la instrucción BCS $01 siempre produce un salto relativo de un byte hacia adelante (siempre, claro está, que se produzca efectivamente algún salto, pues depende del estado del flag de arrastre), con independencia de la dirección de posición en donde resida el lenguaje máquina. Aquí, la instrucción BCS $01 siempre va seguida por la instrucción INX, que en sí misma es una instrucción de byte único; por consiguiente, cuando el flag de arrastre está activado, se salta INX

Saltos absolutos
En este ejemplo, la instrucción JP $65A2 produce un salto incondicional cada vez que se pasa por ella. Su efecto consiste en dirigir de nuevo la ejecución del programa hasta la dirección que conforma su operando: $65A2, en este caso. No se indica ninguna condición y la dirección que tiene la posición de la instrucción en el momento de la ejecución carece de importancia; la ejecución del programa continúa siempre a partir de la dirección especificada.
Ambas modalidades de salto poseen sus ventajas y sus inconvenientes, pero el criterio más importante al elegir entre un salto relativo o un salto absoluto es la posibilidad de volverse a posicionar: en la programación en lenguaje assembly es bastante común escribir una rutina y ensamblarla en una dirección ORG, volviéndola a utilizar luego en la misma forma pero con un valor ORG distinto. Si todos los saltos de la rutina son relativos, entonces cambiar las direcciones de posición de las instrucciones no importará en absoluto y el programa fluirá uniformemente a lo largo de sus rutas previstas; no obstante, si alguno de los saltos es absoluto, cuando la rutina se ensamble en una ORG diferente, los saltos seguirán enviando el control a la dirección especificada, que bien podría no tener la más mínima importancia para la rutina. Los saltos relativos son reposicionables, los saltos absolutos no.

Tanto la versión 6502 como la Z80 producen el mismo efecto: crean en la posición $5E22 una tabla de almacenamiento de los sucesivos valores del acumulador a medida que se va ejecutando el programa, y finalmente almacenan en $5E20 el valor final del contador del bucle, que es, asimismo, el número de bytes de la tabla que comienza en $5E22.

La versión 6502 lo consigue con la instrucción STA $5E22,X, que significa "sumar el contenido del registro X a la dirección base, $5E22, luego almacenar el contenido del acumulador en la dirección así formada". Aquí la instrucción STA está en modalidad indexada directa absoluta (véase p. 677): es decir, se utiliza el registro X como un índice para modificar la dirección base, $5E22. Dado que el registro X está inicializado en $00 y se incrementa subsiguientemente a cada vuelta, el valor de comienzo del acumulador se almacenará en $5E22, el siguiente valor en $5E23, y así sucesivamente. Una vez decidida la salida del bucle, STX almacena el valor final del contador del bucle en la posición $5E20.

La versión Z80 utiliza el registro IX como un señalador de la dirección de almacenamiento corriente, mientras continúa utilizando al byte *lo* de IX como contador del bucle. La instrucción LD IX, $5E00 coloca la dirección base, $5E00, en el registro IX, de modo que el byte *lo* de IX contendrá $00. La instrucción LD (IX + $22),A, que tiene un aspecto tan peculiar, significa "sumar $22 a la dirección contenida en IX y almacenar el contenido del acumulador en la dirección así formada". Puesto que el registro IX está inicializado en $5E00, y se incrementa subsiguientemente a cada vuelta del bucle, el valor de comienzo del acumulador se almacenará en $5E22, el valor siguiente en $5E23, y así sucesivamente.

Mientras tanto, el byte *lo* de IX registra el número de vueltas del bucle, hasta que finalmente se almacena en $5E20 cuando termina el bucle. Aquí la instrucción LD (IX + $22),A está en la modalidad de direccionamiento indexada indirecta absoluta, que es bastante más complicada que la versión 6502 pero mucho más eficaz.

Hemos analizado con cierta profundidad el bucle y las estructuras de matriz en lenguaje assembly. Ambas técnicas de programación son de inmenso valor para el lenguaje máquina. En el próximo capítulo del curso las aplicaremos.

El próximo capítulo de nuestro curso de lenguaje máquina, en efecto, va a representar un nuevo e importante paso hacia adelante en este proceso cuya meta final será la generalización de nuestros programas en este código. Muchos fragmentos de programas pueden muy bien convertirse en subrutinas adaptables a las necesidades de la programación principal. No basta simplemente con saber cómo se realizan las bifurcaciones y los bucles, sino que debemos también conocer la manera en que las rutinas que hemos creado puedan "rescatarse" y a continuación utilizarse en cualquier momento que las necesitemos. Este planteamiento nos llevará al examen, en la siguiente lección del curso de lenguaje máquina, de uno de los dispositivos más útiles de la CPU, la pila.

Finalizada la investigación, el curso quedará completo en sus bases más imprescindibles prefijadas por nosotros, después de un estudio sencillo pero exhaustivo, por una parte, de las cuatro operaciones aritméticas por excelencia y, por otra, de las rutinas de visualización.

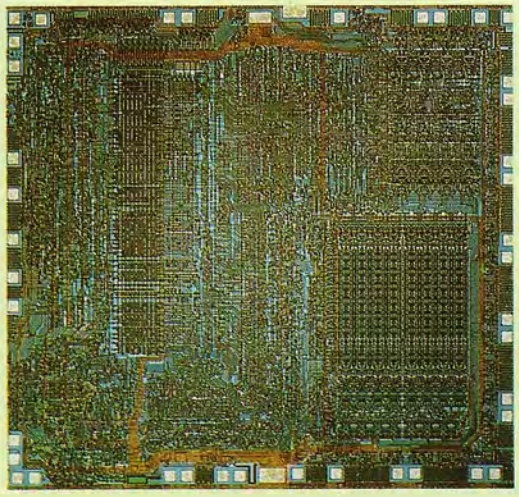
Las instrucciones de bifurcación condicionada, como hemos visto, dependen del contenido del registro de estado del procesador. Una de las razones por las que agregamos la opción de visualización binaria al programa monitor (véanse pp. 598 y 678) fue para permitirle a usted inspeccionar el contenido del PSR (indicador de estado) antes y después de ejecutar una instrucción, y observe los cambios en los flags. Ni en el 6502 ni en el Z80 hay ninguna instrucción para almacenar el contenido del PSR, de modo que hemos de utilizar estas órdenes:

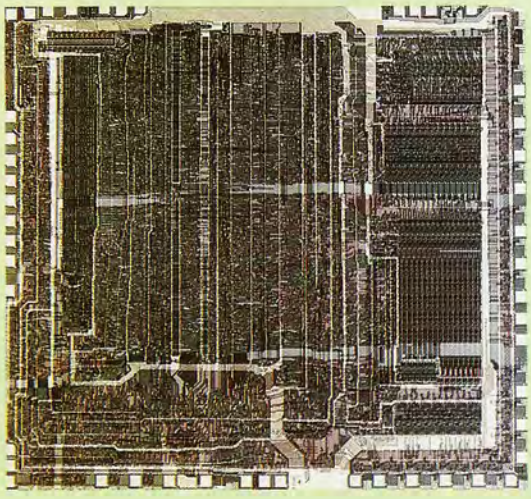
estas órdenes:	**Z80**
3E00 F5	PUSH AF
3E01 F5	PUSH AF
3E02 E1	POP HL
3E03 22 lo hi	LD (STORE1),HL
3E06 F1	POP AF
	6502
3E00 48	PHA
3E01 08	PHP
3E02 48	PHA
3E03 08	PHP
3E04 68	PLA
3E05 8D lo hi	STA STORE1
3E08 68	PLA
3E09 8D lo' hi'	STA (1+STORE1)
3E0C 28	PLP
3E0D 68	PLA

Esta secuencia de instrucciones hará que el contenido corriente del PSR se almacene en el byte direccionado por STORE1 (una dirección apropiada para su máquina), mientras que el contenido del acumulador se almacenará en (1 + STORE1). Para utilizar estas instrucciones, simplemente insértelas como un bloque antes y después de la instrucción del programa cuyo efecto usted desea observar. Debe acordarse, no obstante, de sumarle dos al valor de STORE1 cada vez que inserte este bloque. Después de haber ejecutado el programa puede emplear el monitor para visualizar la sección de la memoria en donde ha almacenado los diversos contenidos del PSR y del acumulador.
Quizá haya pensado que este bloque se debería tratar como una rutina en vez de darle entrada repetidamente donde se necesita. En lenguaje assembly hay un equivalente al GOSUB del BASIC, pero utilizarlo aquí complicaría las cosas, dado que emplea la pila y ésta interferiría con la utilización, por parte del bloque, de la misma lista (PLA, PUSH, PHP, etc., son todas manipulaciones de la pila, que explicaremos más adelante con detenimiento). Observe la diferencia, en cuanto a longitud, del código Z80 y el 6502: los responsables de esta variación son los registros de dos bytes del Z80 y las instrucciones relacionadas con los mismos.

En vanguardia

El microprocesador Z80, junto con su competidor, el 6502, están convirtiendo en realidad lo que se creía fantasía de la ciencia-ficción: un ordenador en cada hogar





Juego de chips
Estas fotomicrografías ilustran la complejidad de los circuitos de los microprocesadores. A la izquierda vemos un imagen enormemente ampliada del tan exitoso Z80 de Zilog y, a la derecha, un chip más reciente perteneciente a la serie Z8000. Observe que el Z8000, al ser un chip de 16 bits, tiene un diseño más denso y más conexiones alrededor del borde del chip

Frank de Weeger, presidente de Zilog

Cuando Zilog Incorporated introdujo el microprocesador Z-80 en 1977, casi nadie intuyó que comenzaba una revolución. Sin embargo, en pocos años, el Z-80 y su principal competidor, el 6502, convertirían en realidad lo que antes pareció mera fantasía: la presencia de un ordenador en cada hogar.

La historia de Zilog empieza a principios de los años setenta, cuando Frederico Faggin y Masatoshi Shima, dos empleados de la fábrica de microchips Intel, se independizan para formar su propia empresa. Ambos habían desempeñado un importante papel en el desarrollo del microchip 8080A (considerado como el primer "ordenador en un chip"), valiéndose de su experiencia empezaron a trabajar en un nuevo desarrollo del microprocesador. El 8080 ya se estaba haciendo muy popular entre los diseñadores y aficionados a la informática, de modo que Faggin y Shima, decidieron diseñar un nuevo chip que fuera compatible con el 8080. De esta manera, podría sacar partido de la gran cantidad de software que ya se había escrito para el 8080. Aprovechando el detallado conocimiento que poseían acerca del 8080, pudieron ampliar el juego de instrucciones (la lista de éstas en lenguaje máquina que contiene el microprocesador) mediante la instrucción de registros adicionales, opcodes de dos bytes y otras técnicas. El microprocesador, el Z80, representó una considerable mejora.

Esta revolución de hardware coincidió con una revolución paralela de software. En 1972, Gary Kildall y John Torode escribieron un programa llamado Control Program/Monitor, o CP/M, que permitió que un microprocesador manipulara unidades de disco flexible, introducidas recientemente. Dado que Kildall era consultor de Intel, el programa estaba diseñado para funcionar en el 8080 y el 8085. El CP/M se fue convirtiendo rápidamente en el sistema predominante de manipulación de discos para microordenadores, y con su nuevo y poderoso microprocesador compatible con el 8080, Zilog estaba en condiciones ideales para sacar partido de la afluencia hacia el software CP/M.

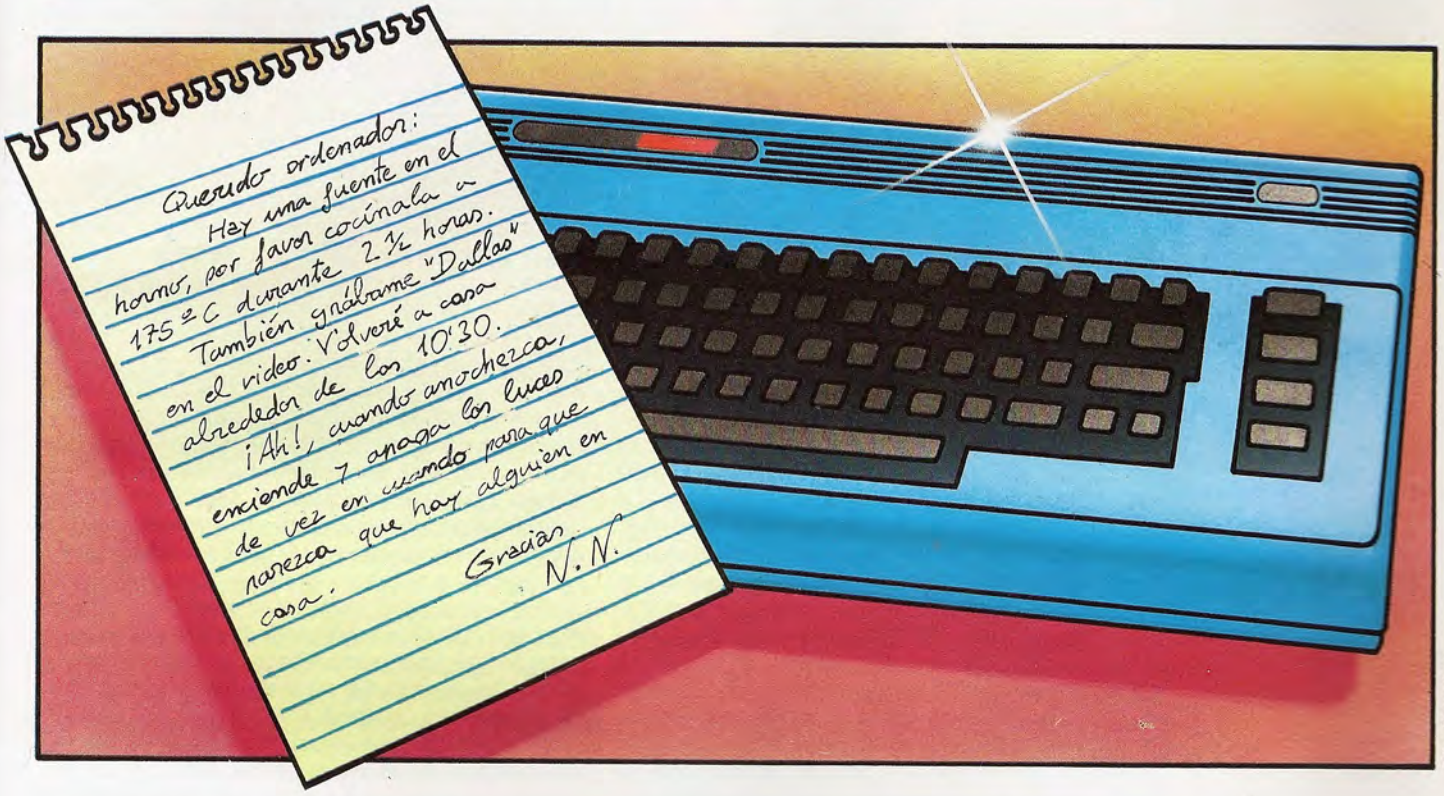
En los años siguientes el camino de Zilog no ha seguido la misma línea. A pesar de que el Z80 se sigue vendiendo en grandes cantidades (la empresa todavía produce alrededor de un millón por mes), los intentos por mejorar el chip para el mercado de 16 bits han tenido controvertidos resultados.

El primer intento de Zilog por conseguir un procesador de 16 bits fue el Z8000. Aunque por lo geneal se lo reconoce como un dispositivo muy potente, con un vasto juego de instrucciones y un gran número de registros, demostró ser un chip sumamente complejo de programar. El Z8000 se vio enfrentado a varios obstáculos importantes. Para empezar, a pesar de ser compatible con el Z80000, o Z80K (el procesador de 32 bits aún por lanzar al mercado), no lo era con el Z80 y, por lo tanto, no pudo sacar partido de la gama y la diversidad de programas que se habían escrito para éste en los años precedentes. Esto tuvo como consecuencia que aquellos fabricantes interesados en versiones mejoradas de 16 bits para sus máquinas se decidieran por incorporar microchips menos exigentes.

