# DATA BUS

Número 1 Segunda Epoca Octubre 1991

EDITA

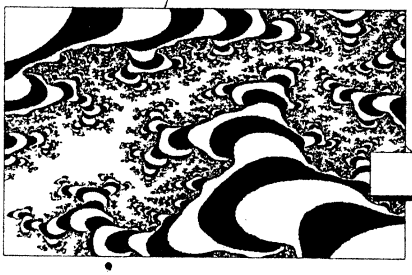
A.J.I.V.

## HISTORIA DE LA INFORMATICA PARTE IV

## ALGORITMOS
## GENERADOR DE HUFFMANN

## MICROPROCESADORES EL Z80 DE ZILOG

## HARDWARE
## CIRCUITOS IMPRESOS



## FRACTALES
## LOS MANDELBROT

# DATA BUS

SEGUNDA EPOCA

NUMERO 1

**DIRECTOR**

JOSE M. SUAREZ POUSA

**REDACTORES**

JESUS CEA

NACHO AGULLO

**COLABORADORES**

TELMO LAGO

PABLO LOBARIÑAS

SIMON GOMEZ

**DISEÑO Y MONTAJE**

JOSE M. SUAREZ POUSA

**EDITA**

A.J.I.V.

ASOCIACION VIGUESA PARA LA INFORMATICA JUVENIL

C/ Caldas de Reis 12-6 Izq.

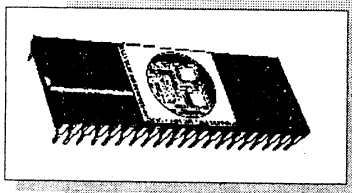
TEL: 23 18 35

41 01 13

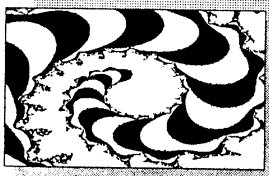
DEPOSITO LEGAL

VG-87-90

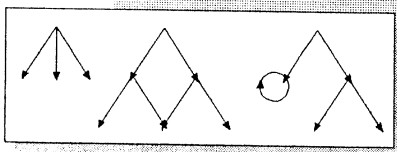

# SUMARIO



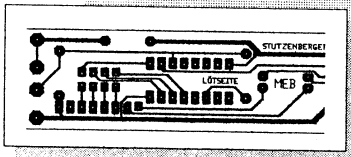












# EDITORIAL

Aquí estamos otra vez, después de este largo verano, con otro número de Data Bus que va evolucionando hacia su diseño definitivo. Esta vez les ofreceremos los habituales temas de la historia de la informática, hardware, fractales y algoritmos, además de una serie sobre los microprocesadores,gracias a los cuales la gran mayoría de nosotros tenemos acceso a los ordenadores.

Estamos preparando para la próxima edición un extenso reportaje sobre el tema de la "Guerra Nuclear", un concurso en el que todos los que tengan acceso a computadoras podrán participar, y cuyos vencedores obtendrán interesantes premios.

Sin más, os emplazamos para la siguiente edición de nuestra revista.

UN SALUDO DE LA REDACCION

## Equipo utilizado para la realización de Data Bus

### HARDWARE

*ATARI 1040 STF*

*ATARI STE* 2.5 Megabytes

Monitores *SM 124* y *Sony Black Trinitron*

*Amiga 2000* 3 Megabytes

Impresora de chorro de tinta *Hewlett Packard Deskjet 500*

Digitalizador de video *VidiST*

Video VHS y Cámara *Sony 8mm*

### SOFTWARE

Programa de autoedición *Calamus*

Procesadores de texto *Le Rédacteur 3.1* y *First Word+ 3.15*

Programa de Digitalización *VidiST*

Programa de Tratamiento Digital de Imagenes *Retouche 1.0*

Programas de Dibujo Vectorial *Outline Art* y *Didot LineArt*

Programa Vectorizador *Avant Vector*

FRACTALES

JESUS CEA

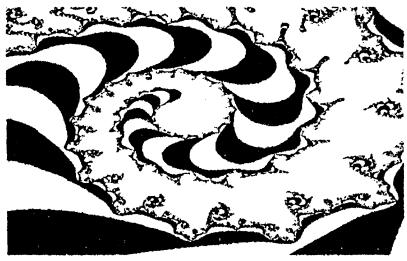
# EL CONJUNTO DE MANDELBROT

***Desde hace unos pocos años existe un gran interés por unas nuevas teorías matemáticas de variadas aplicaciones en la vida real: LOS FRACTALES.***

