Código	Texto y significado	Situación
m	Note out of range Nota fuera de margen Una serie de bemoles o sostenidos ha producido una nota que excede el margen del chip de sonido.	PLAY
n	Out of range Fuera de margen El parámetro de una orden es demasiado grande o demasiado pequeño. Si es muy grande, se produce el error I.	PLAY
o	Too many tied notes Demasiadas notas ligadas Se ha intentado hacer una ligadura con demasiadas notas.	PLAY

Parte 29
Información de referencia

Temas tratados:

Hardware

El +2 está diseñado en torno al microprocesador Z80, el cual funciona a una velocidad de 3.54 MHz (3.54 millones de ciclos por segundo).

La memoria del +2 está dividida en 32K de ROM y 128K de RAM y distribuida en páginas de 16K. A las dos páginas de ROM se accede a través de los 16K inferiores (direcciones 0 a 3FFFh) del mapa de la memoria. Las ocho páginas de RAM (0-7) son accesibles a través de los 16K superiores del mapa (direcciones C000h a FFFFh). La página 5 de la RAM también está situada en el margen de 4000h a 7FFFh, y la página 2 en el comprendido entre 8000h y BFFFh.

Físicamente hablando, la ROM es un solo dispositivo de 32K (similar a un 27256) que el sistema trata como si fuesen dos chips de 16K. La RAM está compuesta por dieciséis chips de 64K × 1 bit (4164), algunos de los cuales (bancos RAM4 a RAM7) están compartidos entre los circuitos que producen la imagen de televisión (de los que hablaremos más adelante) y el Z80A. Los otros ocho chips (bancos RAM0 a RAM3), así como la ROM, son de uso exclusivo del Z80A.

La matriz lógica no comprometida (ULA, *Uncommitted Logic Array*) controla la mayor parte de las operaciones de E/S: teclado, magnetófono, pantalla, etc. Convierte los bytes que se encuentran en la memoria en formas y colores para la pantalla y permite al Z80A explorar el teclado y leer y escribir los datos en la cinta.

El sonido de tres canales es producido por el AY-3-8912, un chip de sonido muy popular; este dispositivo controla también las puertas RS232, MIDI y el subteclado numérico. La manera en que trabaja es bastante compleja; se recomienda a quienes se sientan tentados a experimentar que consulten la hoja de datos del AY-3-8912. No obstante, la siguiente información debería ser suficiente para empezar. El chip de sonido contiene dieciséis registros que se seleccionan escribiendo primero el número de registro en la puerta de escritura de direcciones (direción de E/S FFFDh, 65533 en decimal) y después leyendo el valor del registro (en la misma dirección) o escribiendo en la dirección de escritura de registros de datos (BFFDh, 49149 en decimal). Una vez que ha sido seleccionado un registro, se puede realizar cualquier número de operaciones de lectura o escritura de datos. Sólo habrá que volver a escribir en la puerta de escritura de direcciones cuando se necesite seleccionar otro registro.

La frecuencia de reloj básica de este circuito es 1.7734 MHz (con precisión del 0.01%).

Los registros hacen lo siguiente:

R0 Ajuste fino del tono, canal A
R1 Ajuste aproximado del tono, canal A
R2 Ajuste fino del tono, canal B
R3 Ajuste aproximado del tono, canal B
R4 Ajuste fino del tono, canal C
R5 Ajuste aproximado del tono, canal C

El tono de cada canal es un valor de 12 bits formado combinando los bits D3-D0 del registro de ajuste aproximado y los bits D7-D0 del registro de ajuste fino. La unidad básica del tono es la frecuencia de reloj dividida por 16 (es decir, 110.83 KHz). Como el contador es de 12 bits, se puede generar frecuencias de 27Hz a 110 KHz.

R6 Control del generador del ruido, D4-D0

El periodo del generador de ruido se toma contando los cinco bits inferiores del registro de ruido cada periodo del reloj de sonido dividido por 16.

R7 Control del mezclador y de E/S

D7 No utilizado
D6 1=puerta de entrada, 0=puerta de salida
D5 Ruido en el canal C
D4 Ruido en el canal B
D3 Ruido en el canal A
D2 Tono en el canal C
D1 Tono en el canal B
D0 Tono en el canal A

Este registro controla la mezcla de ruido y tono para cada canal y la dirección de la puerta de E/S de ocho bits. Un cero en un bit de mezcla indica que la función está activada.

R8 Control de amplitud del canal A
R9 Control de amplitud del canal B
RA Control de amplitud del canal C

D4 1=utilizar generador de envolvente
0=utilizar el valor de D3-D0 como amplitud
D3-D0 Amplitud

Estos tres registros controlan la amplitud de cada canal y si ésta debe ser modulada o no por los registros de envolvente.

RB Ajuste aproximado del periodo de envolvente
RC Ajuste fino del periodo de envolvente

Los valores de ocho bits de RB y RC se combinan para producir un número de 16 bits que se cuenta en unidades de 256 por el periodo del reloj de sonido. Las frecuencias de envolvente pueden estar entre 0.1Hz y 6KHz.

RD Control de envolventes

D3 Continua
D2 Ataque
D1 Alternada
D0 Sostenida

El diagrama de las formas de envolvente (Parte 19 de este capítulo) da una ilustración gráfica de los posibles estados de este registro.

Parte 30
BASIC

Temas tratados:

Números
Variables
Cadenas
Funciones
Resumen de palabras clave
Operaciones matemáticas

Los números se almacenan en BASIC con precisión de 9 o 10 dígitos. El mayor número que puede manejar BASIC es aproximadamente 10^{38}; el más pequeño (positivo) es aproximadamente $4*10^{-39}$.

La forma de almacenamiento es la *binaria de punto flotante*, con un byte para el exponente (e, $1 \leqslant e \leqslant 255$) y cinco bytes para la mantisa (m, $\frac{1}{2} \leqslant m < 1$), *lo que representa el número* $m*2^{e-128}$.

Puesto que $\frac{1}{2} \leqslant m < 1$, el bit más significativo de la mantisa m siempre es 1. Por lo tanto, podemos utilizarlo como indicador del signo (0 para los números positivos y 1 para los negativos).

Los enteros pequeños tienen una representación especial en la que el primer byte es 0, el segundo es el byte del signo (0 o FFh) y el tercero y el cuarto son el entero propiamente dicho (en complemento a dos) con el byte menos significativo en primer lugar.

Los nombres de las variables numéricas son de longitud arbitraria; su primer carácter siempre es una letra y los siguientes pueden ser letras o dígitos. BASIC permite los espacios, pero los ignora; además convierte internamente todas la letras en minúsculas.

Los nombres de las variables de control de los bucles FOR...NEXT consisten en una sola letra.

Los nombres de las matrices numéricas consisten en una sola letra, que puede coincidir con el nombre de una variable sencilla. Estas matrices pueden tener múltiples dimensiones de tamaño arbitrario. Los subíndices empiezan en el 1.

Las cadenas son de longitud completamente arbitraria. Su nombre consiste en una sola letra seguida de $.