A la vista de la escasa aceptación del Z8000 en el mercado del microordenador, Zilog optó por diseñar un nuevo modelo. La empresa está a punto de lanzar el procesador de 16 bits Z800, que es compatible con el Z80. Las expectativas también parecen ser promisorias en otros aspectos: Commodore ha anunciado que su nueva gama de máquinas de oficina estará basada en el Z8000.

Su seguro servidor

Ya es posible hallar en el mercado componentes que permitan crear dispositivos que realicen las más diversas tareas



Mike Brownlow

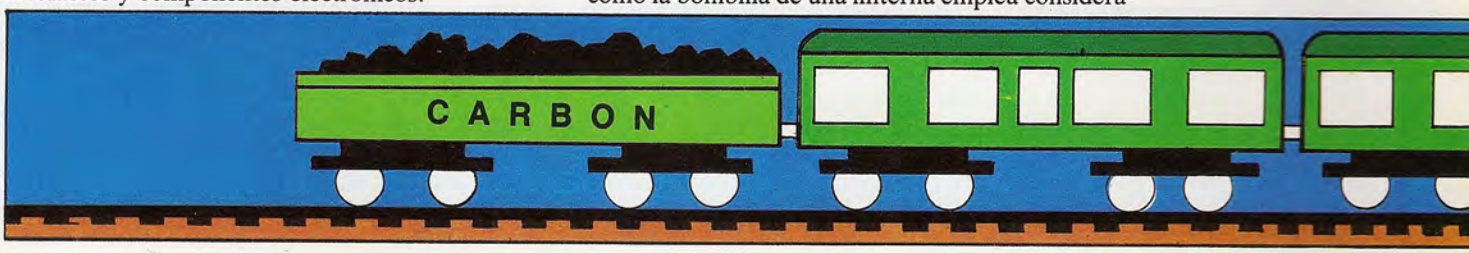
La justificación para montar un micro con el fin de que abriera las cortinas por la mañana o de que regara las plantas cuando uno estuviera de vacaciones podría ser, simplemente, porque hacerlo resulta divertido. Y no hay nada de criticable en realizar algo porque a uno le agrada hacerlo.

En la actualidad, construirse sus propios dispositivos periféricos se podría considerar como "juguetear" con artefactos de fabricación casera, pero a la larga esta actividad podría resultar productiva. Mucha gente piensa que los robots controlados por ordenador y otros dispositivos serán pronto una parte integral en nuestras vidas (del mismo modo que los ordenadores se han convertido en algo cotidiano en apenas cinco años), y, por consiguiente, las habilidades aprendidas ahora podrían tener un valor incalculable en el futuro. Después de todo, empresas informáticas gigantescas, como Apple y Atari, nacieron en los garajes particulares de personas que trabajaban informalmente y a su aire con artefactos y componentes electrónicos.

Cualquier sistema controlado por ordenador precisa de varios elementos. Obviamente, son primordiales la propia máquina y el artículo que esté bajo su control. También es indispensable que haya algún medio en virtud del cual el ordenador le transmita al dispositivo los mensajes de control, y el software que permita que el ordenador decida cuáles han de ser esos mensajes. Y esto es sólo la mitad del proceso. El ordenador suele precisar de algún sistema para medir el efecto que está teniendo su control, con el fin de realizar los ajustes necesarios. Esto se conoce como *realimentación* (*feedback*) y sin ella el ordenador sería tan inútil como un conductor de automóvil con los ojos vendados.

Todos los sistemas controlados por ordenador dependen de las señales eléctricas para realizar su control. Lamentablemente, las señales eléctricas que se utilizan en el interior de un ordenador son demasiado imperceptibles como para poderlas utilizar directamente. Hasta un dispositivo tan pequeño como la bombilla de una linterna emplea considera-

Control remoto
Conectar un ordenador a otros aparatos proporciona toda clase de posibilidades de operación automática bajo el control de un programa. El ordenador puede responder a horas preprogramadas o reaccionar ante hechos tales como un descenso de la temperatura o la desconexión de una alarma antirrobo



Conexión de control
Las interfaces aptas para usos de control se venden ya hechas para ser usadas con muchos micros personales, sobre todo con el BBC y el ZX Spectrum. Estas unidades normalmente son unidades de conmutación basadas en relés. El ordenador puede conectar o desconectar un aparato determinado, y posteriormente enterarse de si el piloto del mismo se ha encendido o no

Ian McKinnell

blemente más energía que cualquiera de las partes que se hallan dentro de un ordenador; de modo que se necesita algún medio de traducir los mínimos voltajes del interior de un ordenador a voltajes más potentes (y algunas veces éstos pueden serlo tanto como los voltajes de la red eléctrica). Normalmente esto se cumple en una serie de etapas. La primera se realiza dentro del propio ordenador. Éste necesita enviar de alguna manera señales al mundo exterior. Esto se hace eligiendo una porción de la memoria del ordenador y reservándola exclusivamente para este fin. El microprocesador enviará mensajes a esta parte de la memoria igual que a cualquier otra, pero cuando éstos lleguen allí serán tratados de diferente forma. Por esta razón esta parte de la memoria se denomina *toma para el usuario* y la información almacenada en ella se puede leer electrónicamente desde fuera del micro.

Algunos micros poseen una toma para el usuario como estándar, otros disponen de ella como una opción. En sí misma, la toma para el usuario se puede utilizar para encender y apagar LED (diodos emisores de luz), pero la mayoría de los sistemas prácticos necesitan además otros componentes. Tal vez lo más útil es añadir unos pocos componentes electrónicos, más una pequeña fuente adicional de energía eléctrica, para permitir controlar los relés. Los *relés* son esencialmente interruptores que pueden encender y apagar corrientes eléctricas relativamente grandes y que, no obstante, se pueden controlar mediante pequeñas corrientes eléctricas. Proporcionan una de las mejores maneras de convertir los impulsos eléctricos que se emplean en un ordenador en corrientes útiles. Una de las primeras tareas de cualquiera que experimente con el control por ordenador es crear un sistema que permita usar relés, que pueden controlar variados mecanismos, desde motores eléctricos a trenes en miniatura y coches controlados por radio.

La mayoría de los relés que emplean los aficionados sólo pueden hacer frente a la clase de dispositivos que funcionan con pilas. Esto representa un amplísimo espectro para la mayoría de proyectos. Pocas personas se encontrarán con la necesidad de conmutar aparatos que se conecten a la red. Dado que la electricidad de la red es sumamente peligrosa, sólo se deben utilizar productos comerciales que hayan sido probados con todo rigor, y no hay muchos disponibles. Quien trabaja con voltajes pequeños puede elegir entre construir y comprar unidades de conmutación de relé ya hechas que se conectan directamente en un ordenador.

La conmutación de la red permite que la máquina controle calefactores, luces intensas y docenas de otros artículos domésticos. También permite que el ordenador actúe como un reloj para encender el televisor a la hora del programa favorito, o para encender y apagar las luces de la casa transcurridos determinados lapsos para disuadir a los ladrones de penetrar en la casa.

En la actualidad, el ordenador debe estar conectado directamente a las unidades que esté controlando, y esto es más bien una limitación. Varias empresas están desarrollando productos para superar este inconveniente. Estos productos funcionan permitiendo que los cables que distribuyen por toda la casa la electricidad de la red transporten también datos. El sistema tendrá un ordenador en una habitación enviándole señales a unidades *esclavas* dispuestas por toda la casa y conectadas a enchufes comunes. El ordenador puede enviar mensajes individuales a cada unidad indicándole que se conecte o se desconecte. Cualquier aparato doméstico se puede enchufar en la unidad y ser controlado a través del ordenador.

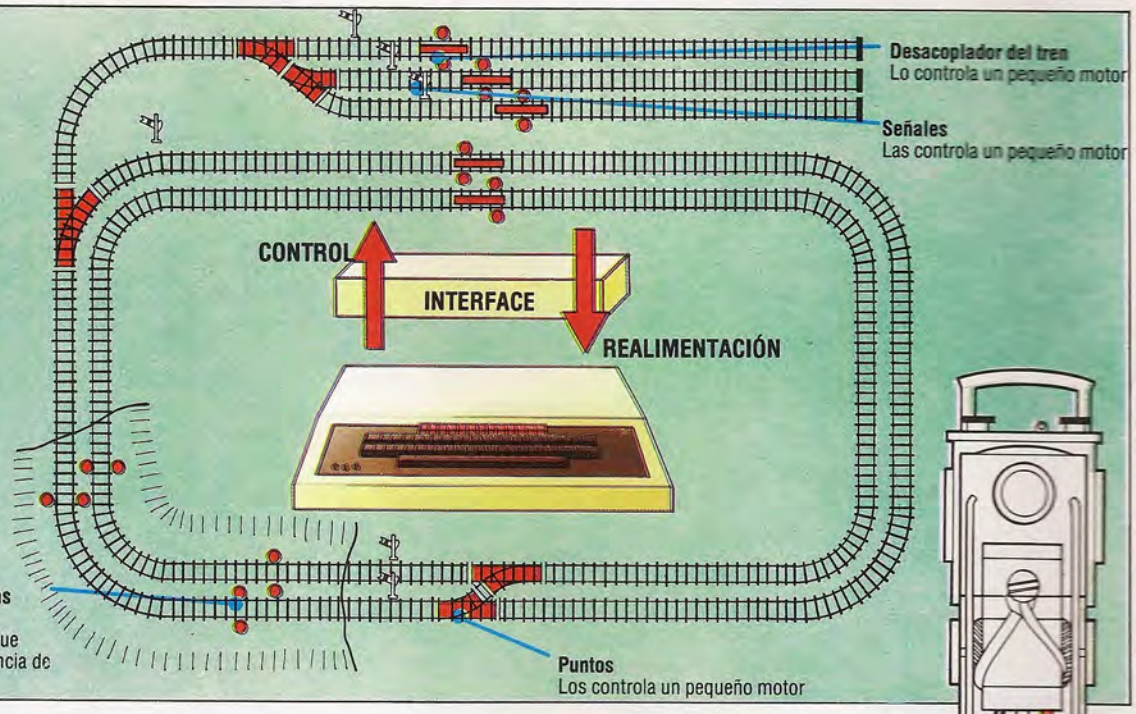
Ya hemos dicho que casi todos los sistemas controlados por ordenador necesitan algún tipo de realimentación para medir cómo está funcionando el sistema. Podría parecer que cuando un ordenador se está limitando a encender y apagar las luces no hay necesidad de realimentación. Todavía sería mucho mejor si el ordenador pudiera saber cuándo está oscuro afuera y encender las luces en respuesta a esa información. También sería útil que el ordenador pudiera detectar cuándo alguien entra en una habitación y encenderle la luz. Si un ordenador está controlando un calefactor, necesita realimentación para determinar las alteraciones de temperatura de la habitación y poder así mantener ésta dentro de unos límites prefijados.

Existen dos tipos de señales de realimentación. Algunas sólo pueden estar encendidas o apagadas, sin estados intermedios. Una señal de este tipo podría ser un interruptor que reconozca si una ventana está abierta o cerrada, o si ha sonado el timbre de la puerta o no. Las tomas para el usuario son capaces de leer este tipo de *señales on/off*.

Un tipo de realimentación más útil, pero ligeramente más complicado, proporciona una *señal ana-*

El cerebro del ferrocarril

Controlar cualquier tipo de aparato con un micro (desde un tren en miniatura hasta una casa entera) implica la misma técnica básica. Se establece un bucle en el cual el micro envía señales de control al dispositivo en cuestión a través de una interface. Éstas controlan servomotores, luces, etc. El dispositivo devuelve entonces información de realimentación mediante interruptores accionados por el peso del tren o células fotosensibles. Este bucle de realimentación permite que el ordenador maneje con precisión el aparato



Kevin Jones

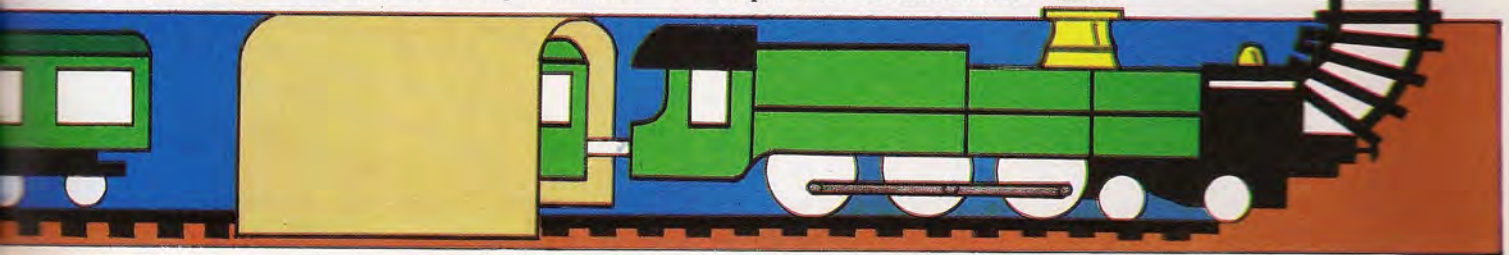
lógica. Una señal de esta clase puede ser de una escala de valores y, por tanto, se puede utilizar para medir el calor que hace, qué distancia se ha desplazado o ha girado un objeto, cuánto pesa algo o qué voltaje está dando una pila. El dispositivo que acepta este tipo de señal se denomina *convertidor de analógico a digital* (A/D): toma la escala variable de valores del equipo que se está controlando (la señal analógica) y la convierte a una forma digital que el ordenador puede comprender.

La disponibilidad de realimentación significa una gran diferencia en cuanto a lo que se puede hacer bajo control del ordenador. Si un motor está accionando una rueda, el ordenador podría calcular cuánto ha girado ésta en un cierto tiempo. Esto podría no funcionar, sin embargo, si se aplicara una carga sobre la rueda o si las pilas que activasen el motor se estuviesen agotando, porque entonces la rueda giraría más despacio. Un sensor óptico podría informar al ordenador cada vez que la rueda completara una revolución: de este modo, el ordenador podría controlar perfectamente su giro.

Algunos tipos de motores eléctricos hechos especialmente para usos de control llevan incorporada una cierta clase de realimentación. Ello significa que el ordenador envía una señal que les indica que se muevan hasta una posición determinada y el motor sigue funcionando hasta que llega a la misma. Existen dos tipos principales de esta clase de motores: *motores paso a paso* y *servomotores*. Un motor paso a paso puede girar continuamente, como un motor normal, o se puede detener en cualquier posición. No obstante, carece de potencia, de modo que sólo puede hacer frente a cargas pequeñas. Los servomotores son potentes pero sólo pueden girar un ángulo pequeño, por lo general de poco más de 90°. Esto a menudo se convierte en un movimiento de tipo empuje-tracción. Tanto los motores paso a paso como los servomotores necesitan unidades de control especiales para hacerlos funcionar con ordenadores, y éstas no están disponibles para muchas de las marcas de ordenadores personales menos usuales. Los servomotores se emplean en muchos brazos-robot. Es posible hallar en el mercado numerosos brazos-robot que se pueden conectar con ordenadores personales, pero son muy caros. Pueden construirse por un precio más reducido a partir de unos pocos servos.

La categoría final de dispositivos controlados por ordenador que vamos a considerar aquí requiere para su control un voltaje variable. Un ejemplo de ellos es un pequeño motor eléctrico que gire a distintas velocidades, según el voltaje que se le aplique. Un *convertidor de digital a analógico* (o D/A) es lo contrario de uno A/D: cambia las señales digitales que utiliza el ordenador a voltajes variables. Éste se podría emplear, por ejemplo, para producir sonido conectándolo a un altavoz.

Utilizar micros para controlar otros aparatos es como si uno escribiera su propio software. Se ha de combinar una idea con algún conocimiento técnico y mucho tiempo. A veces los resultados no están a la altura de los estándares comerciales, pero "hacerlo uno mismo" es, sin lugar a dudas, mucho más gratificante para el usuario que adquirir en el comercio productos de fabricación masiva.