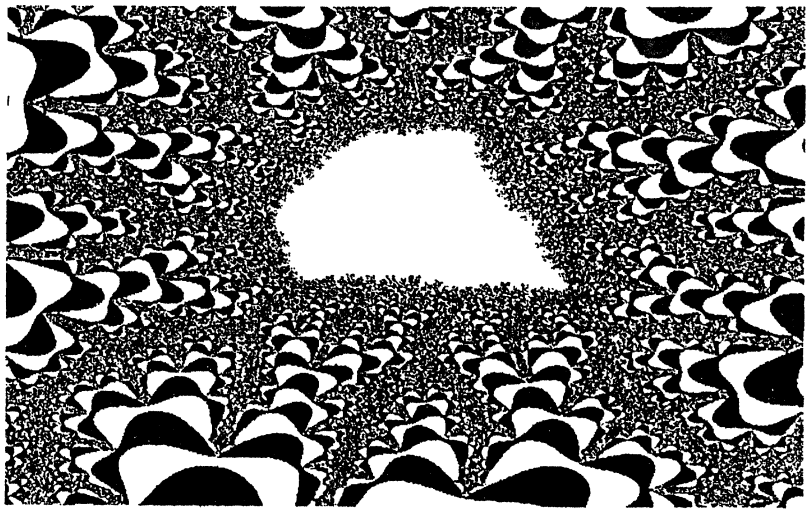
La gran importancia de los fractales está fuera de toda duda, al margen de su elevado valor estético. A lo largo de esta serie de artículos he intentado introduciros en el tema para que así podáis disponer de unos conocimientos, aunque sean someros, sobre los principios que los rigen, sin entrar en profundidad -por ahora- en sus más que notables posibilidades prácticas.

Sin embargo, todo estudio de los diferentes tipos de fractales está incompleto si no incluye al que es, quizás, el más espectacular, impresionante y conocido de todos: *"El conjunto de Mandelbrot"*. El conjunto de Mandelbrot recibe su nombre en honor de Benoît Mandelbrot, matemático del centro de investigación Thomas J. Watson (en New Jersey). Aunque al parecer existen disputas sobre la paternidad de dicha estructura matemática, casi todo el mundo está de acuerdo en que merece conservar su nombre aunque sólo sea porque B. Mandelbrot fue el que dio a conocer este conjunto fractal al gran público.

Pero, ¿qué es lo que hace tan interesante al conjunto de Mandelbrot? Al contrario de lo que ocurría con los demás tipos de fractales que hemos estudiado hasta ahora en los que el resultado final es más o menos predecible (aunque las transformaciones afines pueden sorprendernos fácilmente), en los "fractales de fórmula" el resultado final es tan complejo que no se pueden distinguir pautas precisas. Adicionalmente, el conjunto de Mandelbrot goza de otra propiedad importante y que quizás sea el quiz de la cuestión: En el conjunto de Mandelbrot NO existen estructuras equivalentes. ¿Qué quiere decir esto? Significa que no hay 2 formas idénticas en todo el conjunto, mientras que otros tipos de fractales consisten en un mismo modelo que se va repitiendo a distintas escalas. Existen otros fractales de fórmula pero, en general, presentan simetrías o bisemejanzas. En el hecho de que cada punto del conjunto de



Mandelbrot sea único es lo que lo hace tan atractivo. Podemos encontrarnos desde hermosas espirales de 1, 2, 3, 4, 5,... brazos (cada una única e irrepetible), hasta figuras semejantes a colas de pavo real, pasando por ojos de mosca, agujeros negros de increibles colores en sus bordes, paisajes fractales (con sus rios, sus desiertos, sus mares, sus montañas...),... Todo lo que puedas imaginar y mucho más. Esta infinita variedad



es lo que hace tan interesante este conjunto, por lo demás desprovisto de toda utilidad que no sea estética (aunque parece que hay científicos que lo relacionan con los estados REALES de transición orden-caos, y que lo usan como "mapa" para describir los estados de interfase líquido-gas, y lo mejor es que parece que funciona...).