Las matrices literales pueden tener múltiples dimensiones de tamaño arbitrario. Su nombre consiste en una sola letra seguida de $ y no puede coincidir con el nombre de

una variable literal sencilla. Todas las cadenas de una matriz dada tienen la misma longitud fija, que está especificada por el último parámetro de la sentencia DIM. Los subíndices empiezan con el 1.

Disección de cadenas. Dada una cadena, se puede especificar una subcadena utilizando los llamados *cortadores*. Los cortadores pueden tener cualquiera de las siguientes formas:

(i) vacío
(ii) una expresión numérica
(iii) *expresión numérica opcional* TO *expresión numérica opcional*

La subcadena se expresa mediante:

(*a*) *expresión literal*(*cortador*)

o

(*b*) *variable de matriz literal*(*subíndice,...,subíndice,cortador*)

que es lo mismo que

variable de matriz literal(*subíndice,...,subíndice*)(*cortador*)

Supongamos que la expresión literal, caso (*a*), es s$.

En el caso (*a*), si el cortador es vacío el resultado es la misma expresión literal s$ (que se considera como subcadena de sí misma).

Si el cortador es una expresión numérica, con valor *m*, el resultado es el *m*-ésimo carácter de s$ (o sea, la subcadena tiene longitud 1).

Si el cortador tiene la forma (iii), supongamos que la primera expresión numérica tiene el valor *m* (el valor por defecto es 1) y la segunda el valor *n* (el valor por defecto es la longitud de la expresión literal). Si $1 \leqslant m \leqslant n \leqslant$ LEN s$, el resultado es la subcadena de s$ que comienza en el *m*-ésimo carácter y acaba en el *n*-ésimo.

Si $0 \leqslant n < m$, el resultado es la cadena vacía. En cualquier otro caso se produce el error 3.

BASIC realiza la disección de las cadenas antes de evaluar las funciones y operaciones, a menos que los paréntesis impongan lo contrario.

A las subcadenas se les puede asignar valor (véase LET). Si una cadena debe contener comillas, al especificarla se debe poner el signo de comillas (") repetido.

Funciones

El argumento de una función no necesita paréntesis si es una constante o una variable (opcionalmente, subindexada o producto de una disección).

Función	Tipo de argumento	Resultado
ABS	Número	Valor absoluto.
ACS	Número	Arco coseno en radianes. Error A si el argumento no está entre -1 y $+1$.
AND	Operación binaria. El operando de la derecha siempre es un número.	
	Si el de la izquierda es un número:	a AND b es $\begin{cases} a \text{ si } b<>0 \\ 0 \text{ si } b=0 \end{cases}$
	Si el de la izquierda es una cadena:	a$ AND b es $\begin{cases} a\$ \text{ si } b<>0 \\ "" \text{ si } b=0 \end{cases}$
ASN	Número	Arco seno en radianes. Error A si el argumento no está entre -1 y $+1$.
ATN	Número	Arco tangente en radianes.
ATTR	Dos argumentos, x e y, ambos numéricos, escritos entre paréntesis	Un número cuya forma binaria codifica los atributos de la posición de carácter que está en la línea x, columna y, de la pantalla. El bit 7 (el más significativo) es 1 para parpadeo, 0 para fijo. El bit 6 es 1 para brillo intenso, 0 para normal. Los bits 5 al 3 dan el color del papel. Los bits 2 al 0 dan el color de la tinta. Error B a menos que $0 \leqslant x \leqslant 23$ y $0 \leqslant y \leqslant 31$.
BIN		No es realmente una función, sino una notación alternativa para números: BIN seguido de una secuencia de ceros y unos es el número cuya representación binaria es la especificada por esa secuencia.
CHR$	Número	El carácter cuyo código es el argumento (redondeado al entero más próximo).

Función	Tipo de argumento	Resultado
CODE	Cadena	Código del primer carácter de la cadena (o 0, si se trata de la cadena vacía).
COS	Número (en radianes)	Coseno del argumento.
EXP	Número	Número *e* elevado al argumento.
FN		FN, seguida de una letra, invoca una función definida por el usuario (véase DEF). Los argumentos tienen que estar entre paréntesis. (Hay que poner los paréntesis aunque no haya argumentos.)
IN	Número	Resultado de leer (a nivel del microprocesador) la puerta de entrada especificada por el argumento x ($0 \leqslant x \leqslant$ FFFFh). Carga el par de registros bc con x y ejecuta la instrucción de lenguaje ensamblador in a,(c).
INKEY$	Ninguno	Lee el teclado. El resultado es el carácter que representa la tecla pulsada, si está pulsada una y sólo una; de lo contrario, el resultado es la cadena vacía.
INT	Número	Parte entera (redondeado hacia abajo).
LEN	Cadena	Longitud del argumento.
LN	Número	Logaritmo neperiano o natural (de base *e*). Error A si $x \leqslant 0$.
NOT	Número	0 si $x<>0$, 1 si $x=0$. NOT tiene prioridad de nivel 4.
OR	Operación binaria. Ambos operandos son numéricos	a OR b es $\begin{cases} 1 \text{ si } b<>0 \\ a \text{ si } b=0 \end{cases}$ OR tiene prioridad de nivel 2.
PEEK	Número	El valor del byte que está en la posición de memoria cuya dirección es el argumento (redondeado al entero más cercano). Error B si el argumento no está entre 0 y 65535.
PI	Ninguno	Número π (3.1415927...).
POINT	Dos argumentos, x e y, ambos numéricos, escritos entre paréntesis	1 si el pixel espeficidao por (x,y) tiene el color de la tinta; 0 si tiene el del papel. Error B a menos que $0 \leqslant x \leqslant 255$ y $0 \leqslant y \leqslant 175$.

Función	Tipo de argumento	Resultado
RND	Ninguno	El siguiente número pseudoaleatorio de una sucesión que se genera tomando las potencias de 75 módulo 65537, restando 1 y dividiendo por 65536. $0 \leqslant y < 1$.
SCREEN$	Dos argumentos, x e y, ambos numéricos, escritos entre paréntesis	El carácter que está (normal o invertido) en la posición de la pantalla especificada por la fila x, columna y. Da la cadena vacía si no se reconoce el carácter.
		Error B a menos que $0 \leqslant x \leqslant 23$ y $0 \leqslant y \leqslant 31$.
SGN	Número	Signo del número. Da -1 para los negativos, 0 para el 0 y $+1$ para los positivos.
SIN	Número (en radianes)	Seno del argumento.
SQR	Número	Raíz cuadrada. Error A si $x<0$.
STR$	Número	La cadena de caracteres que aparecería en la pantalla si se escribiera el número.
TAN	Número (en radianes)	Tangente del argumento.
USR	Número	Invoca a la subrutina de código de máquina cuya dirección inicial es el argumento. Al retornar, el resultado es el contenido del par de registros bc.
	Cadena	Dirección de la descripción de la forma del gráfico de usuario correspondiente al argumento. Error A si el argumento no es una sola letra, de la a a la u, o un gráfico de usuario.
VAL	Cadena	Evalúa el argumento (tras suprimir las comillas) como si fuera una expresión numérica. Error C si el argumento no es válido como expresión numérica. Pueden darse otros errores, dependiendo de la forma de la expresión.
VAL$	Cadena	Evalúa el argumento (tras suprimir las comillas de sus extremos) y da la expresión literal resultante. Error C si el argumento contiene un error de sintaxis o da un valor numérico. Pueden darse otros errores, dependiendo de la expresión.
–	Número	Negación.