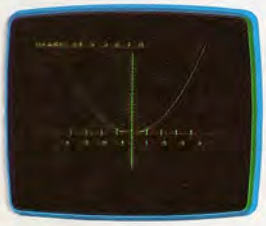
Diseño gráfico

En este capítulo veremos cómo con instrucciones sencillas se pueden convertir gráficos matemáticos en originales diseños

Para trazar el gráfico de una expresión matemática se toma una escala de valores para una de las variables y se calculan los valores correspondientes para la otra variable. Con un gráfico simple como $Y = X^2$, podríamos elaborar esta tabla:

X	–5	–4	–3	–2	–1	0	1	2	3	4	5
Y	25	16	9	4	1	0	1	4	9	16	25

Dibujando los puntos en una hoja de papel cuadriculado y uniéndolos luego mediante una curva continua obtenemos la familiar curva en forma de collar. La curva es el gráfico de $Y = X^2$. Otra forma de ver la curva es pensar que es el camino que sigue un punto que se desplaza a lo largo de $Y = X^2$. Al camino se lo suele llamar *lugar geométrico*, y combinando distintos lugares geométricos podemos producir sorprendentes patrones. Para hacer esto analicemos un lugar geométrico: la circunferencia. La mejor forma de describir una circunferencia mediante una fórmula es ligeramente más complicada que el método anterior. Tanto X como Y se definen en función de una tercera variable o parámetro. Haciendo variar éste a través del juego de valores dado, se pueden calcular pares de X e Y. La circunferencia la da esta ecuación:



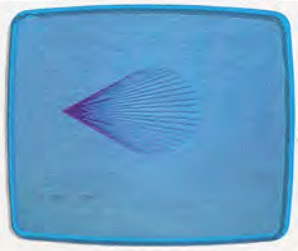
X = R SIN (I)
Y = R COS (I)

Si elaboramos un juego de valores para X e Y a medida que el ángulo I se gira en una circunferencia completa (de 0 a 360°), obtenemos el lugar geométrico de una circunferencia. El pequeño programa descrito a continuación para el Spectrum creará una circunferencia. (Véase "Complementos al BASIC" para las otras versiones de BASIC.)

```
10 REM Trazar circunferencia
20 LET xm=256: LET ym=176: LET xc=INT(xm/2): LET yc=INT
   (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 INK 2: PLOT xc+r*SIN (i),yc+r*COS (i)
70 NEXT i
```

Observe que cambiando el valor de STEP entre los puntos trazados se puede variar la definición de la circunferencia y alterar la velocidad a la cual se dibuja. La mayoría de las versiones de BASIC no son lo bastante rápidas como para dibujar circunferencias continuas a partir de muchos puntos individuales a una velocidad aceptable. Para superar este inconveniente, suele ser preferible utilizar un número de líneas rectas para unir los puntos de la circunferencia. Con un gran número de líneas rectas se puede conseguir un equilibrio razonable entre la velocidad y la uniformidad del dibujo. También puede emplearse esta fórmula para dibujar arcos y elipses. Para arcos, seleccione distintos valores de I. Para elipses, déle a R un valor diferente en la fórmula X al de la fórmula Y.

Nuestro primer patrón se crea dibujando una línea desde un punto fijo hasta todos los puntos que componen el lugar geométrico. Los siguientes programas posicionarán el punto en el centro de la circunferencia y a la izquierda de ella, respectivamente.



El siguiente paso consiste en utilizar un punto móvil en vez de uno fijo. Trazaremos dos lugares geométricos simultáneamente y dibujaremos líneas para conectar los puntos correspondientes en los dos caminos. El diseño más simple que esto creará es el de dos circunferencias anidadas unidas por numerosas líneas rectas.

```
10 REM flor
20 LET xm=256: LET ym=176: LET xc=INT (xm/2): LET
   yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 INK 2: PLOT xc+r*SIN (i), yc+r*COS (i): DRAW
   xc-(xc+r*SIN (i)), yc-(yc+r*COS (i))
70 NEXT i
```

```
10 REM Circunferencias anidadas
20 LET xm=256: LET ym=176: LET xc=INT (xm/2): LET
   yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 LET x=xc+(r+30)*SIN (i): LET y=yc+(r+30)*COS (i)
70 LET p=xc+r*SIN (i): LET q=yc+r*COS (i)
80 INK 2: PLOT x,y: DRAW p-x,q-y
90 NEXT i
```

```
10 REM Circunferencia y punto fijo
20 LET xm=256: LET ym=176; LET xc=INT (xm/2): LET
   yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 INK 2: PLOT xc+r*SIN (i),yc+r*COS (i): DRAW
   xc-100-(xc+r*SIN (i)),yc-(yc+r*COS (i))
70 NEXT i
```

El programa desarrollado ahora incorpora información suficiente para dibujar cientos de patrones. Lo que hay que hacer es cambiar un lugar geométrico o ambos. Una variación sencilla consiste en permutar SIN y COS en una de las fórmulas. También podrían utilizarse potencias de SIN y COS (multiplicándolas juntas) para producir otros efectos.

```
10 REM Circunferencias retorcidas
20 LET xm=256: LET ym=176: LET xc=INT (xm/2): LET
   yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 LET x=xc+ (r+30)*COS (i): LET y=yc+ (r+30)*SIN(i)
70 LET p=xc+r*SIN (i): LET q=yc+r*COS (i)
80 INK 2: PLOT x,y: DRAW p-x, q-y
90 NEXT i
```

Existen muchas variantes sobre estas ideas. Un libro de texto de matemáticas estándar le proporcionará las fórmulas para crear curvas alternativas. Sin embargo, probablemente encontrará que experimentar con su propia programación es un pasatiempo mucho más gratificante. Unas modificaciones al programa, pocas y sencillas, harán incluso que el ordenador haga la experimentación por usted. Aquí el programa final itera indefinidamente a través de un número de patrones generados al azar, aunque, naturalmente, no agota todas las posibilidades.

```
10 REM Circunferencia y SIN
20 LET xm=256: LET ym=176: LET xc=INT (xm/2): LET
   yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 LET x=xc+SIN (i)*80: LET y=yc+(r+30)*SIN (i)
70 LET p=xc+r*SIN (i): LET q=yc+r*COS (i)
80 INK 2: PLOT x,y: DRAW p-x, q-y
90 NEXT i
```

```
10 REM Circunferencia y SIN*COS
20 LET xm=256: LET ym=176; LET xc=INT (xm/2): LET
   yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 LET x=xc+r*SIN (i)*COS (i): LET y=yc+r*SIN (i)*COS (i)
70 LET p=xc+r*SIN (i): LET q=yc+r*COS (i)
80 INK 2: PLOT x,y: DRAW p-x,q-y
90 NEXT i
```

```
10 REM Patrones de circunferencias al azar
15 RANDOMIZE
20 LET xm=256: LET ym=176: LET xc=INT (xm/2): LET
   yc=INT (ym/2)
30 LET r=60: LET s=PI/20
35 CLS
40 LET c=INT (RND*4)+1: LET d=INT (RND*4)+1: LET
   e=INT (RND*4)+1: LET f=INT (RND*4)+1
50 FOR i=0 TO 2*PI STEP s
60 LET x=xc+r*SIN (i/c)*COS (i*d): LET y=yc+r*SIN (i/e)*
   COS (i*f)
70 LET p=xc+r*SIN (i): LET q=yc+r*COS (i)
80 PLOT x,y: DRAW p-x,q-y
90 NEXT i
100 IF INKEY$="" THEN GO TO 100
110 GO TO 35
```

Complementos al BASIC

Los programas listados aquí funcionarán en un Spectrum de 16 Kbytes y de 48 Kbytes. Sin embargo, convertir estas ideas y utilizarlas en otras máquinas es muy sencillo. Para ello, su micro necesita gráficos de alta resolución (preferentemente, al menos 256 × 176), un BASIC con coma flotante con las funciones SIN y COS, y una instrucción para trazar puntos individuales y dibujar líneas rectas.

Los primeros ajustes necesarios son XM e YM, los valores X e Y máximos que se pueden trazar en su máquina. Según las funciones que utilice, es posible que deba modificar los valores de otras variables del programa como R y S. Seguidamente debe asegurarse de que su micro esté en la modalidad de gráficos apropiada y seleccionar un color para el trazado. Por último, necesita una orden para dibujar líneas entre las coordenadas dadas en X e Y y en P y Q. En el Spectrum esto se ha de hacer con PLOT seguida de DRAW. La orden DRAW es complicada porque en el Spectrum siempre se relaciona con el primer punto dibujado, mientras que en este caso necesitamos dibujar una posición absoluta determinada. La mayoría de los micros posee una función para trazado de líneas absolutas y, por consiguiente, esta etapa es mucho más sencilla.

BBC MICRO Todas las modalidades del BBC utilizan un cuadriculado de la pantalla de 1280×1024 puntos para el trazado, en el que la instrucción MODE 0 produce resultados espectaculares. Utilice GCOL para seleccionar el color del trazado y MOVE y DRAW para dibujar las líneas.

DRAGON 32/64 La PMODE 4 del Dragon proporciona un cuadriculado de 256×192 apto para estos programas. Utilice SCREEN 1, 0 o SCREEN 1, 1 para seleccionar un fondo verde o bien marrón. La instrucción LINE se puede utilizar (LINE (X,Y)-(P,Q),PSET) para dibujar las líneas.

COMMODORE 64/VIC-20 Estas máquinas poseen gráficos de alta resolución adecuados, pero no las instrucciones apropiadas. Para ejecutar estos programas es necesario o bien suministrar las instrucciones para cada punto y línea o bien utilizar un cartucho de ampliación de BASIC, como el BASIC de Simon.

CAMPUTERS LYNX El Lynx es muy adecuado para esta clase de trabajo, puesto que posee una completa visualización para gráficos de 256×248 en ocho colores. Al igual que el Spectrum, no es necesaria ninguna instrucción de cambio de modalidad. Utilice INK para seleccionar el color del diseño y MOVE y DRAW para trazar las líneas.

ORIC 1/ATMOS HIRES activa la pantalla para gráficos de 240×200 del Oric. Las líneas se pueden dibujar utilizando CURSET para establecer el punto de comienzo (X,Y) y luego DRAW para trazar la línea. En el Oric, DRAW es una instrucción relativa, de modo que la orden DRAW ha de tener la forma DRAW p-x,q-y para funcionar.

Ideas de diseño

1) Volviendo al bucle simple para dibujar una circunferencia, hemos descrito cómo crear arcos y elipses mediante el mismo programa. Ahora vea si puede hallar una forma de dibujar espirales.

2) Intente utilizar otras funciones como SQR y TAN para generar lugares geométricos. Sepa que se deben emplear con cuidado porque tienden a generar números complicados. No obstante, debería ser capaz de producir resultados interesantes.

3) Realice versiones animadas de los programas. Utilizando matrices para registrar las últimas cinco líneas dibujadas, debería ser capaz de dibujar un grupo de cinco líneas persiguiéndose las unas a las otras alrededor de dos lugares geométricos.

4) ¿Por qué no intenta crear patrones basados en tres lugares geométricos? Utilice dos muy simples (p. ej., una circunferencia y una línea recta) para no crear dibujos demasiado confusos.

Continúa el serial

Ahora nos corresponde analizar cómo se crean, se accede a ellos y se actualizan los archivos secuenciales

Un archivo secuencial es un bloque de datos sólido contenido en un disco o una cinta y, como tal, existen limitaciones en cuanto a cómo puede ser accedido y actualizado. Para recuperar un ítem cualquiera, uno primero debe leer todos los datos precedentes. Para actualizar el archivo se suele hacer una copia del mismo hasta el punto que se necesita modificar, luego añadir los cambios al archivo nuevo y proseguir con la copia del archivo original inmediatamente después de las modificaciones.

Es importante comprender que la información ha de estar organizada de una forma adecuada dentro del archivo. La elección acerca del tipo de organización está en manos del programador y dependerá en cada caso de la aplicación. Si se trata de un archivo que contiene texto en castellano, es probable que sea suficiente una secuencia de códigos ASCII seguida de una marca de final de archivo. Sin embargo, si el archivo ha de contener una base de datos, como puede ser un catálogo de libros, entonces es necesario organizar la información de acuerdo con la estructura del catálogo. La forma normal de hacerlo consiste en dividir el archivo en *registros* y *campos*. Cada libro posee su propio registro (entrada) en el archivo y dentro de cada registro hay un número de campos (el título del libro, su autor, su editor, etc). En un archivo secuencial, estas divisiones se deben señalar utilizando caracteres especiales colocados entre los datos.

Esto se suele realizar utilizando un carácter de retorno del carro (código ASCII 13) para que actúe como delimitador entre los campos y registros. Dado que el archivo tendrá el mismo número de campos en cada registro, para el programa será fácil llevar la cuenta de dónde termina un registro y empieza otro.

Una vez que se ha creado un archivo secuencial, uno necesita ser capaz de acceder a él y actualizarlo. Las operaciones básicas sobre archivos son: recuperación de registros, adición de registros, eliminación de registros y modificación (edición) de registros. Los diagramas ilustran las diversas maneras de realizar estas operaciones con archivos secuenciales. Dado que usted sólo puede leer un archivo secuencial por orden y no puede cambiar libremente los datos que contiene, estas operaciones trabajan leyendo a través del archivo, creando una nueva copia a medida que se producen. Toda información que se va a cambiar se escribe en el nuevo archivo a medida que se crea. Por último, el nuevo archivo se convierte en el archivo en curso o actual y el antiguo es eliminado o bien se conserva como una copia de seguridad.

Estas sencillas técnicas constituyen la base de todas las rutinas de tratamiento de archivos secuenciales. No obstante, implican un supuesto fundamental acerca de las capacidades del sistema operativo: que éste pueda tener abiertos dos archivos al mismo tiempo con el objeto de leer en uno y escribir en otro simultáneamente. Esto no es posible en todos los sistemas de disco, y en los micros basados en cassette sólo es posible en aquellos que tengan acopladas dos grabadoras de cassette. Tanto la gama Grundy Newbrain como la Commodore PET disponen de interfaces para cassette gemelas por este motivo. Las máquinas con unidades de cassette individuales están limitadas a archivos suficientemente pequeños como para ser leídos por completo sobre la memoria y ser procesados allí.

Estos procedimientos de tratamiento de archivos también poseen un interesante efecto colateral. Después de que se haya introducido cualquier modificación en el archivo (ya sean adiciones, eliminaciones o modificaciones) se dispone de dos copias del archivo: una antigua, que es el archivo antes de que se lo actualizase, y una copia nueva con los cambios realizados. Es una práctica estándar en la gestión empresarial conservar ambos archivos, de modo que si algo le sucediera al nuevo siempre habría una copia que sólo estaría anticuada en un conjunto de cambios (o una generación). De hecho, se suelen conservar tres generaciones de cualquier archivo: al nuevo se lo denomina archivo *hijo* y al precedente se lo conserva como archivo *padre*. El archivo que se utilizó como base para el archivo padre se conoce como archivo *abuelo*.

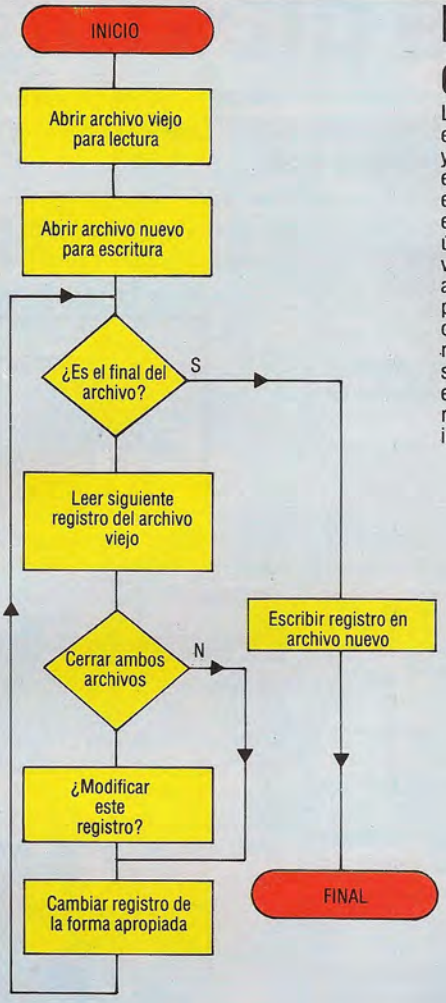
Estas técnicas pueden trabajar con archivos que sean demasiado grandes como para caber en su totalidad en la memoria del ordenador, porque sólo una parte del archivo se está procesando en un momento dado. Sin embargo, con los archivos pequeños se puede obtener un rendimiento mucho mayor leyendo todo el archivo en matrices en la memoria y procesándolo allí. Todas las operaciones sobre archivos se pueden llevar a cabo a gran velocidad en la memoria antes de volver a escribir el nuevo archivo ya procesado en disco o en cinta.