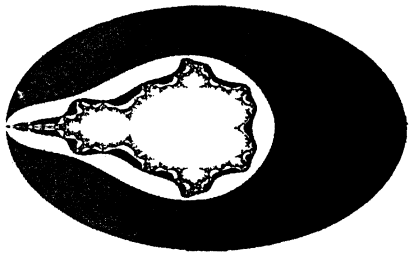
¿Qué es realmente el conjunto de Mandelbrot? No es ni más ni menos que el resultado de iterar la fórmula $Z=Z^2+C$, con Z y C números complejos. Mediante esta fórmula los puntos del plano complejo se dividen en dos categorías: Los divergentes y los acotados. Los divergentes son aquellos cuyas iteraciones los llevan al infinito. Los acotados permanecen en las proximidades del origen por muchas iteraciones a las que se los someta. Estos últimos puntos son los que forman el conjunto de Mandelbrot. A medio camino entre uno y otro tipo se encuentran unos puntos de divergencia muy lenta. Se trata de la *frontera*, lugar donde habitan las mas sorprendentes criaturas de ese universo fractal. En realidad el interés se centra, no en el conjunto de Mandelbrot propiamente dicho, sino en su frontera, verdadera maravilla matemática facilmente cartografiable utilizando nuestros ordenadores.

Para representar el conjunto de Mandelbrot es imprescindible el empleo de una computadora por la simple y llana (y dura) razón de que se requieren **MILLONES** de operaciones matemáticas para representar una simple pantalla. El ordenador que se utilice es lo de menos, aunque influirá enormemente en la velocidad y la calidad final obtenida.

Para saber si la iteración de un punto tiende o no hacia el infinito se utiliza un teorema matemático que dice que si la iteración de cualquier punto se "sale" de la circunferencia de lado 2 y centrada en el origen (es decir, que el módulo sea mayor que dos), es irremediable que el punto tienda hacia el infinito, por lo que será divergente. Para no esperar eternamente hacemos que sobre cada punto se efectúen un número máximo de iteraciones X. Si se alcanza ese máximo y el punto todavía no ha sido declarado "prófugo" entonces lo consideraremos perteneciente al conjunto. Esta aproximación provoca pequeñas imprecisiones facilmente subsanables aumentando el número de iteraciones, pero a costa de unos tiempos de cómputo mucho más dilatados. Para que el resultado final sea más interesante podemos asignar colores distintos según la "velocidad de escape". Así, si utilizamos 15 iteraciones de máximo, asignamos el negro a las que se salen con 0 iter. (las que tienen ya en un principio un

módulo mayor o igual a 2), azul a las de 1, azul claro a las de 2, verdes, rojos, amarillos, etc. Por convenio los puntos pertenecientes al conjunto (los considerados acotados) se pintan de negro.

La forma de calcular los puntos a iterar se hace de la siguiente manera: Primero el programa pide la posición (x,y) del que será el centro de nuestro dibujo. La primera coordenada es la real y la segunda la imaginaria (recordad que se trata de números complejos). Luego viene el tamaño de la ventana de dibujo (*l*). Aquí cada uno escoge el significado preciso de "*l*". Mi programa ***FRACTALS*** calcula los puntos



incritos en el cuadrado de vértices (x-l,y-l), (x+l,y-l), (x+l,y+l), (x-l,y+l), con lo que el tamaño real de la ventana es 2*l. Otros programas podían utilizar realmente "l" como el tamaño de la ventana, por los que el cuadrado de cálculo tendría los vertices en (x-l/2,y-l/2), etc. Bien, ahora necesitamos calcular los puntos comprendidos en nuestro cuadrado. Para ello se utilizan 2 bucles anidados, uno para la parte real y otro para la parte imaginaria, que hagan corresponder a cada píxel de la pantalla una coordenada compleja. Ahora sólo queda iterar cada punto.

A la variable "*C*" se le asigna la posicion de dicho punto y la "*Z*" se pone a 0. El porqué de ello lo explicaré más adelante. Ya podemos ejecutar la fórmula de iteración, pero teniendo muy en cuenta que tanto "*C*" como "*Z*" son complejos y tendremos que operar separadamente con las partes reales y las imaginarias. En cada paso de la iteración comprobamos si el módulo de "*Z*" es mayor o igual a 2. Si es así, entonces el punto diverge y lo coloreamos del color que le corresponda según el número de iteraciones efectuadas. Si se llega al máximo de iteraciones sin superar un módulo de dos, entonces el punto se considera acotado y debe imprimirse en negro (por el convenio). Es interesante que el programa pregunte al principio el número máximo de iteraciones a realizar, así como el tamaño final del dibujo, para no tener que esperar demasiado si sólo queremos ver que tal queda (calcular, por ejemplo, 80*50 puntos en vez de la pantalla entera).

Con este procedimiento tan sencillo (una docena de líneas de programa), se obtienen una imágenes francamente impresionantes, como podéis constatar a través de los dibujos que acompañan este artículo y que también adornaron algunas páginas, y portadas, de número anteriores (lástima que estén en blanco y negro). Estas pantallas han sido calculadas utilizando el ya comentado programa ***FRACTALS***, de producción propia, con rutinas de cálculo en lenguaje ensamblador para reducir el tiempo de espera.