Las siguientes son operaciones binarias:

+	Suma de números o concatenación de cadenas
−	Resta
*	Multiplicación
/	División
↑	Elevación a una potencia. Error B si el operando de la izquierda es negativo
=	Igual que
>	Mayor que
<	Menor que
<=	Igual o menor que
>=	Igual o mayor que
<>	Distinto de

Ambos operandos deben ser del mismo tipo. El resultado es el número 1 si la proposición es 'verdadera'; 0 si no lo es.

Las funciones y operaciones tienen las siguientes prioridades:

Operación	Nivel de prioridad
Subíndices y disección de cadenas	12
Todas las funciones excepto NOT y negación	11
↑	10
Negación (signo menos usado para negar)	9
*, /	8
=, − (signo menos usado para restar)	6
=, >, <, <=, >=, <>	5
NOT	4
AND	3
OR	2

Sentencias

Notación utilizada en esta lista:

a	Representa una sola letra.
v	Representa una variable.
x, *y*, *z*	Representan expresiones numéricas.
m, *n*	Representan expresiones numéricas que BASIC redondea al entero más próximo.
e	Representa una expresión.
f	Representa una expresión cuyo valor es una cadena.
s	Representa una secuencia de sentencias separadas por signos de dos puntos.
c	Representa una secuencia de cláusulas de color, cada una de las cuales termina en una coma o en un signo de punto y coma. Una cláusula de color tiene la forma de una sentencia PAPER, INK, FLASH, BRIGHT, INVERSE u OVER.

Observe que en todo lugar se puede poner expresiones en lugar de constantes (excepto para el número de línea al principio de una sentencia).

Todas las sentencias, a excepción de INPUT, DEF FN y DATA, pueden ser usadas como órdenes directas o en líneas de programa (aunque no tendrán la misma utilidad en ambos casos). Tanto las órdenes directas como las líneas de programa pueden constar de varias sentencias separadas por signos de dos puntos. No hay restricción alguna sobre en qué lugar de la línea puede aparecer una sentencia concreta; véase, no obstante, IF y REM.

BEEP *x,y*
Hace sonar una nota a través del altavoz del televisor durante *x* segundos a una altura de *y* semitonos por encima de la nota DO medio.

BORDER *m*
Establece el color del borde de la pantalla superior y también el color del papel para la pantalla inferior.

BRIGHT *n*
Establece el brillo de los caracteres que se escriba en lo sucesivo; *n*=0 para normal, 1 para brillo y 8 para transparente.

Error K si *n* no es 0, 1 ni 8.

CAT
Sólo funciona con microunidades, etc.

CAT !
Da una lista de los ficheros que hay actualmente en el disco de silicio.

CIRCLE *x,y,z*
Dibuja un arco de circunferencia, con centro en (*x,y*) y radio *z*.

CLEAR
Borra todas las variables, liberando así el espacio que ocupaban. Ejecuta RESTORE y CLS; reajusta la posición del cursor gráfico en el extremo inferior izquierdo y borra la pila de GO SUB.

CLEAR *n*
Igual que CLEAR, pero cambia la variable de sistema RAMTOP a *n* (si ello es posible) y coloca en esa dirección la nueva pila de GO SUB.

CLOSE #
Sólo funciona con microunidades, etc.

CLS
Borra el fichero de imagen (memoria de pantalla).

CONTINUE
Reanuda la ejecución del programa a partir de la sentencia en que éste se había detenido emitiendo un informe distinto del 0. Si el mensaje fue 9 o L, continúa a partir de la sentencia siguiente; de lo contrario, repite la sentencia en la que se produjo el error. Si el último informe se produjo en una línea de órdenes, CONTINUE intentará completar dicha línea y entrará en un bucle si el error fue en 0:1, generará el mensaje 0 si fue en 0:2, o bien el mensaje N si tuvo lugar en 0:3 o posterior.

COPY — Envía a la impresora (si está conectada) una copia de las 22 líneas superiores de la pantalla en formato de mapa de bit Epson de densidad cuádruple; de lo contrario, no hace nada. Mensaje D si se pulsa BREAK.

DATA $e_1,e_2,e_3,\ldots$ — Define un tramo de la lista de datos. Debe estar en un programa; de lo contrario, no produce efecto.

DEF FN $a(a_1,\ldots,a_k)=e$ — Definición de una función de usuario. Debe estar en un programa; de lo contrario, no produce efecto. a y a_1 a a_k son, cada una, una sola letra (o una letra seguida de $ para especificar un resultado o un argumento literal). Toma la forma DEF FN $a()=e$ si no hay argumentos.

DIM $a(n_1,\ldots,n_k)$ — Borra la matriz a (si existe) y forma la matriz numérica a con k dimensiones, $n_1,\ldots,n_k$.

Inicializa todos los valores a 0.

DIM $a\$(n_1,\ldots,n_k)$ — Borra la matriz o cadena $a\$$ (si existen) y forma una matriz de caracteres $a\$$ con k dimensiones, $n_1,\ldots,n_k$. Inicializa todos los valores a "". Se puede considerar esta matriz como matriz de cadenas de longitud fija n_k, con $k-1$ dimensiones $(n_1,\ldots,n_{k-1})$. Una matriz está indefinida hasta que se la dimensiona con DIM.

Error 4 si no queda espacio para la matriz en la memoria.

DRAW x,y — Equivale a DRAW $x,y,0$.

DRAW x,y,z — Dibuja una línea (recta o arco de circunferencia) desde la posición actual del cursor gráfico desplázandose x unidades en horizontal e y unidades en vertical en relación con el punto de partida, y al mismo tiempo girando un ángulo z. Error B si la línea se sale de la pantalla.

ERASE — Sólo funciona con microunidades, etc.

ERASE ! f — Borra un fichero del disco de silicio.

FLASH n — Establece si los caracteres deben ser parpadeantes o fijos; $n=0$ para fijos, $n=1$ para parpadeo, $n=8$ para que no haya cambios.

FOR $a=x$ TO y FOR $a=x$ TO y STEP 1 FOR $a=x$ TO y STEP z

Borra la variable ordinaria a (si existe); define la variable de control a con valor inicial x, límite y, paso z y con una dirección de retorno del bucle que hace referencia a la sentencia que sigue a FOR. Comprueba si el valor inicial es mayor (si $z\geqslant 0$) o menor (si $z<0$) que el límite; en caso afirmativo, salta a la sentencia NEXT a, dando el error I si ésta no existe. Véase NEXT.