Este enfoque tiene un peligro fundamental: los cambios introducidos en el archivo sólo adquieren carácter permanente cuando la información se vuelve a escribir en cassette o en disco y, por consiguiente, los datos se podrían perder si el programa funcionase mal o si se apagara el ordenador durante la ejecución. Si usted utiliza programas que trabajan así, asegúrese de que se ha grabado una copia del archivo en curso antes de que el programa termine.

Con un poco de experiencia en tratamiento de archivos secuenciales comprobará que las técnicas pertinentes, a pesar de ser engorrosas, se basan fundamentalmente en el sentido común. En muchos sistemas pequeños, los archivos secuenciales son la única estructura de archivos proporcionada. Cuando analicemos los archivos de acceso directo, o aleatorio, descubriremos técnicas que complementan a los archivos secuenciales, proporcionando tanto un acceso como una actualización sencillos y rápidos.

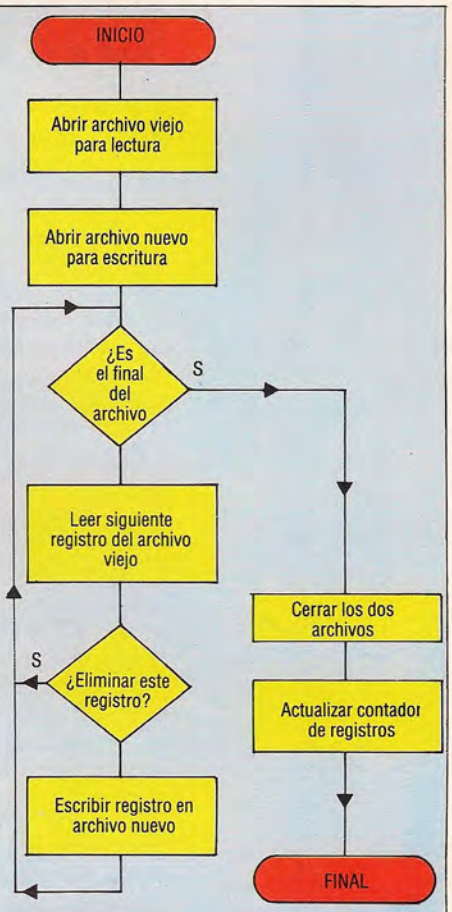
Modificación de un registro

Modificar un registro (alterar la información que contiene) se consigue mediante una combinación de estos métodos. En primer lugar se deben copiar todos los registros precedentes en un archivo nuevo. Cuando se ha llegado al registro a modificar y se lo ha leído en RAM, éste puede ser modificado por el programa. El registro corregido se escribe en el archivo nuevo y todos los registros subsiguientes del archivo viejo se copian en el archivo nuevo a continuación del que se modifica. En una pasada a través del archivo se puede corregir cualquier número de registros



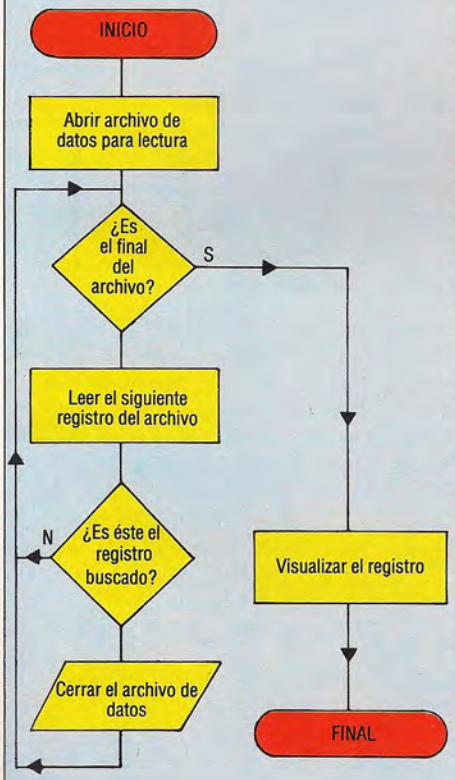
Eliminación de registros

Los registros se pueden eliminar de un archivo leyendo y copiando hasta el registro a eliminar. Este registro se lee entonces, pero no se escribe en el archivo nuevo. Por último, el resto del archivo viejo se lee y se copia en el archivo nuevo. En una sola pasada se puede eliminar cualquier número de registros. Al igual que cuando se añaden registros, es esencial que el contador de registros se actualice inmediatamente



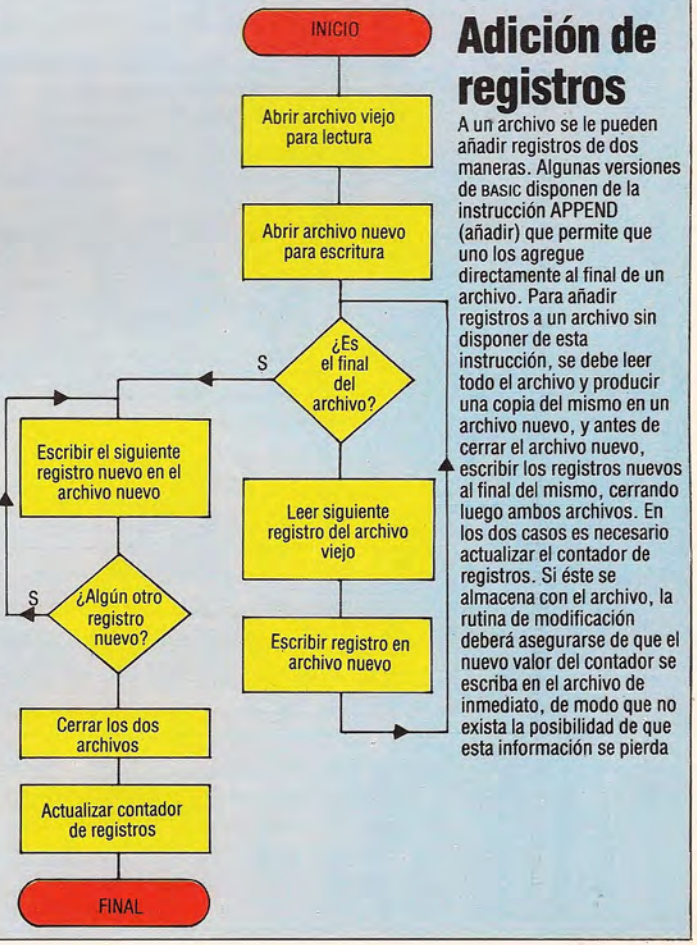
Recuperación de registros

Los archivos secuenciales no son muy idóneos para la clase de aplicación en la que uno coge registros antiguos del archivo. Cada vez que se busca uno de ellos, se ha de volver a leer el archivo desde el principio, lo que consume mucho tiempo. Si usted está buscando un registro determinado en una lista de nombres, entonces su programa simplemente pasará a través de un bucle, examinando cada registro y siguiendo adelante si no existe correspondencia entre el leído y el buscado. Si usted necesita un cierto número de registros, éstos se pueden leer uno después del otro pero sólo por el orden en que están grabados en el archivo. Por este motivo, a menudo los archivos secuenciales se clasifican por algún orden (p. ej., alfabéticamente) antes de almacenarlos



Adición de registros

A un archivo se le pueden añadir registros de dos maneras. Algunas versiones de BASIC disponen de la instrucción APPEND (añadir) que permite que uno los agregue directamente al final de un archivo. Para añadir registros a un archivo sin disponer de esta instrucción, se debe leer todo el archivo y producir una copia del mismo en un archivo nuevo, y antes de cerrar el archivo nuevo, escribir los registros nuevos al final del mismo, cerrando luego ambos archivos. En los dos casos es necesario actualizar el contador de registros. Si éste se almacena con el archivo, la rutina de modificación deberá asegurarse de que el nuevo valor del contador se escriba en el archivo de inmediato, de modo que no exista la posibilidad de que esta información se pierda

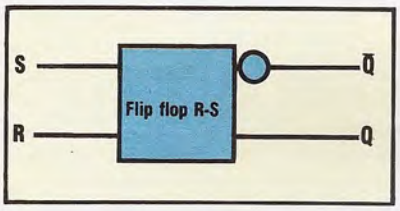


Circuitos biestables

Esta vez examinaremos detalladamente el diseño y la función de un circuito biestable: el flip-flop R-S

Todos los circuitos que hemos analizado hasta ahora en el curso de lógica producen salidas determinadas al recibir ciertas señales de entrada. Los circuitos secuenciales, por otra parte, son capaces de producir una señal de salida uniforme en respuesta a un único impulso de entrada. Estudiemos, pues, con detenimiento el diseño y la función de un circuito de este tipo: el flip-flop R-S.

Existen diversos tipos de flip-flop, aunque todos ellos funcionan siempre en base a los mismos principios. Tal como se puede apreciar en la figura que aparece a continuación, el flip-flop R-S posee dos líneas de entrada y dos de salida.

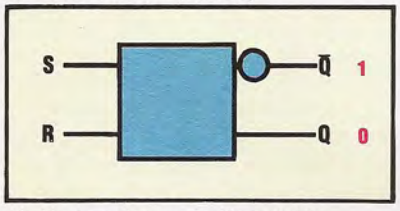


El circuito está diseñado de modo que las líneas de salida, Q y $\overline{Q}$, siempre sean opuestas. Es decir:

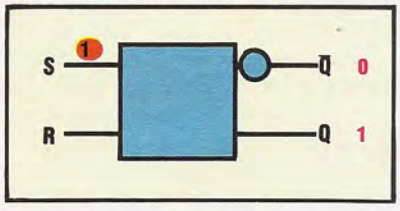
si Q=1 entonces $\overline{Q}$=0 (el estado SET)
si Q=0 entonces $\overline{Q}$=1 (el estado RESET)

Suponiendo que inicialmente el flip-flop está en estado RESET, entonces un impulso en la línea S hace que el circuito "salte" al estado SET.

1) Estado inicial (RESET)

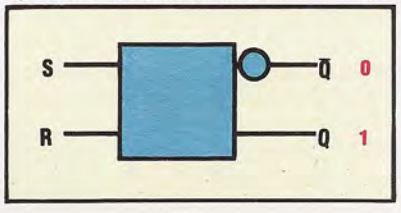


2) Un impulso en la línea SET



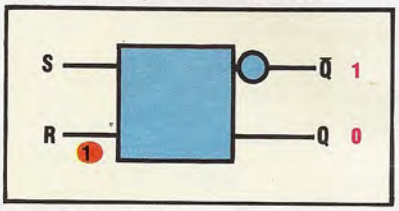
Cuando cesa el impulso de entrada en la línea S, el circuito permanece en el estado SET.

3) El circuito permanece en el estado SET



Un impulso enviado por la línea R hace saltar al circuito otra vez a su estado RESET original.

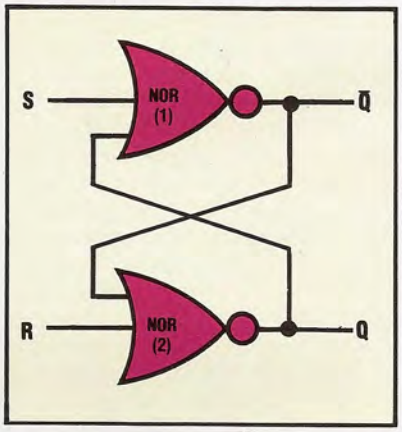
4) Un impulso en la línea RESET



Ahora, habiendo descrito la función de un biestable R-S, veamos con más detalle los elementos lógicos del circuito.

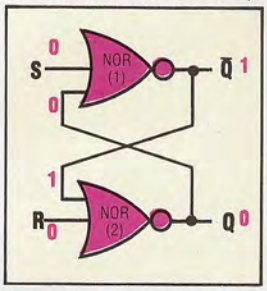
Circuito flip-flop R-S

Un flip-flop R-S se puede construir utilizando varias técnicas, como enlazar entre sí dos puertas NAND o, como en el ejemplo que proporcionamos aquí, enlazando entre sí dos puertas NOR de modo tal que la salida de una de éstas constituya una de las entradas de la otra. Es este "bucle hacia atrás" de las señales lógicas lo que le proporciona al flip-flop su capacidad de "memoria".

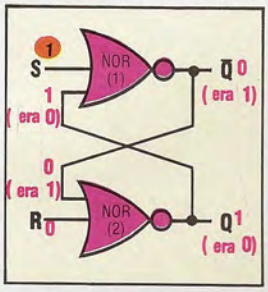


Vamos a investigar las funciones SET y RESET del flip-flop y ver cómo se obtienen mediante esta combinación de puertas NOR. Si suponemos que inicialmente el flip-flop está en estado RESET y que no hay impulsos de entrada, entonces el circuito se mantendrá en estado estable. (Recuerde que una puerta NOR sólo da una salida de uno si ambas entradas son cero.) Un impulso por la línea S modificará esta situación estable haciendo que la salida "no Q" ($\overline{Q}$) cambie a cero. Esto afecta a la entrada de la segunda puerta NOR (2), que proviene de la salida de la primera, haciendo que la salida de esa puerta, Q, cambie a uno. Esto significa que si aún está presente el impulso en la primera puerta NOR (1), entonces ambas entradas a la puerta NOR (1) serán uno. Por consiguiente, la salida de la puerta NOR (1) seguirá siendo cero y el circuito habrá cambiado al otro estado posible: SET.

1) Estado inicial (RESET)

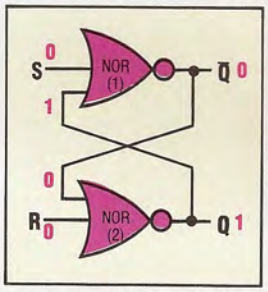


2) Un impulso en la línea SET

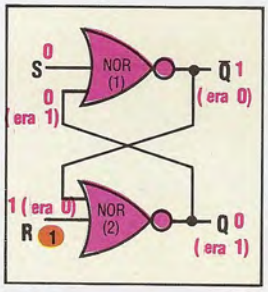


Aun cuando se elimine el impulso de la línea S, el circuito continuará estabilizado en el nuevo estado. Cuando se envía un impulso por la línea R, el circuito cambia otra vez, volviendo al estado RESET, por medio de un proceso similar al que ya hemos descrito.