Error 4 si no queda espacio para la variable de control en la memoria.

FORMAT *f;n*

Establece en *n* baudios la velocidad de transmisión para el dispositivo *f*. Son dispositivos válidos "p" y "P" (la puerta RS232) y velocidades válidas las comprendidas entre 75 y 19200.

GO SUB *n*

Deposita el número de línea *n* en la pila; después actúa como GO TO *n*.

Error 4 si no hay suficientes sentencias RETURN.

GO TO *n*

Salta a la línea de número *n* (o, si no existe, a la primera línea posterior a ese número).

IF *x* THEN *s*

Si *x* es 'verdadero' (distinto de cero), ejecuta *s*. Observe que *s* comprende todas las sentencias hasta el final de la línea. La forma IF *x* THEN *número-de-línea* no está permitida.

INK *n*

Establece el color de la tinta (primer plano) para los caracteres que se escriba en lo sucesivo; *n* está en el margen de 0 a 7 para un color, $n=8$ para transparentes y $n=9$ para contraste.

Error K a menos que $0 \leqslant n \leqslant 9$.

INPUT ...

'...' es una secuencia de elementos de INPUT, separados, al igual que en una sentencia PRINT, por comas, signos de punto y coma o apóstrofos. Un elemento de INPUT puede ser cualquiera de los siguientes:

(i) Cualquier elemento de PRINT que no empiece por una letra.

(ii) Un nombre de variable.

(iii) LINE seguido de un nombre de variable de tipo literal.

Los elementos de PRINT y los separadores del apartado (i) son tratados exactamente igual que en PRINT, con la excepción de que la escritura se dirige a la pantalla inferior. Para (ii), el ordenador se detiene y espera la introducción de una expresión por el teclado; el valor de esa expresión es asignado a la variable. Los caracteres escritos son reproducidos en la pantalla en la forma habitual; los errores de sintaxis producen un signo de interrogación en negativo y parpadeante. En el caso de expresiones de tipo literal, el tampón de entrada se inicializa de forma que contenga dos comillas (que pueden ser borradas, si es necesario). Si el primer carácter de la entrada es STOP (SIMB A), el programa se detiene y emite el error H. (iii) es igual que (ii), con la diferencia de que la entrada es tratada como constante literal sin comillas y el mecanismo de STOP no funciona (en su lugar, para detener la sentencia hay que pulsar la tecla ↓).

INVERSE *n*	Controla la inversión de los caracteres que se escriba en lo sucesivo. Si $n=0$, los caracteres aparecerán en video normal, es decir, color de la tinta sobre color del papel. Si $n=1$, los caracteres aparecen en video inverso, esto es, color del papel sobre color de la tinta. Error K (véase la Parte 28 de este capítulo) si *n* no es 0 ni 1. En 48 BASIC, la pulsación de la tecla VIDEO INV es equivalente a INVERSE 1; la pulsación de VIDEO NORM equivale a INVERSE 0.
LET *v=e*	Asigna el valor de *e* a la variable *v*. No se puede omitir la palabra 'LET'. Una variable sencilla está indefinida mientras no se le asigne valor en una sentencia LET, READ o INPUT. Si *v* es una variable literal indexada o una variable literal obtenida por disección de cadenas (subcadena), la asignación es «procrusteana» (de longitud fija); es decir, el valor de *e* es truncado o rellenado con espacios por la derecha, con el fin de darle la longitud que se ha especificado para *v*.
LIST	Equivale a LIST 0.
LIST *n*	Lista el programa en la pantalla superior, comenzando en la primera línea cuyo número es, como mínimo, *n*; convierte la línea *n* en línea actual.
LLIST	Equivale a LLIST 0.
LLIST *n*	Como LIST, pero dirigiendo el listado hacia la impresora.
LOAD *f*	Carga el programa y las variables.
LOAD *f* DATA ()	Carga una matriz numérica.
LOAD *f* DATA $()	Carga una matriz literal.
LOAD *f* CODE *m,n*	Carga (como máximo) *n* bytes, comenzando en la dirección *m*.
LOAD *f* CODE *m*	Carga bytes comenzando en la dirección *m*.
LOAD *f* CODE	Carga bytes en la dirección desde la cual fueron grabados.
LOAD *f* SCREEN$	Equivale a LOAD *f* CODE 16384,6912.
LOAD !	Como LOAD (véase más arriba las opciones), pero usa el disco de silicio.
LPRINT ...	Como PRINT, pero dirigiendo la salida hacia la impresora.
MERGE *f*	Como LOAD *f*, pero no borra las líneas ni las variables del programa antiguo, salvo las que tengan el mismo número o nombre que en el programa nuevo.
MERGE ! *f*	Como MERGE *f*, pero usa el disco de silicio.
MOVE f_1,f_2	Sólo funciona con microunidades, etc.

NEW

Inicializa el sistema BASIC, borrando todos los programas y variables y toda la memoria a la que BASIC tiene acceso (hasta la dirección dada por la variable de sistema RAMTOP). Conserva variables de sistema UDG, P-RAMT, RASP y PIP. Devuelve el control al menú de presentación, pero no afecta al disco de silicio.

NEXT *a*

(i) Busca la variable de control *a*.
(ii) Añade a su valor el valor del paso especificado en FOR.
(iii) Si el paso es igual o mayor que 0 y el valor de *a* es mayor que el límite, o si el paso es menor que 0 y el valor de *a* es menor que el límite, salta a la sentencia de retorno del bucle.

Error 2 si no existe la variable *a*.

Error 1 si la variable *a* no es la misma que la especificada en FOR.

OPEN #

Sólo funciona con microunidades, etc.

OUT *m,n*

Envía el byte *n* a la puerta *m*. (Carga el par de registros bc con *m* y el registro a con *n* y ejecuta la instrucción out (c),a.)

Error B a menos que $0 \leqslant m \leqslant 65535$ y $-255 \leqslant n \leqslant 255$.

OVER *n*

Controla la sobreimpresión para los caracteres que se escriba en lo sucesivo.

Si $n=0$, los caracteres sustituyen a los que hubiera previamente en esa posición.

Si $n=1$, los caracteres nuevos se mezclan con los antiguos para dar el color de la tinta dondequiera que uno de ellos (pero no ambos) tuviese dicho color, y el color del papel donde ambos fuesen papel o ambos tinta.

Error K a menos que *n* sea 0 o 1.

PAPER *n*

Como INK, pero para controlar el color del papel (fondo).

PAUSE *n*

Detiene el programa y muestra la imagen durante *n* barridos del cuadro (a 50 barridos por segundo; 60 por segundo en EE.UU.), o hasta que se pulsa una tecla. Si $n=0$ la pausa no se cronometra, sino que dura hasta que se pulsa una tecla.

Error B a menos que $0 \leqslant n \leqslant 65535$.

PLAY *f(,f,f,f,...)*

Interpreta hasta ocho cadenas (véase la Parte 19 de este capítulo) y las ejecuta simultáneamente. Las primeras tres cadenas son ejecutadas a través del altavoz del televisor y (opcionalmente) a través de la puerta de MIDI; todas las siguientes van a la puerta MIDI.