3) El circuito permanece en estado SET

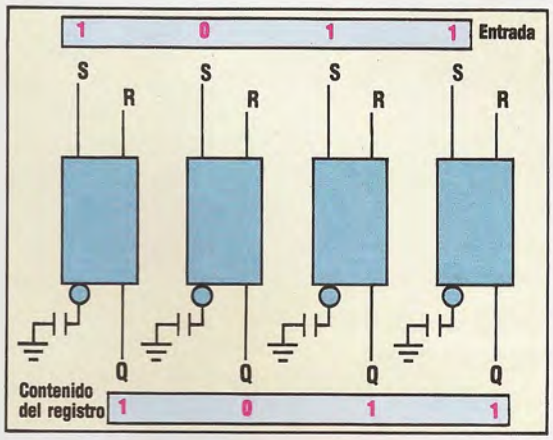


4) Un impulso en la línea RESET



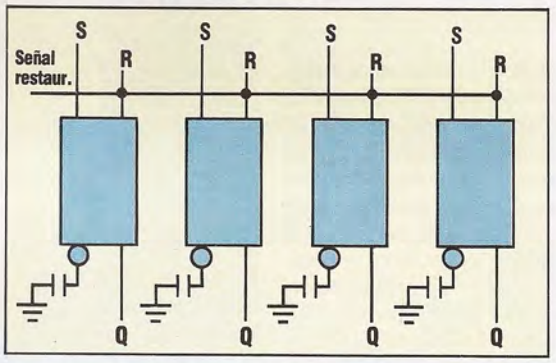
Registros

Un microprocesador se compone en gran medida de una serie de registros, como el acumulador, los registros de instrucciones y el registro índice. La mayoría de los registros pueden mantener palabras de ocho bits (es decir, números binarios comprendidos entre 0 y 255). Como estos registros tienen que aceptar y almacenar información binaria, no es sorprendente que se compongan de una serie de ocho flip-flops. Para hacer las cosas más sencillas, vamos a analizar cómo acepta y almacena números un registro de cuatro bits. Si deseamos almacenar el número binario 1011, todo cuanto se requiere es alimentar las líneas con este patrón binario.



Observe que en esta disposición la salida "no Q" no se utiliza. Como el patrón binario de entrada se aplica a las líneas S de los flip-flops, las líneas Q producen la salida correspondiente. Si deseáramos sustituir el primer número almacenado en el registro por otro, por ejemplo el 0110, podríamos pensar que todo lo que hace falta es presentar a las líneas S del flip-flop este nuevo patrón binario. De ser así, el número resultante almacenado en el registro sería 1111. Los unos de las posiciones más exteriores serían sobrantes del número anterior.

La solución para este problema consiste en restaurar cada flip-flop antes de almacenar otro número. Puesto que es necesario restaurar todos los flip-flops al mismo tiempo, es conveniente conectarlos entre sí permitiendo que la restauración del registro sea activada por una sola señal.



Liz Dixon

En el próximo capítulo analizaremos otros circuitos secuenciales, incluyendo el flip-flop tipo D y el flip-flop J-K.

Ejercicio 7

1) ¿Por qué al flip-flop también se le denomina "biestable?

2) Inicialmente, cuando se conecta un ordenador, un flip-flop está en el siguiente estado:

$Q = 0, \overline{Q} = 0, S = 0, R = 0$

a) ¿Es éste un estado estable?
b) Si no lo es, ¿a qué estado cambiará el flip-flop?
c) ¿Puede el flip-flop cambiar a un estado distinto al estado dado en su última respuesta? (Una pista: pruebe empezando con la otra puerta.)
d) ¿Qué proceso es necesario, cuando se conecta un ordenador, para asegurar que todos los registros estén en un estado predecible?

Disparos de luz

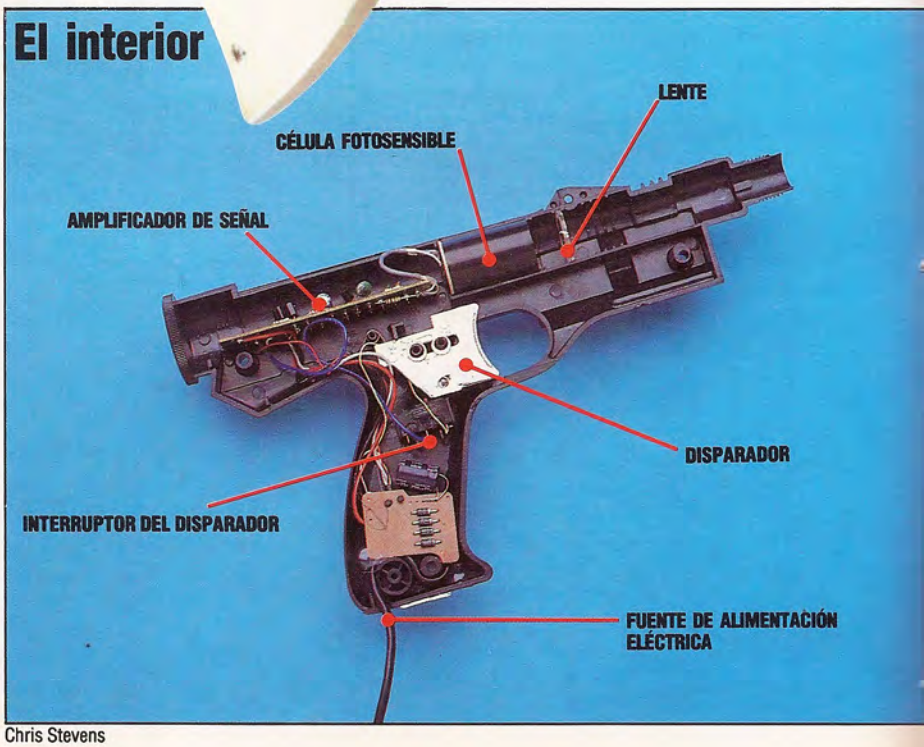
El SLR está diseñado para dar un realismo adicional a los juegos de tiro al blanco para ordenadores personales

El Stack Light Rifle es un novedoso dispositivo que combina la apariencia de un fusil con un sistema óptico estilo cámara y que se basa en la tecnología del lápiz óptico (véanse pp. 156-157).

El principal componente del sistema Stack Light Rifle es la pistola electrónica receptora, que está conectada al ordenador mediante una generosa porción de cable. En un extremo del ordenador, según la versión de que se trate, hay un "enchufe" para el conector apropiado. En la versión ZX Spectrum el conector contiene dos chips y un par de componentes simples para conectar en interface la electrónica principal del interior del fusil al ordenador. Para hacer que la pistola sea más precisa (y para convertirla en rifle) se proporciona con un apoyo para el hombro que se engancha y se asegura a la culata de la pistola, además de un cañón y una mira telescópica falsa.

La electrónica del interior de la pistola se compone de un detector de luz o fotodiodo y un pequeño amplificador y buffer. La luz que penetra por el cañón se enfoca mediante una pequeña lente plástica en el fotodiodo y el dispositivo es suficientemente sensible como para detectar los cambios en la intensidad de la imagen. Una vez intensificada por el amplificador, la señal analógica se recorta para proporcionar un impulso digital, y luego se le envía al ordenador a través del interruptor. La posición de la pantalla que se está explorando en ese momento es la posición hacia la cual está apuntando el fusil. A medida que el ordenador recibe el impulso proveniente del Light Rifle, compara el valor de sus registros de exploración con la posición en pantalla del blanco y, si se halla una pareja, la jugada ha obtenido un golpe directo.

Existen variantes del Light Rifle para el ZX Spectrum, el Commodore Vic-20 y el Commodore 64 y todas ellas realizan la misma función. Stack proporciona tres juegos en cassette con el Light Rifle, y prácticamente son los únicos de que disponen. A pesar de que varias casas de software independientes producen juegos que parecerían ser eminentemente aptos para este tipo de control del usuario, muy pocas han producido o convertido programas para funcionar con él; Micromania constituye una excepción. Posiblemente aún más perjudicial para el potencial de ventas del Light Rifle es el hecho de que Stack no proporciona ninguna rutina de apoyo para permitir que los usuarios escriban sus programas. Esto, junto con la falta de detalles acerca de cómo funciona el fusil, descarta al SLR



Chris Stevens

en cuanto una posible buena alternativa a la palanca de mando.

El Light Rifle se basa en el mismo principio de operación del lápiz óptico, pero es mucho más grande y está diseñado para ser sostenido a una distancia de aproximadamente tres metros del aparato de televisión y no en contacto con la pantalla. Para ayudar a filtrar la luz ambiental, el Light Rifle dispone de un largo tubo oscuro (el cañón) y una lente. Ambos se combinan para proporcionar un grado de precisión razonable (si no perfecto) y permitir que el usuario "dispare" cómodamente desde un sofá. Los juegos que se suministran son ejemplos bastante pobres de lo que sería posible hacer; tanto el uso de gráficos como su potencial lúdico son apenas explotados.

Uno de los principales problemas de la programación para lápices ópticos, o para versiones gigantes como el Light Rifle, es que el programa ha de ser escrito de forma que sea muy eficiente. En todos los ejemplos suministrados por Stack, los juegos se interrumpen momentáneamente al apretar el gatillo. Lo ideal sería explorar continuamente la pantalla, como se realiza por lo general con un lápiz óptico, pero esto reduciría demasiado la velocidad del juego. De modo que cuando se aprieta el gatillo del Light Rifle, el software debe congelar la acción y determinar si el blanco de la pantalla está alineado con la posición de la pistola. Una vez que el software ha determinado si el jugador ha dado o no en el blanco, el juego puede continuar. En teoría, cuando se aprieta el gatillo, la cantidad de código necesario para determinar la posición en pantalla del objeto detectado por la pistola debería ser pequeña, pero ello no siempre es así.

En un ordenador como el BBC, para el cual aún no hay ninguna versión del Light Rifle, el disponer de la facilidad para lápiz óptico en el chip de video simplificaría en gran medida la tarea del software. El Commodore 64 ofrece esta clase de sistema, pero el ZX Spectrum, en el cual se probó al Light Rifle, carece de esta facilidad, y esto se refleja en el tiempo consumido en calcular la posición del rifle cuando se aprieta el gatillo.

Disparo en la oscuridad

La célula fotosensible del Light Rifle detecta el punto en movimiento de la exploración de barrido a medida que ésta refresca la imagen de televisión. El software controla continuamente el contador del raster en el ordenador, de modo que siempre conoce la posición del punto de barrido en la pantalla; cuando el fusil de luz señala que ha detectado ese punto, el software puede convertir el valor del contador del punto de barrido en las coordenadas X,Y adonde apunta el rifle

Blanco móvil

High noon (A pleno día) es el mejor de los tres programas de demostración. La animación es buena y el tirador dispara de forma verosímil. Tanto *Grouse shoot* (Caza de patos) como *Shooting gallery* (Salón de tiro) ofrecen un único blanco móvil, que ha de ser detectado antes de que salga de los límites. La animación es inestable y al apretar el gatillo se produce una perceptible interrupción en la acción en pantalla

Ian McKinnell

Crear caracteres

En este capítulo introduciremos las técnicas de los gráficos definidos por el usuario del Commodore 64 y continuaremos con el juego del "cazador de submarinos"

El juego de caracteres gráficos del Commodore 64 es de gran amplitud, pero, por lo general, es necesario crear algunos caracteres especiales o incluso volver a definir todo el juego de caracteres.

El proceso de crear sus propios caracteres en el Commodore 64 no es directo: en el BASIC Commodore no hay instrucciones para fines especiales, de modo que toda la operación se debe llevar a cabo utilizando PEEK o POKE para acceder y cambiar el contenido de la memoria.

El juego de caracteres del Commodore 64 se compone de un bloque de ROM que comienza en la posición de memoria 53248. Cada carácter aparece en la pantalla como un patrón de puntos en una matriz de puntos de ocho por ocho: describir este patrón de 64 puntos requiere 64 bits u ocho bytes. Los ocho bytes desde la posición 53248 a la 53255 describen el carácter "@", el primer carácter del juego; posee un código de pantalla 0, lo que significa que si usted coloca (POKE) el valor cero en uno de los bytes de la RAM de video, aparecerá este carácter en la pantalla.

Los ocho bytes siguientes, del 53256 al 53263, representan el carácter "A" (código de pantalla 1), y así sucesivamente.

Nosotros no podemos cambiar estas definiciones de la matriz de puntos en ROM, de modo que debemos copiar algunas de ellas o todas en RAM y hacer los cambios allí. Entonces podemos hacer que el Commodore utilice nuestro juego de caracteres de RAM para escribir en la pantalla, en vez de emplear sus propias definiciones en ROM.

El juego de caracteres en ROM comparte su espacio de direcciones en la memoria con dispositivos de entrada-salida tales como grabadoras de cassette y unidades de disco. La CPU 6510A suele tratar este espacio de memoria como una zona para entrada-salida, pero se la puede programar para que la considere como la situación del juego de caracteres. Esto podría parecer extraño, pero la CPU normalmente no hace el trabajo de acceder a las definiciones de caracteres desde ROM y enviarlas a la pantalla. Esta tarea se delega a un chip subsidiario bajo el control de la CPU. El contenido de la posición 1 determina el status de las operaciones de E/S, y el bit 2 de esta posición actúa como un indicador de la forma en que la CPU considera el juego de caracteres en ROM. Si este bit es puesto a 0, entonces la CPU asume que los dispositivos de E/S ocupan este espacio.

Los otros bits de la posición 1 poseen funciones especiales similares en el control del sistema, de modo que debemos ser cuidadosos y no alterar ninguno de ellos mientras cambiamos el valor del bit 2. La mejor forma de hacerlo es empleando los operadores lógicos AND y OR.

Supongamos que el contenido de la posición 1 es:

Bit	7	6	5	4	3	2	1	0
	0	1	1	0	1	1	1	1

Nosotros deseamos cambiar el bit 2 a cero. Una forma de hacerlo sería calcular el valor decimal de 01101011 y colocarlo (POKE) en la posición 1, pero esto sólo funciona si sabemos que el contenido previo de la posición 1 era 01101111. Una forma mejor de modificar el bit 2 consiste en utilizar AND y PEEK. La siguiente orden coge (PEEK) el contenido de la posición 1, le aplica AND con el valor 251, y coloca (POKE) el resultado de nuevo en la posición 1:

```
POKE 1, PEEK (1) AND 251
```

El efecto de esta instrucción se ilustra así:

Bit	7	6	5	4	3	2	1	0	
	0	1	1	0	1	1	1	1	= contenido inicial
AND	1	1	1	1	1	0	1	1	= 251 binario
	0	1	1	0	1	0	1	1	= resultado de aplicar el operador AND a cada par de bits

Independientemente del valor original del bit 2, al operarlo mediante AND con cero siempre producirá un resultado cero; operando mediante AND todos los otros bits de la posición con uno simplemente mantiene su valor original. El número binario 11111011 (decimal 251) se denomina *máscara* u *overlay*, y aquí lo estamos empleando como una "máscara AND".

Para poner el bit 2 a uno sin afectar a ninguno de los otros bits, utilizamos la siguiente orden:

```
POKE 1, PEEK (1) OR 4
```

Bit	7	6	5	4	3	2	1	0	
	0	1	1	0	1	0	1	1	= contenido inicial
OR	0	0	0	0	0	1	0	0	= 4 binario
	0	1	1	0	1	1	1	1	= resultado de aplicar el operador OR a cada par de bits

Esto asegura que el BASIC no machacará nuestro juego de caracteres. Cuando la copia está completa, la CPU se puede restaurar para direccionar los dispositivos de E/S restaurando la interrupción.

La pieza final de este rompecabezas es obligar al chip de manipulación de pantalla a que utilice nuestro juego de caracteres en vez del juego de la ROM del sistema. Los bits del 0 al 3 de la posición 53272 señalan la dirección de inicio del juego de caracteres, y la tabla siguiente muestra cómo el Commodore 64 interpreta que los valores de estos bits apuntan a direcciones determinadas:

valor decimal de bits 0 al 3	bits 3, 2, 1, 0	posición a que apuntan
0	0000	0
2	0010	2048
4	0100	4096
6	0110	6144
8	1000	8192
10	1010	10240
12	1100	12288
14	1110	14336

En este registro el valor del bit 0 es irrelevante, mientras que los bits del 4 al 7 controlan otras funciones y no deben ser alterados. Con este fin, empleamos 11110000 (decimal 240) como una máscara AND, y 00001110 (decimal 14) como una máscara OR para hacer que el registro apunte a 14336, la posición inicial de nuestro juego de caracteres:

```
POKE 53272, (PEEK (53272) AND 240) OR 14
```

Ahora, utilizando un bucle FOR...NEXT podemos realizar la copia en curso.

Mientras un programa está copiando en RAM el juego de caracteres de ROM, la CPU no puede tratar interrupciones de dispositivos de E/S. El teclado, por ejemplo, interrumpe a la CPU cada sesentavo de segundo, haciéndola explorar el teclado en busca de alguna pulsación. Estas interrupciones son provocadas por el reloj del sistema. Si un dispositivo de E/S interrumpiera a la CPU mientras el juego de caracteres está ocupando el espacio de ROM para E/S (como en este caso durante el copiado), entonces probablemente el sistema se colgaría y la máquina sólo podría ser reinicializada desconectándola y volviéndola a conectar a la corriente. Por suerte, podemos bloquear el mecanismo de interrupción poniendo a cero el bit 0 de la posición 56334; los otros bits de esta posición no se deben modificar, de modo que hay que utilizar la siguiente instrucción lógica POKE:

```
POKE 56334, PEEK (56334) AND 254
```

Una vez bloqueadas las interrupciones y obligada la CPU a mirar el juego de caracteres en ROM, entonces se puede iniciar la copia.

Nosotros queremos copiar, pongamos por caso, los 64 caracteres desde "@" a "?" (códigos de pantalla de 0 a 63), de modo que debemos copiar las 512 posiciones (8×64=512) comenzando desde 53248 en un bloque de RAM adecuado. Se pueden utilizar varias zonas, y aquí nuestra elección es un bloque que comienza en la posición 14336. Ésta normalmente estaría en el área para programas BASIC, pero nosotros la protegemos bajando el puntero a la parte superior de la memoria que hay en la posición 56, de este modo:

```
POKE 56,32
```

Cada carácter del juego del Commodore está diseñado sobre una matriz de puntos de ocho por ocho. Cada fila de la matriz se interpreta como un número binario (los puntos que están iluminados se interpretan como uno, los que no aparecen en la pantalla se interpretan como cero) y requiere un byte de memoria, y las ocho filas del carácter completo requieren ocho posiciones consecutivas en la memoria. La posición inicial de los bytes que describen un carácter se puede calcular a partir de la dirección inicial del bloque entero y el código de pantalla del carácter en cuestión, de este modo:

Comienzo de carácter = 14336 +8× (código de pantalla)

Una vez que se conocen las posiciones de los bytes que definen a un carácter, podemos colocar (POKE) nuevos valores en estos bytes cambiando, por consiguiente, los patrones de puntos que aparecen en la pantalla cuando se visualiza el carácter. Siempre que el juego de caracteres está activado, pulsar la tecla correspondiente hará que el nuevo carácter aparezca en la pantalla. En el programa de demostración, redefinimos los caracteres [,], £ y ↑ (códigos 27-30) como partes de una figura, y conseguimos su animación imprimiendo y sobreimprimiendo distintas versiones de la figura.

```
130 :
140 REM **** COPIAR JUEGO CARACT. ROM ****
150 POKE56,32
    :REM BAJAR TOPE SUPERIOR MEMORIA
160 POKE56334,PEEK(56334)AND254
    :REM DESCONECTAR RELOJ INTERRUPCIONES
165 POKE1,PEEK(1)AND251
    :PERMUTAR A CARACT. ROM
170 FOR I=0 TO 511      :REM COPIAR
180 POKE14336+I,PEEK(53248+I)
    :REM 64
190 NEXT I
    :REM CARACTERES
200 POKE1,PEEK(1)OR4
    :PERMUTAR A E/S
210 POKE56334,PEEK(56334)OR1
    :REM CONECTAR RELOJ INTERRUPCIONES
220 POKE53272,(PEEK(53272)AND240)OR14:   REM
    PONER PUNTERO CARACT.
230 REM **** COPIA COMPLETA ****
240 :
250 :
300 REM **** EL GIMNASTA ARTURO ****
310 FOR I=14552TO14552+31
    :REM LEER
320 READ A:POKEI,A: NEXT I
    :REM DATOS CARACT.
330 PRINTCHR$(147)
    :REM BORRAR PANTALLA
340 POKE55338,14:POKE55378,14
    :CARACT. COLOR CELESTE
350 POKE1066,27:POKE1106,28
    :REM BRAZOS ARRIBA
360 FORI=1TO500:NEXT I
    :REM BUCLE DEMORA
370 POKE1066,29:POKE1106,30
   :REM BRAZOS ABAJO
380 FORI=1TO500: NEXT I
    :REM BUCLE DEMORA
390 GOTO350
480 :
490 :
500 REM **** DATOS BRAZOS ARRIBA ****
510 DATA129,153,189,153,66,60,60,60
520 DATA60,60,36,36,66,66,195,0
530 REM *** DATOS BRAZOS ABAJO ****
540 DATA0,24,60,24,0,126,189,189
550 DATA189,189,36,36,36,36,102,0
```

128	64	32	16	8	4	2	U	DECIM
1	0	0	0	0	0	0	1	129
1	0	0	1	1	0	0	1	153
1	0	1	1	1	1	0	1	189
1	0	0	1	1	0	0	1	153
0	1	0	0	0	0	1	0	66
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	1	0	0	0	0	1	0	66
0	1	0	0	0	0	1	0	66
1	1	0	0	0	0	1	1	195
0	0	0	0	0	0	0	0	0

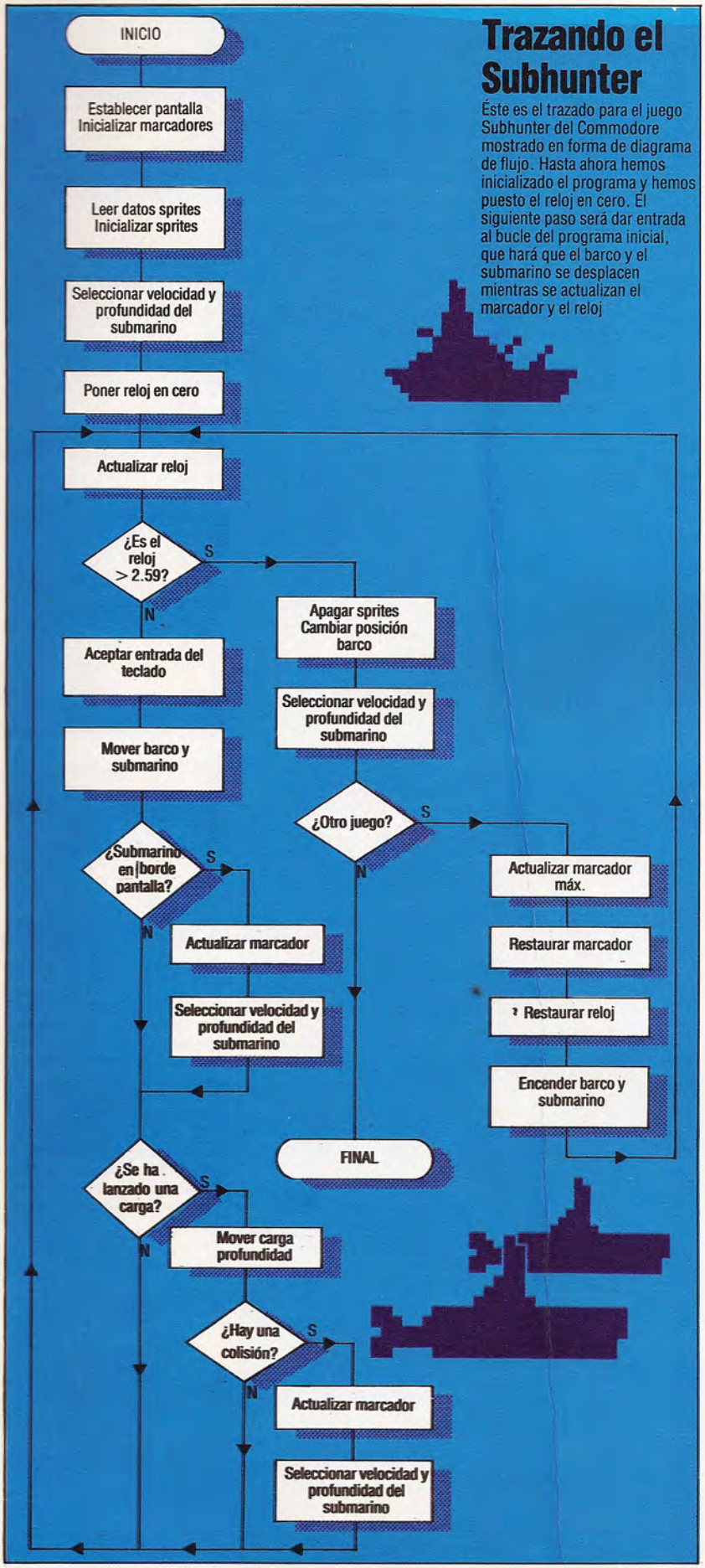
128	64	32	16	8	4	2	U	DECIMA
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	24
0	0	1	1	1	1	0	0	60
0	0	0	1	1	0	0	0	24
0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	126
1	0	1	1	1	1	0	1	189
1	0	1	1	1	1	0	1	189
1	0	1	1	1	1	0	1	189
1	0	1	1	1	1	0	1	189
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	1	1	0	0	1	1	0	102
0	0	0	0	0	0	0	0	0

Liz Dixon

Aspecto de la figura
Los caracteres se construyen en una matriz de puntos de ocho por ocho, descrita en ocho bytes consecutivos. Cada fila del carácter se interpreta como un número binario de un único byte, representando los puntos por 1 y los espacios por 0. Para utilizarlos en programas para el Commodore 64, estos números binarios se deben convertir a decimal

Trazando el Subhunter

Éste es el trazado para el juego Subhunter del Commodore mostrado en forma de diagrama de flujo. Hasta ahora hemos inicializado el programa y hemos puesto el reloj en cero. El siguiente paso será dar entrada al bucle del programa inicial, que hará que el barco y el submarino se desplacen mientras se actualizan el marcador y el reloj



Programa Subhunter

El Commodore 64 posee su propio reloj interno que se puede utilizar para sincronizar programas en BASIC. El reloj tiene seis dígitos, parecidos a un reloj digital, que representan horas (00-23), minutos (00-59) y segundos (00-59). El reloj puede ser accedido desde BASIC con la variable TI$. El valor de TI$ da el tiempo transcurrido desde que se conectó el ordenador a la red, pero se puede reinicializar en cualquier momento. El siguiente programa muestra cómo funciona el reloj:

```
10 REM **** RELOJ ****
20 PRINT CHR$(147)     : REM BORRAR
                         PANTALLA
30 TI$ = "000000"      : REM PONER RELOJ
                         EN CERO
40 PRINTCHR$(145);TI$: REM IMPRIMIR
                         VALOR EN CURSO
                         DEL RELOJ
50                     : REM CHR$(145)=
                         CURSOR ARRIBA
60 GOTO40
```

El programa se ejecuta en un bucle continuo, visualizando el reloj en la pantalla hasta que se pulse la tecla Run/Stop. El juego que estamos escribiendo exige la visualización de un reloj en la pantalla y darlo por terminado después de transcurridos tres minutos. El reloj sólo requiere las partes de TI$ correspondientes a los minutos y segundos. Utilizando las funciones string podemos dividir TI$ así:

RIGHT$(TI$,2)
↓
TI$=HH(MM)(SS)
↑
MID$(TI$,3,2)

Los dos dígitos de los segundos se pueden aislar mediante RIGHT$ (TI$,2) y los dígitos de los minutos se pueden aislar mediante MID$(TI$,3,2).

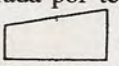
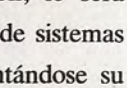
El bucle principal del programa empieza en la línea 200 y termina en la 390. Cargue la subrutina descrita en la última sección y añada estas líneas:

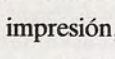
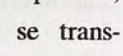
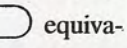
```
140 TI$="000000"
150 :
160 :
200 REM **** BUCLE PRINCIPAL ****
205 :
210 REM ** RELOJ **
220 PRINTCHR$(19);:TAB(14)CHR$(5)"TIEMPO
    ";MID$(TI$,3,2);":"; RIGHT$(TI$,2)
225 IFVAL(TI$)>259THEN400: REM FINAL JUEGO
:
390 GOTO200: REM RECOMENZAR BUCLE
    PRINCIPAL
400 END
```

La línea 140 restablece el reloj en la posición inicial del programa. La línea 220 imprime (PRINT) el actual valor del reloj en minutos y segundos, separados por un punto y coma. TAB (14) hace que queden libres 14 espacios antes de imprimir y posiciona el reloj en el medio de la pantalla. CHR$(5) dará color blanco a los caracteres. La línea 225 transforma TI$ en cantidad numérica. Si el tiempo de juego ha excedido los 2 minutos 59 segundos, éste ha llegado a su fin.

Del diagrama al BASIC

Los símbolos de los ordinogramas tienen una perfecta transcripción a instrucciones BASIC

Todos aquellos símbolos que hasta ahora se han asociado a ideas pueden asimilarse a instrucciones determinadas. Así, el símbolo de entrada por teclado se codificará con el mando INPUT  , y el rectángulo, símbolo de operación, lo será como LET  , si bien la mayoría de sistemas prescinden de su utilización, representándose su contenido en forma algebraica: A = B equivale a LET A = B.

El rombo (símbolo de decisión) se traduce por IF  . La visualización o la impresión, según se emplee  o , se transcribe por PRINT, mientras que las líneas de flujo y los conectores se representan por GOTO. En cuanto a los terminales, el final  equivale a END (en el Spectrum, STOP), y el inicial, y a modo de título, puede representarse con un REM, que también servirá para incluir posibles comentarios. Veamos un ejemplo.

El siguiente ejemplo plantea la visualización de los números pares comprendidos entre 2 y 50, así como sus cuadrados (fig. 1). Se ha utilizado A para denominar al contador y C para el cuadrado. Las líneas 60 y 70, que forman la decisión, pueden transcribirse así:

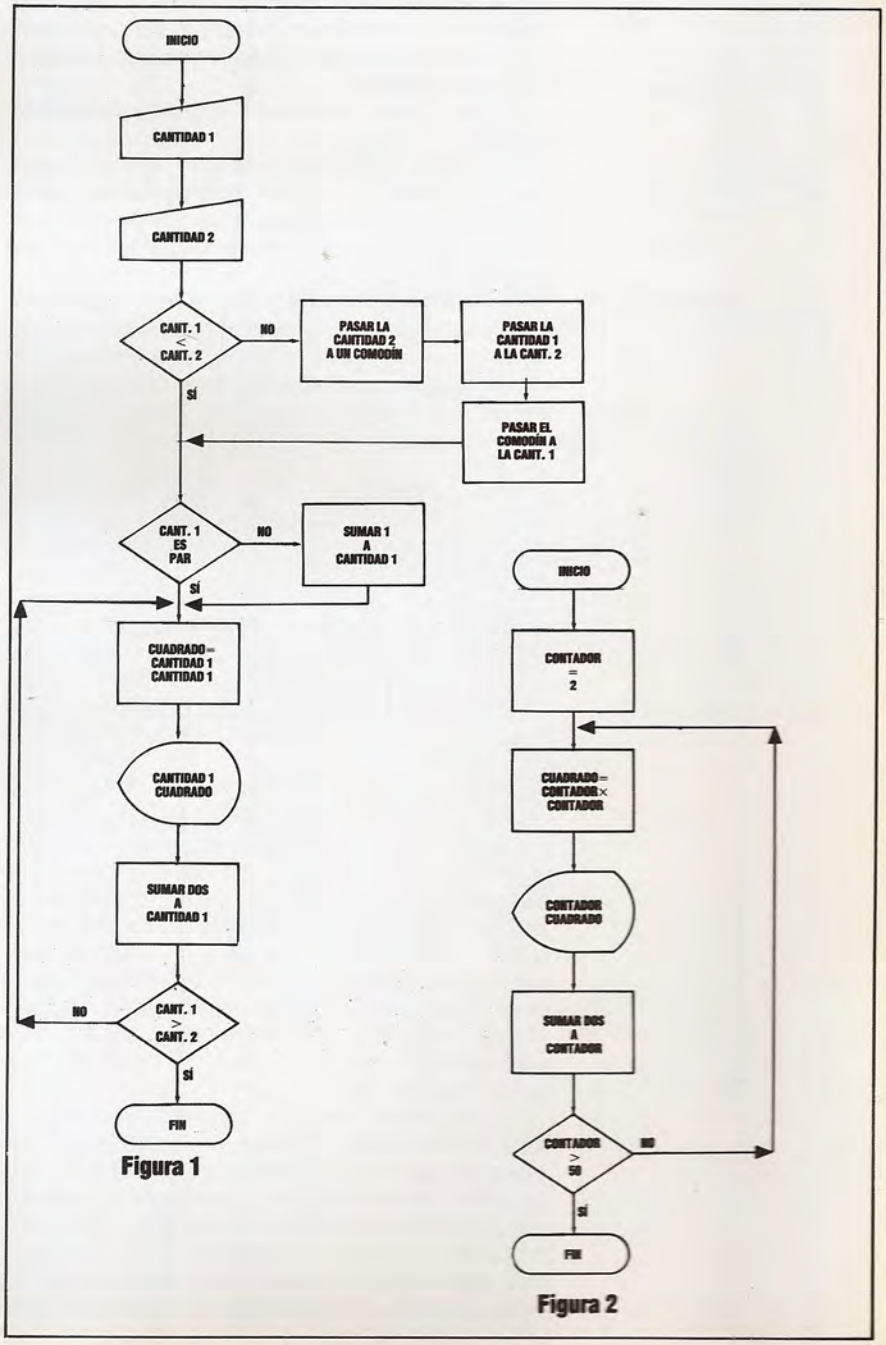
```
60 IF A < 50 OR A = 50 THEN GOTO 30
70 END
```

o bien

```
60 IF A > 50 THEN END
70 GOTO 30
```

Modifiquemos la situación, suponiendo que en lugar de dos números determinados se entren por teclado dos cifras cualesquiera (fig. 2). Se observa entonces que la primera cantidad debe ser siempre par, para que el incremento de dos en dos devuelva el resultado deseado. Pero dicha tarea no queda encomendada a la persona que dé entrada a la cantidad que desee. Debe ser por programa como se controle dicho requisito. Asimismo, debe comprobarse que, dado que el contador se incrementa, la primera cantidad debe ser menor que la segunda, caso contrario se invertirá el orden de las cantidades, con lo cual se visualizarán siempre los números pares de menor a mayor.

```
10 REM PARES Y CUADRADOS
20 INPUT "PRIMERA CANTIDAD";A
30 INPUT "SEGUNDA CANTIDAD";B
35 REM CONTROL DE CANTIDADES
40 IF A<B THEN GOTO 60
50 X=B: B=A: A=X
55 REM CONTROL PARIDAD DE A
60 IF A/2=INT (A/2) THEN GOTO 80
70 A=A+1
75 REM INICIO DEL BUCLE
80 C=A+A
90 PRINT A,C
100 A=A+2
110 IF A>B THEN END
120 GOTO 80
```



Rutina general

Estudiamos los símbolos y etiquetas que hacen manejable el lenguaje assembly y los aplicamos a algunas subrutinas

Debido a que el lenguaje assembly es esencialmente un lenguaje de programación simple que se compone de órdenes "primitivas" que pueda administrar la CPU, usted se encontrará escribiendo y reescribiendo fragmentos de programa para realizar las mismas tareas esenciales que forman parte de las instrucciones de un lenguaje de alto nivel: cómo tratar la entrada-salida, por ejemplo, o las rutinas aritméticas con dos bytes. Lo más recomendable es formar una biblioteca (en cinta, en disco o en papel) de las rutinas que se utilizan con mayor frecuencia para incluirlas en bloque en los nuevos programas en el momento en que se plantee la necesidad de emplearlas.

Existen a este respecto dos problemas fundamentales.

El primero reside en la dificultad de escribir rutinas importantes y, a veces, largas, de una forma que resulte lo suficientemente general como para poder insertarlas en programas diferentes sin reajustes ni reescritura.

El segundo problema está en escribir rutinas útiles que no dependan de unas determinadas posiciones de memoria, y se puedan volver a posicionar en la memoria por medio de un nuevo ensamblaje con una dirección ORG distinta para desempeñar aquí exactamente la misma función que en su posición original.

Ambos problemas son aspectos de un solo tema, el de la generalidad-adaptabilidad, conocido por los programadores de BASIC, y se resuelven de forma bastante similar: hay que recurrir a variables para pasar los valores del programa a la subrutina; usar variables locales en las subrutinas para hacerlas independientes del contexto del programa principal; y, finalmente, evitar de lo posible la utilización de variables absolutas (constantes tanto numéricas como en serie) así como los números de línea de programa.

En la programación en lenguaje assembly nos hemos habituado a la idea de las posiciones de memoria como el equivalente de las variables del BASIC; los programas operan sobre el contenido de estas posiciones, cualesquiera que sean dichos contenidos, igual que un programa en BASIC opera sobre el contenido de sus variables. Lamentablemente, hemos tendido a referirnos a las posiciones de memoria por sus direcciones absolutas, un hábito que al principio era conveniente pero al que en este momento debemos renunciar en favor de la generalización.

La respuesta consiste en utilizar símbolos en lugar de direcciones absolutas y valores, por una parte; y, por otra, en recurrir a la utilización de la gama de formas simbólicas que ofrecen los pseudo-opcodes para el ensamblador como los equivalentes tanto de las variables como de los números de línea de programas. Ya hemos visto anteriormente, en otros apartados del presente curso avanzado, algunos ejemplos de esto.

Habiendo llegado a este punto, consideremos el siguiente programa:

6502			Z80		
DATA1	EQU	$12	DATA1	EQU	$12
DATA2	EQU	$79	DATA2	EQU	$79
	LDA	DATA1	LD	A,(DATA1)	
LOOP	ADC	DATA2	ADC	A,(DATA2)	
	BNE	LOOP	JR	NZ,LOOP	
	RTS		RET		

Aquí hemos utilizado dos tipos de símbolos, dos valores y una etiqueta, todos empleados como operandos de las instrucciones en lenguaje assembly. Debido a ello, el fragmento de programa es a la vez general y transportable. Las únicas cantidades absolutas son los valores de DATA1 y DATA2, y se pueden inicializar en el programa principal sin necesidad de hacerlo al comienzo de la rutina.

Hay otros pseudo-ops que no hemos analizado todavía. En especial, DB, DW y DS (si bien al igual que ORG y EQU, pueden variar de un programa ensamblador a otro). Se trata de siglas inglesas: DB por **Define Byte** (definir byte), DW por **Define Word** (definir palabra) y DS por **Define Storage** (definir almacenamiento), nos permiten inicializar y designar posiciones de memoria, como en este ejemplo:

			ORG	$D3A0
D3A0	5F	LABL1	DB	$5F
D3A1	CE98	LABL2	DW	$98CE
D3B3		LABL3	DS	$10
D3B3		DATA1	EQU	LABL3

TABLA DE SÍMBOLOS:
LABL1 = D3A0: LABL2 = D3A1: LABL3 = D3A3
DATA1 = D3A3
ASSEMBLY COMPLETO — NINGÚN ERROR

En este listado completo de assembly (la salida de un programa ensamblador) vemos en la parte inferior y por primera vez una tabla de símbolos, con los símbolos definidos en el programa y los valores que representan. En este fragmento hay varias cosas importantes que observar. Antes que nada, en la línea que comienza con LABL1 se utiliza el pseudo-op DB. En el listado podemos ver que la directriz ORG ha atribuido a LABL1 la dirección $D3A0, y la tabla de símbolos así lo confirma. Aquí DB tiene el efecto de colocar el valor $5F en el byte direccionado por LABL1; de modo que la posición de memoria $D3A0 contiene inicialmente el valor $5F, tal como podemos ver en la columna de lenguaje máquina del listado.

En segundo lugar, LABL2 hace las veces de la dirección $D3A1. Pero como DW ha de colocar en ella una "palabra" (o sea, dos bytes consecutivos), el valor $98CE se almacena en las posiciones $D3A1 y $D3A2 en forma *lo-hi* (esto se puede ver con toda

claridad en la columna de lenguaje máquina). Dado que DW convierte automáticamente sus operandos a forma *lo-hi*, suele utilizarse para inicializar posiciones de "punteros" con direcciones. LABL2, o la posición $D3A1, podría ser una de tales direcciones: señala la posición $98CE.

Otro punto a considerar es que la instrucción DS $10 tiene el efecto de añadir $10 al contador del programa. Esto está más claro en la tabla de símbolos que en el listado; LABL3 es la posición $D3A3 (la posición que sigue a la instrucción anterior), pero en el listado resulta que su valor es $D3B3. Esto se explica porque la dirección se refiere a la siguiente instrucción después de DS, de modo que DS $10 ha reservado un bloque de 16 bytes (de $D3A3 a $D3B2 inclusive) entre una instrucción y la siguiente. Es un procedimiento parecido al de colocar líneas REM largas en un programa en BASIC con el fin de crear espacio sin utilizar en el área para texto de programas, donde se pueda entonces colocar (POKE) y recuperar (PEEK) un área de programas en lenguaje máquina (véase p. 617).

Por último, se utiliza EQU para establecer un símbolo igual al valor de otro, de modo que DATA1 tiene el valor $D3A3 (el valor de LABL3). Aquí tenemos otra fuente de posible confusión. LABL3 es la representación simbólica de la dirección $D3A3, de manera que DATA1 EQU LABL3 significa "el símbolo DATA1 tendrá el mismo significado y valor que LABL3". El hecho de que la instrucción DB haya hecho que el contenido de $D3A3 fuera igual a $5F no tiene importancia alguna para el significado de los símbolos LABL3 y DATA1. Tener siempre presente con toda claridad la diferencia entre una dirección y su contenido es uno de los ejercicios más aconsejables en las primeras etapas de aprendizaje de la programación en assembly. Es muy probable que usted haya experimentado con anterioridad esta misma sensación con las variables y sus contenidos programando en BASIC.

A primera vista, la directriz DB parece un equivalente de EQU, pero no es así. LABL1 significa "la dirección $D3A0", y DB $5F ha inicializado ese byte con el valor $5F, pero, aunque el valor de LABL1 ahora sea fijo, el contenido de la dirección que simboliza puede cambiar en cualquier momento (p. ej., almacenando allí, según avanza el programa, el contenido del acumulador). De forma similar, ahora DATA1 es un símbolo cuyo valor está fijado por la instrucción EQU; la ejecución del programa no puede cambiar su valor. Y, nuevamente, LABL3 señala el comienzo del área de datos de 16 bytes, cuyos contenidos pueden cambiar en el programa, pero LABL3 en sí mismo es invariable.

Esto aclara, pero no agota, ni mucho menos, la amplia gama de posibilidades que los nuevos pseudo-ops ofrecen al usuario. Consideremos ahora esta nueva versión del fragmento anterior:

```
                        ORG   $D3A0
D3A0   4D4553   LABL1   DB    'MESSAGE 1'
D3A9   CE98     LABL2   DW    $98CE
D3BB            LABL3   DS    $10
D3BB            DATA1   EQU   LABL3

TABLA DE SÍMBOLOS:
LABL1 = D3A0: LABL2 = D3A9: LABL3 = D3AB
DATA1 = D3AB
ASSEMBLY COMPLETO - NINGÚN ERROR
```

A la instrucción DB sigue un string, 'MESSAGE 1', como operando, y el ensamblador ha inicializado las posiciones desde $D3A0 hasta $D3A8 con los valores ASCII de los caracteres encerrados entre comillas simples.

Lo que se deduce de la columna que cuenta las direcciones de posición, y es parcialmente confirmado por la columna con el código de lenguaje máquina: los contenidos de los tres bytes desde $D3A0 hasta $D3A2 resultan ser $4D, $45 y $53, que corresponden a los valores ASCII en hexadecimal de las letras 'M', 'E' y 'S'.

Ésta es una facilidad importante, no sólo porque alivia al programador en la tarea de traducir mensajes y datos de caracteres a listas de códigos ASCII, sino también porque hace que el listado resulte mucho más fácil de leer, y sugiere la posibilidad de obtener alguna salida en pantalla con programas en lenguaje assembly. Esto último es especialmente notable, porque hasta ahora nos hemos limitado a almacenar resultados en la memoria e inspeccionarlos utilizando el programa monitor (véase p. 598).

Naturalmente, en este curso analizaremos cómo tratar la pantalla, pero existen todavía algunos aspectos del lenguaje assembly que necesitamos investigar antes de entrar en ese tema. No obstante, si usted piensa en nuestra costumbre de almacenar resultados en la memoria y si ya ha comprendido que las visualizaciones en pantalla por mapas de memoria son, en realidad, sólo áreas de la memoria, quizá ya intuye la forma de direccionar la pantalla desde un programa.

El aspecto más importante de esta nueva facili-

Ejercicios

1) El primer fragmento de programa del texto principal emplea el pseudo-op DS para reservar $10 bytes de memoria comenzando desde la dirección representada mediante la etiqueta LABL1. Escribir un programa en lenguaje assembly que almacene los números de $0F a $00 por orden descendente en este bloque, a un número por byte. Se ha de usar, para hacerlo, un bucle y técnicas de direccionamiento indexado, para lo que habrá de utilizar las instrucciones DEX (disminuir el registro X) o DEC (IX+0) (disminuir IX). El bucle continuará mientras la disminución del registro índice no ponga en 1 el flag de cero, o sea, ha de utilizar las instrucciones de bifurcación BNE o JR NZ.

2) Utilizando las técnicas del ejercicio anterior, escriba un programa para copiar el mensaje almacenado en LABL1 por el pseudo-op DB (véase el segundo fragmento de programa del texto principal) en un bloque de memoria que comience en la dirección almacenada en LABL2 por el pseudo-op DW. Puede que la dirección $98CE no sea adecuada para su ordenador, y que cambien las posiciones iniciales; pero el programa debe funcionar para cualquier dirección y para mensajes de cualquier longitud. Para implementar esto, su programa debe utilizar el número de caracteres del mensaje como un contador del bucle, o bien debe ser capaz de reconocer el final del mensaje; podría colocar un asterisco, por ejemplo, como último carácter de todo mensaje.

dad DB es que da a LABL1 el rango de una variable string de BASIC. Cuando escribimos en este lenguaje:

```
200 LET A$+"MESSAGE 1"
```

lo que hacemos es crear un señalador del comienzo de una tabla de bytes que contienen los valores ASCII de 'M', 'E', 'S', etc. Cada vez que el intérprete de BASIC encuentra una referencia a A$, busca en su propia tabla de símbolos para hallar la posición que se señala; es decir, la posición de comienzo de los contenidos de A$. Como decimos, en nuestro programa en assembly podemos tratar LABL1 como el equivalente de A$, pues ya hemos escrito un fragmento de programa que nos permite manipular una tabla utilizando el direccionamiento indexado.

Los pseudo-ops, pues, eliminan de nuestros programas direcciones absolutas y valores, y se reemplazan por símbolos. Con esto se disminuyen los problemas de generalización y adaptabilidad. Lo que necesitamos ahora es saber acceder a estos módulos portátiles y adaptables desde el programa principal.

En otras palabras, es preciso contar con un equivalente en lenguaje máquina de la instrucción GOSUB del BASIC.

Existen, por supuesto, instrucciones de estas características: JSR y CALL, en 6502 y Z80, respectivamente. Ambas emplean una dirección absoluta (que puede ser una etiqueta) como operando, y ambas tienen el efecto de sustituir los contenidos del contador del programa con la dirección que constituye su operando. La siguiente instrucción a ejecutar, por lo tanto, será la primera instrucción de la subrutina así direccionada. La ejecución prosigue a partir de esa instrucción hasta encontrar la instrucción RETURN (RTS y RET, respectivamente). Esta orden tiene el efecto de reemplazar los contenidos corrientes del contador del programa con los contenidos inmediatamente anteriores a la ejecución de la instrucción JSR o CALL. La siguiente instrucción a ejecutar, por lo tanto, es la que sigue inmediatamente a JSR o CALL. Éste es exactamente el mecanismo de que se vale el intérprete de BASIC para ir y venir de las sentencias aludidas por GOSUB. Como tal, resulta fácilmente comprensible, pero plantea la cuestión de cómo se restauran los antiguos contenidos del contador del programa una vez ejecutada la instrucción RETURN. Es fácil entenderlo: las instrucciones JSR y CALL primero "empujan" los contenidos del contador del programa en la pila (véase ilustración de p. 616) antes de reemplazarlos por la dirección de la subrutina; y las instrucciones RTS y RET "hacen saltar" esa misma dirección fuera de la pila para devolverla al contador del programa.