PLOT *c;m n*

Dibuja un punto de tinta (supeditado a OVER y a INVERSE) en el pixel (*m,n*); establece en este punto la nueva 'posición actual' del cursor gráfico.

A no ser que los elementos de color *c* especifiquen otra cosa, el color de la tinta de la posición de carácter que contiene al pixel se cambia por el color de tinta permanente actual, y los otros (color del papel, parpadeo y brillo) permanecen como estaban.

Error B a menos que $0 \leqslant m \leqslant 255$ y $0 \leqslant n \leqslant 175$.

POKE *m,n*

Escribe el valor *n* en la posición de memoria cuya dirección es *m*.

Error B a menos que $0 \leqslant m \leqslant 65535$ y $-255 \leqslant n \leqslant 255$.

PRINT ...

'...' es una secuencia de elementos de PRINT, separados entre sí por comas, signos de punto y coma o apóstrofos. La sentencia los escribe en el fichero de imagen (memoria de pantalla) para su salida por la pantalla.

Un punto y coma entre dos elementos no produce ningún efecto (sirve sólo para separar los elementos); una coma produce el carácter de control de la 'coma'; un apóstrofo produce el 'retorno del carro' (equivalente a la pulsación de la tecla INTRO), que PRINT emite por defecto si la sentencia no termina en punto y coma, coma o apóstrofo).

Un elemento de PRINT puede ser:

(i) Vacío; es decir, nada.

(ii) Una expresión numérica.

Primero se escribe un signo menos si el valor es negativo. Supongamos que *x* es el valor absoluto de la expresión. Si $x \leqslant 10^{-5}$ o $x \geqslant 10^{13}$, el número se escribe en notación científica. La mantisa tiene hasta ocho dígitos (sin ceros por la derecha); el punto decimal (ausente si sólo hay un dígito) va detrás del primero. La parte del exponente es E, seguido de + o − y, a continuación, de uno o dos dígitos. Si *x* no está en el margen mencionado, el número se escribe en notación decimal con hasta diez dígitos significativos y sin ceros por la derecha después del punto decimal.

(iii) Una expresión literal.

Las claves contenidas en la cadena son interpretadas, posiblemente con un espacio antes o después. Los caracteres de control producen su efecto de control. Los caracteres irreconocibles se convierten en ?.

(iv) AT *m,n*.

Emite un carácter de control AT seguido por un byte para *m* (número de fila) y otro para *n* (número de columna).

(v) TAB *n*.

Emite un carácter de control TAB seguido por dos bytes para *n* (primero el byte menos significativo), que es tope de tabulación.

(vi) Un elemento de color, que toma la forma de una sentencia PAPER, INK, FLASH, BRIGHT, INVERSE u OVER.

RANDOMIZE Equivale a RANDOMIZE 0.

RANDOMIZE *n* Establece la variable de sistema (llamada SEED) que se va a usar para generar el próximo valor de RND. Si $n<>0$, se le da a SEED el valor *n*. Si $n=0$, a SEED se le da el valor de otra variable de sistema, FRAMES, que lleva la cuenta de los barridos del cuadro exhibidos hasta ahora en la pantalla y que, por consiguiente, debería ser bastante aleatoria. Error B a menos que $0 \leqslant n \leqslant 65535$.

READ $v_1,v_2,\ldots,v_k$ Asigna valores a las variables leyéndolos en las sucesivas expresiones de la lista de datos.

Error C si una expresión es de tipo inadecuado para la variable a la que debería ser asignada.

Error E si se ha agotado la lista de datos cuando todavía quedaban variables a las que asignar valor.

REM ... No produce ningún efecto. '...' puede ser cualquier secuencia de caracteres que termine en 'retorno del carro' (INTRO). No se ejecutará ninguna sentencia que esté detrás de la palabra 'REM' en la misma línea; los signos de dos puntos *no* serán considerados separadores de sentencias.

RESTORE Equivale a RESTORE 0.

RESTORE *n* Dirige el «puntero» de datos hacia la primera sentencia DATA de la línea *n*. Si no existe esa línea (o si no contiene ninguna sentencia DATA), el puntero de dirige hacia la primera sentencia DATA posterior a la línea *n*. La siguiente sentencia READ empezará a leer a partir de ese punto.

RETURN Lee en la pila de GO SUB la referencia a una sentencia y salta a la línea siguiente a ella.

Error 7 cuando en la pila no hay referencia a ninguna sentencia. (Esto quiere decir que probablemente hay algún error en el programa; asegúrese de que todos los GO SUB están emparejados con un RETURN.)

RUN	Equivale a RUN 0.
RUN *n*	Ejecuta CLEAR y después GO TO *n*.
SAVE *f*	Graba el programa y las variables en un fichero al que da el nombre *f*. Error F si el nombre está vacío o consta de más de diez caracteres. Véase la Parte 20 de este capítulo.
SAVE *f* LINE *m*	Graba el programa y las variables de forma tal que, cuando más tarde se lo cargue, BASIC realizará un salto automático a la línea *m*.
SAVE *f* DATA ()	Graba la matriz numérica.
SAVE *f* DATA $()	Graba la matriz de caracteres.
SAVE *f* CODE *m,n*	Graba *n* bytes a partir de la dirección *m*.
SAVE *f* SCREEN$	Equivale a SAVE *f* CODE 16384,6912. Graba la imagen actual de la pantalla.
SAVE ! *f*	Como SAVE, pero sobre el disco de silicio. Véase Parte 20 de este capítulo.
SPECTRUM	Pasa de 128 BASIC a 48 BASIC, conservando el programa que pueda haber en la RAM. De 48 BASIC no se puede volver a 128 BASIC.
STOP	Detiene el programa y emite el informe 9. CONTINUE reanudará la ejecución del programa a partir de la sentencia siguiente.
VERIFY	Como LOAD, pero sin cargar en la RAM la información leída en la cinta, sino sólo comparándola con la que hay actualmente en la RAM. Error R si la comparación detecta alguna diferncia entre lo grabado en la cinta y lo almacenado en la RAM.

Parte 31
Programas de ejemplo

Programas:

Días
I Ching
Animales
Bandera
Hangman

En esta sección vamos a dar algunos programas de ejemplo que pueden serle de interés. Si desea ejecutar los programas más de una vez, no olvide grabarlos (con SAVE) permanentemente en una cinta, o temporalmente en el disco de silicio.