Qué es la pila, cómo se "empuja" o se hace "saltar" y por qué uno ha de hacerlo constituyen los temas del próximo capítulo del curso.

Juego de instrucciones

BEQ — BIFURCAR EN CERO — Relativo F0 (2 bytes) — 6502

El valor del byte que sigue al opcode desplaza el contenido del contador del programa.

EFECTO EN EL PSR

S V B D I Z C

MSB [ ][ ][ ][ ][ ][ ][ ] LSB

SIN EFECTO

Ejemplo:

POSICIÓN	LENG. MÁQUINA	LENG. ASSEMBLY
8F00	F0 16	BEQ $16

	ANTES		DESPUÉS
Contador programa	02	lo	18
	8F	hi	8F

Memoria programas: F0 (en $8F00), 16

JR Z — SALTO RELATIVO EN CERO — Relativo 28 (2 bytes) — Z80

El valor del byte que sigue al opcode desplaza el contenido del contador del programa.

EFECTO EN EL PSR

S Z H V N C

MSB [ ][ ][ ][ ][ ][ ][ ] LSB

SIN EFECTO

Ejemplo:

POSICIÓN	LENG. MÁQUINA	ASSEMBLY
8F00	28 16	JR Z,$16

	ANTES		DESPUÉS
Contador programa	02	lo	18
	8F	hi	8F

Memoria programas: 28 (en $8F00), 16

INX — INCREMENTAR REGISTRO X — Implícito E8 (1 byte) — 6502

El contenido del registro X se incrementa en una unidad.

EFECTO EN EL PSR

S V B D I Z C

MSB [X][ ][ ][ ][ ][X][ ] LSB

Ejemplo:

POSICIÓN	LENG. MÁQUINA	ASSEMBLY
F391	E8	INX

	ANTES	DESPUÉS
PSR	????????	0??????1?
X	FF	00

Memoria programas: E8 (en $F391)

INC IX — INCREMENTAR IX — Implícito DD23 (2 bytes) — Z80

El contenido de IX se incrementa en una unidad.

EFECTO EN EL PSR

S Z H V N C

MSB [ ][ ][ ][ ][ ][ ][ ] LSB

SIN EFECTO

Ejemplo:

POSICIÓN	LENG. MÁQUINA	ASSEMBLY
F391	DD23	INC IX

	ANTES		DESPUÉS
Contador programa	FF	lo	00
	E7	hi	E8

Memoria programas: DD (en $F391), 23

Un gran porvenir

Desde su introducción, el CP/M se ha convertido en el estándar de la industria para sistemas operativos

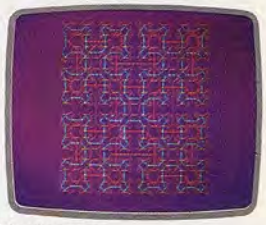
Gary Kildall, uno de los miembros del equipo de Intel que desarrolló el microprocesador 8080, creó su primera versión del sistema CP/M en 1974 para apoyar a un compilador para PL/M, el primer lenguaje de alto nivel que produjo Intel. En 1975 agregó un editor (ED), un ensamblador (ASM) y un *debugger* (DDT), o programa de depuración de programas usuario. Le ofreció el nuevo sistema operativo a Intel, que lo rechazó, hecho que probablemente haya sido lo mejor que le pudiera haber ocurrido a Kildall. Asociado con Dorothy McEwan, empezó a publicar revistas para aficionados y a vender el CP/M de forma particular. El CP/M de Kildall superó rápidamente en ventas a las revistas para aficionados.

Ya sea por el diseño o debido a un rotundo golpe de suerte, Kildall había dado con un sistema que solucionaba en gran medida el principal problema del microordenador en sus primeros años: el de la compatibilidad. Los tres ordenadores de mayor consumo más importantes de finales de los años setenta (el PET, el Apple y el Tandy) tenían sistemas operativos de disco incompatibles, y los productores independientes de software tenían que optar por un formato u otro. El código debía reescribirse por completo para lograr que un producto de software funcionara con una máquina diferente de aquella para la cual había sido diseñado. Pero el CP/M vino a cambiar completamente esta situación: su considerable popularidad significó que una gran mayoría de fabricantes comenzara a adoptarlo, creando por consiguiente un "estándar" de facto. Muchas firmas que habían elegido para sus máquinas los procesadores Intel 8080 o Zilog Z80 especificaron el CP/M porque ofrecía una forma sencilla de manejar el acceso a pantalla, impresora, discos, etc. Su popularidad iba en aumento y cada vez se disponía de más software CP/M, lo que representaba un incentivo aún mayor para adoptarlo.

El Programa de Control para Microprocesadores al principio se autorizó para unos pocos usuarios selectos. La ahora famosa abreviatura inicialmente respondía a "Control Program/Monitor", ¡pero este título tan humilde se cambió enseguida! Hacia 1976, Kildall estaba abrumado por los pedidos que se le hacían del producto. Renunció como profesor de informática en una escuela de Monterey y fundó Digital Research en Pacific Grove (California).

Mientras el CP/M se difundía, Digital Research se concentró en los sistemas para múltiples usuarios y produjo el MP/M. Éste estaba diseñado para ser compatible con el CP/M en todos los aspectos, si bien sus primeras versiones no compartieron nada el éxito del CP/M. El fraccionamiento de las áreas de usuario y otras configuraciones que el programador de sistemas pudiera necesitar hacer no eran en absoluto sencillas y en el tratamiento de archivos difería a veces del CP/M. No obstante, debido a que los costos de los microprocesadores han disminuido a medida que ha ido aumentando la producción, la necesidad de que varios usuarios compartan un procesador ya no tiene ningún sentido desde el punto de vista económico, y el ahora revisado MP/M no ha logrado hacerse popular.

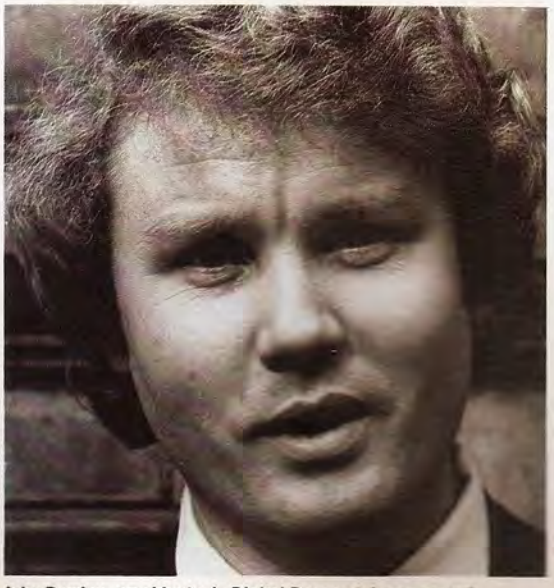
Digital Research reunió fondos de varias empresas de inversión en 1981, para convertirse en una auténtica multinacional, con una presencia notablemente fuerte en Europa (donde tiene oficinas en Gran Bretaña, Alemania y Francia). Aproximadamente al mismo tiempo, Digital Research fue una de las primeras compañías en asegurarse el contrato para desarrollar un sistema operativo para el re-



Diseños LOGO
Digital Research ha entrado en el campo de los lenguajes y está en el primer lugar con su DR LOGO. Uno de sus puntos fuertes lo constituyen sus gráficos

Ian McKinnell

Gráficos de gestión
GXS es un paquete de software pionero diseñado para lograr que las aplicaciones de gestión sean portables entre máquinas diferentes, como el paquete para gráficos de gestión que vemos en la ilustración



Cortesía de Digital Research Inc.

John Rowley, presidente de Digital Research Incorporated

Gary Kildall

Digital Research Incorporated, Massachusetts (Estados Unidos)

Cortesía de Digital Research Inc.

cientemente diseñado Personal Computer de IBM. A pesar de que el contrato para el IBM-PC al final lo obtuvo Microsoft, Digital Research no salió para nada derrotada. Desde entonces actualizó el CP/M para los procesadores de Intel 8088/8086 para hacerlos similares al MS-DOS y también ha dado un nuevo paso adelante con el Concurrent CP/M.

El Concurrent CP/M es el opuesto del MP/M, permitiendo la ejecución simultánea de diversos programas en el mismo procesador. Con este programa, un usuario podría trabajar en tres tareas diferentes al mismo tiempo (p. ej., hoja electrónica, generación de informes y correo electrónico) pasando de una a otra a voluntad. Las versiones existentes del Concurrent CP/M pueden visualizar simultáneamente cada pantalla (o parte de ella) utilizando una configuración de "ventana". Nuevas versiones del Concurrent CP/M prometen ejecutar directamente la mayoría de los programas escritos para el IBM PC-DOS.

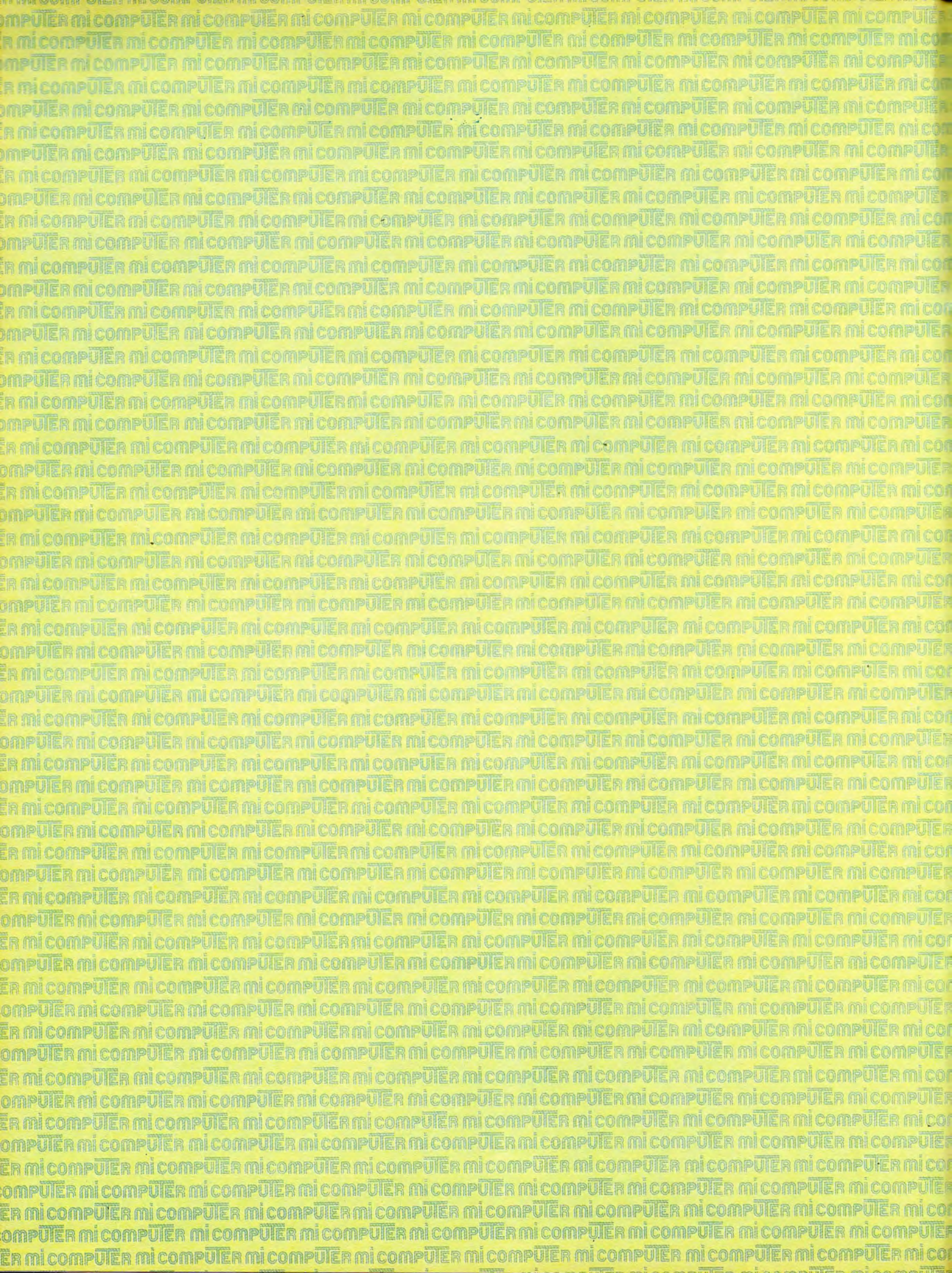
Entre las decisiones estratégicas que han tomado Digital Research y muchas otras firmas de sistemas y lenguajes, está la de concentrar todo su trabajo de desarrollo en el lenguaje C, que es especialmente notable por su portabilidad. El código escrito en C sólo debe ser recompilado para que pueda ser usado en otro procesador, aunque esta característica ha despertado críticas en cuanto a que se trata de una codificación incómoda. Es mejor, razonan sus detractores, hacer un trabajo adecuado en ensamblador para cada procesador individual. Sin embargo, ha ido adquiriendo una creciente popularidad, y puesto que el sistema operativo UNIX, tan ampliamente utilizado, está escrito en C, la tendencia hacia este lenguaje parece irreversible.

Digital Research ha sido ciertamente coherente con su idea de que la verdadera portabilidad sólo es posible a través de lenguajes de alto nivel. Ahora proporciona varios lenguajes para una amplia gama de micros. En el extremo más bajo del mercado, sin embargo, Digital Research ha organizado un Departamento de Productos para el Consumidor que venderá BASIC Personal, CP/M Personal y su propia versión de LOGO. El Personal CP/M, al igual que el CP/M-86, está diseñado para ser almacenado en ROM y pronto estará disponible en un chip Z80 en virtud de un acuerdo con Zilog. Digital Research describe esto como *microware* y está segura de prolongar la vida activa de un gran número de programas CP/M "estándar" en vías de quedar obsoletos haciendo que sean suficientemente baratos para el usuario de ordenadores personales.

Posteriores desarrollos que prometen son VIP y GSX. VIP es una "concha" barata que permite que quienes desarrollan programas puedan presentar una interface uniforme para el usuario, independientemente del paquete de aplicaciones que se esté ejecutando. Varias aplicaciones pueden compartir los mismos datos, y éstos pueden ser transferidos de una a otra. En este sentido el VIP es similar a la tecnología Lisa de Apple y Macintosh pero requiere mucha menos memoria. El VIP puede funcionar en cualquier ordenador con más de 50 Kbytes de RAM y equipado con 150 Kbytes o más de espacio de disco.

El GSX está destinado a ser para los gráficos lo que el CP/M es para los discos. Utiliza un juego estándar de funciones para gráficos que se pueden emplear en una variedad de distintos elementos de hardware. Un programa GSX funciona en una pantalla en color, una pantalla en blanco y negro, una impresora matricial y un plotter, sin ninguna clase de modificaciones. Sin embargo, Digital Research está teniendo dificultades con la creación de un nuevo estándar para gráficos, porque el sistema no produce la misma calidad que los programas que están escritos especialmente para una máquina. Su popularidad también ha sufrido la influencia negativa de la carencia de software.

Digital Research se ha establecido como una de las principales casas de software en el negocio del microordenador. Y, sin embargo, no se está durmiendo en los laureles. Después de su incursión en productos tales como el GSX, el VIP y el LOGO, tiene potencial para seguir a empresas como Microsoft en el campo del software para aplicaciones. Con el insuperable antecedente del éxito del CP/M, la empresa parece preparada para un largo futuro.


mi COMPUTER