Días

El primero de estos programas pide una fecha (de este siglo), y después da el día de la semana correspondiente.

```
 10 REM Convertir fecha en día de la semana
 20 DIM d$(7,9): REM Días de la semana
 30 FOR n=1 TO 7: READ d$(n): NEXT n
 40 DIM m(12): REM Días que tiene cada mes
 50 FOR n=1 TO 12: READ m(n): NEXT n
100 REM Captar fecha
110 INPUT "Dia? ";día
120 INPUT "Mes? ";mes
130 INPUT "Año (sólo siglo XX? ";año
140 IF año<1901 THEN PRINT "El siglo XX empieza en el 1901": GO TO 100
150 IF año>2000 THEN PRINT "El siglo XX termina en el 2000": GO TO 100
160 IF mes<1 THEN GO TO 210
170 IF mes>12 THEN GO TO 210
180 IF año/4-INT (año/4)=0 THEN LET m(2)=29: REM Bisiesto
190 IF dia > m(mes) THEN PRINT "Ese mes solo tiene ";m(mes);" dias.": GO TO 500
200 IF dia>0 THEN GO TO 300
210 PRINT "No trates de engañarme.  Dame una fecha real."
220 GO TO 500
300 REM Convertir fecha en número de días desde el principio del siglo
310 LET a=año-1901
320 LET b=365*a+INT (a/4): REM Número de días hasta el principio del año
```

```
330 FOR n=1 TO mes-1: REM Sumar meses anteriores
340 LET b=+m(n): NEXT n
350 LET b=b+dia
400 REM Convertir a día de la semana
410 LET b=b-7* INT (b/7)+1
420 PRINT dia;"/";mes;"/";año
430 PRINT" is ";d$(b)
500 LET m(2)=28: REM Restaurar febrero
510 INPUT "¿Otra vez? ",a$
520 IF a$="n" THEN GO TO 540
530 IF a$<> "N" THEN GO TO 100
1000 REM Días de la semana
1010 DATA "lunes", "martes", "miercoles"
1020 DATA "jueves", "viernes", "sabado", "domingo"
1100 REM Días de los meses
1110 DATA 31, 28, 31, 30, 31, 30
1120 DATA 31, 31, 30, 31, 30, 31
```

I Ching

El siguiente programa lanza monedas para el 'I Ching'. Las formas que produce están «tumbadas», pero los resultados son aceptables:

```
  5 RANDOMIZE
 10 FOR m=1 TO 6: REM Seis lanzamientos
 20 LET c=0: REM Inicializar total a 0
 30 FOR n=1 TO 3: REM Para 3 monedas
 40 LET c=c+2+INT (2*RND)
 50 NEXT n
 60 PRINT"  ";
 70 FOR n=1 TO 2: REM Primero para el hexagrama, segundo para los cambios
 80 PRINT "---";
 90 IF c=7 THEN PRINT "-";
100 IF c=8 THEN PRINT " ";
110 IF c=6 THEN PRINT "X";: LET c=7
120 IF c=9 THEN PRINT "0";: LET c=8
130 PRINT "---  ";
140 NEXT n
150 PRINT
160 INPUT a$
170 NEXT m: NEW
```

Cuando termine de copiar el programa, ejecútelo (RUN) y después pulse cinco veces [INTRO] para obtener los dos hexagramas. Búsquelos en el 'Libro chino de los cambios'. El texto le describirá una situación y las acciones apropiadas para ella, y usted

debe meditar profundamente para encontrar la analogía entre esto y su propia vida. Pulse [INTRO] por una sexta vez y el programa se borrará a sí mismo (para que usted no lo utilice frívolamente).

Muchas personas piensan que los textos están más acertados que si se debieran a mera casualidad; éste puede o no ser el caso con su +2. *En general, los ordenadores son criaturas bastante ateas.*

Animales

El siguiente programa le permitirá enseñarle zoología al +2. Usted piensa en un animal y el ordenador trata de adivinar cuál es haciéndole preguntas que pueden ser contestadas con 'sí' o 'no'. Si la máquina no ha oído hablar nunca de su animal, entonces le pide que escriba una pregunta que pueda utilizar para distinguirlo de los que ya conoce.

```
  5 REM Animales
 10 LET nq=100: REM Número de preguntas y de animales
 15 DIM q$(nq,50): DIM a(nq,2): DIM r$(1)
 20 LET qf=8
 30 FOR n=1 TO qf/2-1
 40 READ q$(n): READ a(n,1): READ a(n,2)
 50 NEXT n
 60 FOR n=n TO qf-1
 70 READ q$(n): NEXT n

100 REM Empezar el juego
110 PRINT "Piensa en un animal.","Pulsa una tecla para continuar."
120 PAUSE 0
130 LET c=1: REM Empezar con la primera pregunta
140 IF a(c,1)=0 THEN GO TO 300
150 LET p$=q$(c): GO SUB 910
160 PRINT "?": GO SUB 1000
170 LET i=1: IF r$="s" THEN GO TO 210
180 IF r$="S" THEN GO TO 210
190 LET i=2: IF e$="n" THEN GO TO 210
200 IF e$<>"N" THEN GO TO 150
210 LET c=a(c,i): GO TO 140

300 REM animal
310 PRINT "¿Estas pensando en"
320 LET p$=q$(c): GO SUB 900: PRINT "?"
330 GO SUB 1000
340 IF r$="s" THEN GO TO 400
350 IF r$="S" THEN GO TO 400
360 IF r$="n" THEN GO TO 500
370 IF r$="N" THEN GO TO 500
380 PRINT"¡Contestame correctamente cuando";"te pregunto!": GO TO 300
```

```
400 REM Acertado
410 PRINT "¡Lo sabia!": GO TO 800
500 REM Otro animal
510 IF qf>nq-1 THEN PRINT "No dudo que tu animal sera muy","interesante, pero
    no me queda","memoria para el.": GO TO 800
520 LET a$(qf)=q$(c): REM Retirar animal antiguo
530 PRINT "Entonces ¿que es?": INPUT q$(qf+10
540 PRINT "Dime con que pregunta distin-","guirias entre"
550 LET p$=q$(qf): GO SUB 900: PRINT " y"
560 LET p$=q$(qf+1): GO SUB 900: PRINT " "
570 INPUT s$: LET b=LEN s$
580 IF s$(b)="?" THEN LET b=b-1
590 LET q$(c)=s$(TO b): REM Insertar pregunta
600 PRINT "¿Cual es la respuesta para"
610 LET p$=q$(qf+1): GO SUB 900: PRINT "?"
620 GO SUB 1000
630 LET i=1: LET io=2: REM Respuestas para animales nuevo y antiguo
640 IF r$="s" THEN GO TO 700
650 IF r$="S" THEN GO TO 700
660 LET i=2: LET io=1
670 IF r$="n" THEN GO TO 700
680 IF r$="N" THEN GO TO 700
690 PRINT "¡Eso no vale!": GO TO 600
700 REM Actualizar respuestas
710 LET a(c,i)=qf+1: LET a(c,io)=qf
720 LET qf=qf+2: REM Siguiente animal libre
730 PRINT "Eso no lo sabia."
800 REM Otra vez?
810 PRINT "¿Quieres jugar otra vez?": GO SUB 1000
820 IF r$="s" TEHN GO TO 100
830 IF r$="S" THEN GO TO 100
900 REM Escribir sin espacios por la derecha
905 PRINT " ";
910 FOR n=50 TO 1 STEP -1
920 IF p$(n)<>" " THEN GO TO 940
930 NEXT n
940 PRINT p$(TO n);: RETURN
1000 REM Captar respuesta
1010 INPUT r$: IF r$="" THEN RETURN
1020 LET r$=r$(1): RETURN
2000 REM Animales iniciales
2010 DATA "¿Vive en el mar",4,2
2020 DATA "¿Vive en la tierra",3,5
2030 DATA "¿Come hormigas",6,7
2040 DATA "una ballena", "un buitre", "un oso", "una hormiga"
```

Bandera

Este programa dibuja la bandera británica:

```
  5 REM Bandera británica
 10 LET r=2: LET w=7: LET b=1
 20 BORDER 0: PAPER b: INK w: CLS
 30 REM Parte inferior de la pantalla en negro
 40 INVERSE 1
 50 FOR n=40 TO 0 STEP -8
 60 PLOT PAPER 0:7,n: DRAW PAPER 0;241,0
 70 NEXT n: INVERSE 0
100 REM Dibujar zonas blancas
105 REM San Jorge
110 FOR n=0 TO 7
120 PLOT 104+n,175: DRAW 0,-35
130 PLOT 151-n,175: DRAW 0,-35
140 PLOT 151-n,48: DRAW 0,35
150 PLOT 104+n,48: DRAW 0,35
160 NEXT n
200 FOR n=0 TO 11
210 PLOT 0,139-n: DRAW 111,0
220 PLOT 255,139-n: DRAW -111,0
230 PLOT 255,84+n: DRAW -111,0
240 PLOT 0,84+n: DRAW 111,0
250 NEXT n
300 REM San Andrés
310 FOR n=0 TO 35
320 PLOT 1+2*n,175-n: DRAW 32,0
330 PLOT 224-2*n,175-n: DRAW 16,0
340 PLOT 254-2*n,48+n: DRAW -32,0
350 PLOT 17+2*n,48+n: DRAW 16,0
360 NEXT n
370 FOR n=0 TO 19
380 PLOT 185+2*n,140+n: DRAW 32,0
390 PLOT 200+2*n,83-n: DRAW 16,0
400 PLOT 39-2*n,83-n: DRAW 32,0
410 PLOT 54-2*n,140+n: DRAW -16,0
420 NEXT n
425 REM Completar
430 FOR n=0 TO 15
440 PLOT 255,160+n: DRAW 2*n-30,0
450 PLOT 0,63-n: DRAW 31-2*n,0
460 NEXT n
470 FOR n=0 TO 7
480 PLOT 0,160+n: DRAW 14-2*n,0
```

```
485 PLOT 255,63-n: DRAW 2*n-15,0
490 NEXT n
500 REM Bandas rojas
510 INVERSE 1
520 REM San Jorge
530 FOR n=96 TO 120 STEP 8
540 PLOT PAPER r;7,n: DRAW PAPER r;241,0
550 NEXT n
560 FOR n=112 TO 136 STEP 8
570 PLOT PAPER r;n,168: DRAW PAPER r;0,-113
580 NEXT n
600 REM San Patricio
610 PLOT PAPER r;170,140: DRAW PAPER r;70,35
620 PLOT PAPER r;179,140: DRAW PAPER r;70,35
630 PLOT PAPER r;199,83: DRAW PAPER r;56,-28
640 PLOT PAPER r;184,83: DRAW PAPER r;70,-35
650 PLOT PAPER r;86,83: DRAW PAPER r;-70,-35
660 PLOT PAPER r;72,83: DRAW PAPER r;-70,-35
670 PLOT PAPER r;56,140: DRAW PAPER r;-56,28
680 PLOT PAPER r;71,140: DRAW PAPER r;-70,35
690 INVERSE 0: PAPER 0: INK 7
```

Ahora puede tratar de dibujar la bandera española o la de su comunidad autónoma. En general las banderas tricolores son bastante fáciles, aunque algunos colores (por ejemplo, el naranja) pueden presentar dificultades. Cuando termine de dibujar una bandera puede grabarla en el disco de silicio con la orden SAVE ! "bandera1" SCREEN$, después dibujar otra bandera diferente y almacenarla con nombre diferente, etc. En el disco de silicio hay espacio para unas diez pantallas diferentes.

Hangman

A continuación le ofrecemos una versión de este clásico juego. Por si no lo conoce, podemos decirle que un jugador introduce una palabra y el otro trata de adivinarla:

```
  5 REM Hangman
 10 REM Preparar pantalla
 20 INK 0: PAPER 7: CLS
 30 LET x=240: GO SUB 1000: REM Dibujar hombre
 40 PLOT 238,128: DRAW 4,0: REM Boca
100 REM Preparar palabra
110 INPUT p$: REM Palabra que hay que adivinar
120 LET b=LEN p$: LET v$=" "
130 FOR n=2 TO b: LET v$=v$+" "
140 NEXT n: REM v$=palabra adivinada hasta ahora
150 LET c=0: LET d=0: REM Contador de intentos y aciertos
```

```
160 FOR n=0 TO b-1
170 PRINT AT 20,n;"-";
180 NEXT n: REM Escribir guiones en vez de letras
200 INPUT "Adivine una letra: ";g$
210 IF g$="" THEN GO TO 200
220 LET g$=g$(1): REM Sólo primera letra
230 PRINT AT 0,c;g$
240 LET c=c+1: LET u$=v$
250 FOR n=1 TO b: REM Actualizar palabra
260 IF p$(n)=g$ THEN LET v$(n)=g$
270 NEXT n
280 PRINT AT 19,0;v$
290 IF v$=p$ THEN GO TO 500: REM Palabra acertada
300 IF v$<>u$ THEN GO TO 200: REM Letra acertada
400 REM Dibujar siguiente trozo del patíbulo
410 IF d=8 THEN GO TO 600: REM Colgarlo
420 LET d=d+1
430 READ x0,y0,x,y
440 PLOT x0,y0: DRAW x,y
450 GO TO 200
500 REM Liberarlo
510 OVER 1: REM Borrar hombre
520 LET x=240: GO SUB 1000
530 PLOT 238,128: DRAW 4,0: REM Boca
540 OVER 0: REM Redibujar hombre
550 LET x=146: GO SUB 1000
560 PLOT 143,129: DRAW 6,0: PI/2: REM Sonrisa
570 GO TO 800
600 REM Colgar hombre
610 OVER 1: REM Borrar suelo
620 PLOT 255,65: DRAW -48,0
630 DRAW 0,-48: REM Abrir trampilla
640 PLOT 238,128: DRAW 4,0: REM Borrar boca
650 REM Mover miembros
655 REM Brazos
660 PLOT 255,117: DRAW -15,-15: DRAW -15,15
670 OVER 0
680 PLOT 236,81: DRAW 4,21: DRAW 4,-21
690 OVER 1: REM Piernas
700 PLOT 255,66: DRAW -15,15: DRAW -15,-15
710 OVER 0
720 PLOT 236,60: DRAW 4,21: DRAW 4,-21
730 PLOT 237,127: DRAW 6,0,-PI/2: REM Triste
740 PRINT AT 19,0;p$
800 INPUT "¿Otra vez? ";a$
810 IF a$="" THEN GO TO 850
```

```
 820 LET a$=a$(1)
 830 IF a$="n" THEN STOP
 840 IF a$(1)="N" THEN STOP
 850 RESTORE: GO TO 5
1000 REM Dibujar hombre en la columna x
1010 REM Cabeza
1020 CIRCLE x,132,8
1030 PLOT x+4,134: PLOT x-4,134: PLOT x,131
1040 REM Cuerpo
1050 PLOT x,123: DRAW 0,-20
1055 PLOT x,101: DRAW 0,-19
1060 REM Piernas
1070 PLOT x-15,66: DRAW 15,15: DRAW 15,-15
1080 REM Brazos
1090 PLOT x-15,117: DRAW 15,-15: DRAW 15,15
1100 RETURN
2000 DATA 120,65,135,0,184,65,0,91
2010 DATA 168,65,16,184,81,16,-16
2020 DATA 184,156,68,0,184,140,16,16
2030 DATA 204,156,-20,-20,240,156,0,-16
```

Parte 32
Binario y hexadecimal

Temas tratados:

Sistemas de numeración
Bits y bytes

Esta sección describe la forma en la que los ordenadores manejan internamente los números, utilizando el sistema binario.

En la mayor parte de los idiomas europeos, los nombres de los números reflejan una relación fundamental con el número diez. En castellano, por ejemplo, solamente son excepcionales, en este sentido, el 'once', el 'doce', el 'trece', el 'catorce' y el 'quince'. A partir de éste la regularidad es manifiesta:

... dieciséis, diecisiete, dieciocho, diecinueve
veinte, veintiuno, veintidós,..., veintinueve
treinta, treinta y uno, treinta y dos,..., treinta y nueve
cuarenta, cuarenta y uno, cuarenta y dos,..., cuarenta y nueve
...

Esto facilita el uso sistemático de los números. La razón por la que el número diez desempeña este papel fundamental es el hecho de que tenemos diez dedos en las manos. Este sistema de numeración se llama *decimal.*

Los ordenadores, en lugar del sistema decimal (de *base* diez), utilizan el *hexadecimal* (abreviadamente, hex), cuya base en el número dieciséis. Ya que en nuestro sistema numérico decimal sólo tenemos diez dígitos, necesitamos otros seis para representar los números en hexadecimal. Las seis cifras adicionales son A, B, C, D, E y F. ¿Y qué viene después de la F? Bien, cuando se nos acaban los dígitos en el sistema decimal, volvemos a empezar por el 0 poniéndole un 1 delante (10); lo mismo ocurre en el sistema hexadecimal: el número que sigue a F es 10 (que es el 16 decimal), el siguiente es 11 (17 decimal), etc.

Veamos una tabla que da la equivalencia entre los dos sistemas:

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4

Decimal	Hexadecimal
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10
17	11
	. . .
25	19
26	1A
27	1B
	. . .
31	1F
32	20
33	21
	. . .
158	9E
159	9F
160	A0
161	A1
	. . .
255	FF
256	100
etcétera.	

Si un número está escrito en notación hexadecimal, se puede evidenciar este hecho escribiendo una 'h' como sufijo. Por ejemplo, el número 256 decimal es 100h en hexadecimal.

Puede usted estar preguntándose qué tiene que ver todo esto con los ordenadores. De hecho, los ordenadores se comportan como si sólo tuvieran dos dígitos, representados por una tensión eléctrica baja ('apagado', 0) y una tensión alta ('encendido', 1). Este sistema de numeración en que sólo se dispone de dos dígitos se denomina *binario*. Los dos dígitos se llaman *bits* (de ***bi**nary **dig**it*, dígito *binario*). Así pues, un bit es un estado que puede tener dos valores: 0 o 1.

La siguiente tabla da las versiones decimal, hexadecimal y binaria de los primeros números:

Decimal	Hex	Binario
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
etcétera.		

Es habitual rellenar los números binarios con ceros, de forma que siempre contengan al menos cuatro bits; por ejemplo, 0000, 0001, 0010, 0011 (que representan los números 0 al 3 en decimal).

La conversión entre binario y hexadecimal es muy fácil (consultando la tabla anterior):

Para convertir un número binario en hexadecimal, divida el binario en grupos de cuatro bits (comenzando por la derecha del número) y convierta cada grupo en su correspondiente dígito hexadecimal. Finalmente, una los dígitos hexadecimales para formar el número hex completo. Por ejemplo, para pasar el número binario 10110100 a hexadecimal, convierta el primer grupo (por la derecha) de cuatro bits (0100) en el hexadecimal 4; después convierta el siguiente grupo (1011) en el hexadecimal B, únalos y obtendrá el número hexadecimal completo, B4h. Si el número binario tiene más de ocho bits, puede continuar convirtiendo cada grupo de cuatro bits en un dígito hex. Por ejemplo, el número binario 11101011110000 corresponde al 3AF0h.

Para convertir un número hexadecimal en binario, transforme cada dígito hex en cuatro bits y a luego una los bits para formar un número binario completo. Por ejemplo, para pasar F3h a binario, convierta primero a 3 en su correspondiente 0011 binario (recuerde que debe rellenar con ceros por la izquierda para que el número binario conste de cuatro

bits); después transforme F en su correspondiente 1111 binario, únalos y tendrá el número binario completo, 11110011.

A pesar de que los ordenadores usan un sistema binario puro, los humanos a menudo escribimos los números que vamos a suministrar al ordenador utilizando la notación hexadecimal; después de todo, el número 3AF0h, por ejemplo, es más fácil de leer y recordar que su equivalente, 0011101011110000, en notación binaria de dieciséis bits.

Los bits que se encuentran dentro del ordenador están agrupados normalmente en bloques de ocho, es decir en *bytes*. Un byte puede representar cualquier número decimal del 0 al 255 (11111111 binario o FFh).

Se puede agrupar dos bytes para formar lo que se llama técnicamente una *palabra*. Una palabra se puede expresar mediante dieciséis bits o cuatro dígitos hex y representa un número decimal del 0 al 65535 (1111111111111111 binario o FFFFh).

Un byte se compone *siempre* de ocho bits, pero la longitud de las palabras varía de un ordenador a otro.

La notación BIN usada en la Parte 14 de este capítulo proporciona un medio de introducir números binarios en el +2. Por ejemplo, BIN 10 representa el 4 decimal, BIN 111 representa el 7 decimal, BIN 11111111 representa el 255 decimal, etc.

En esta notación, después de BIN sólo podemos poner ceros y unos, de forma que el número sólo puede ser un entero no negativo. Por ejemplo, no se puede usar BIN −11 para representar el −3 decimal; lo que sí podemos hacer es escribir −BIN 11. El número tampoco puede ser mayor que el decimal 65535; es decir, no puede constar de más de dieciséis bits. Si rellenamos un número binario con ceros por la izquierda, por ejemplo, BIN 00000001, BIN los ignora y trata el número como si fuera BIN 1.