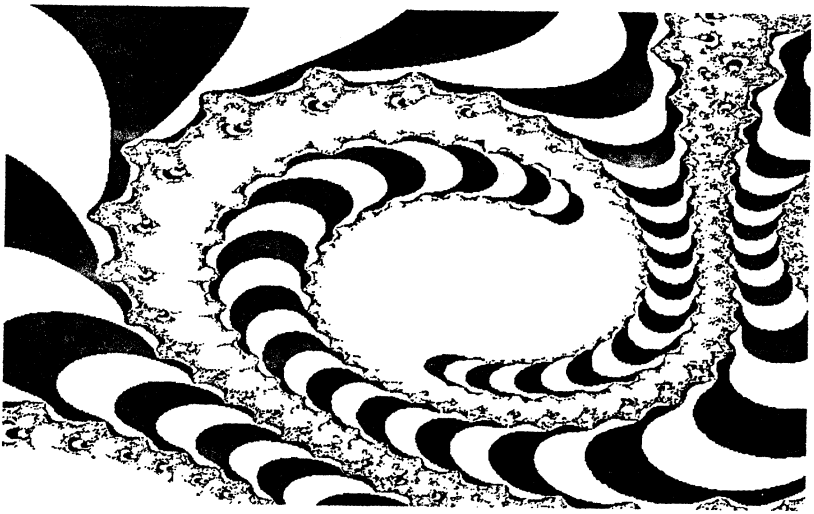
El programa funciona sobre un ATARI ST(E) con, al menos, 128Kbytes libres, aunque se precisa más memoria para poder explotar todas sus posibilidades a fondo. Funciona en las 3 resoluciones gráficas básicas del Atari, y permite grabar las pantallas generadas, crear vistas tridimensionales, y un sin fin de opciones más. La velocidad de cálculo es bastante alta, calculando la visión general del Mandelbrot en 320*200 puntos, 32 iteraciones y 16 colores en 1 minuto y 10 segundos. El programa permite, adicionalmente, calcular el conjunto de Julia asociado a cada uno de los infinitos puntos del conjunto de Mandelbrot, realizar las ampliaciones utilizando el "ratón", 3 grupos de rutinas matemáticas de diferentes precisiones y velocidades, etc. En este momento la última versión es la V2.2r. El programa es "*SHAREWARE*", por lo que no habría ningún problema en facilitar una copia a todo aquel que lo solicite. Los interesados pueden

ponerse en contacto conmigo a través de la dirección o el teléfono de la asociación, que aparecen en la página dos. Los interesados en escribir sus propios programas visualizadores también pueden contactar conmigo para conseguir más información sobre este tema, posiciones interesantes, etc. (nunca está de más hacerse un poco de publicidad...)

Volviendo al tema en sí, decir que al margen de la gran belleza de las formas obtenidas, el conjunto de Mandelbrot se ofrece también al análisis matemático. Así, existen teoremas y demostraciones que aceleran los cálculos y que prueban matematicamente resultados obtenidos de forma empírica. Sobre estas facetas matemáticas no existe información alguna, salvo meras referencias, lo que no ayuda en absoluto al profano que quiere introducirse en este mundo comprendiendo plenamente

hechas, pero no hay que olvidar que todo este trabajo lo desarrollé durante los meses de mayo y junio de 1.990, cuando no era más que un simple alumno de C.O.U., con unos conocimientos matemáticos bastante más reducidos que los actuales, y cuya única comprensión sobre números complejos venía de 2º de B.U.P. ...

Los interesados en conseguir dichas demostraciones pueden ponerse en contacto conmigo siguiendo el mismo procedimiento que ya expliqué antes. Estas demostraciones incluyen la razón de que el círculo que marca la divergencia tenga radio dos (en realidad, cada punto tiene un radio del círculo DISTINTO, pero se toma el dos por ser el supremo de todos ellos, y así cortar a todos "por el mismo patrón"), la asimetría del Mandelbrot (en una visión general el Mandelbrot parece simétrico respecto al eje real, pero cuanto más



sus bases formales. En esta situación estaba cuando decidí a probar por mis propios medios algunas de las demostraciones que la ya de por sí "escueta" información disponible se saltaba a la torera. Quizás hoy en día parezcan demostraciones en algunos casos triviales o, incluso, mal

ampliamos, la asimetría se nos presenta de forma más exagerada), etc. Lo siento mucho pero me parece la única forma de que colaboréis un poco...

Anteriormente comenté que al principio de las iteraciones la variable "Z" debía ser

inicializada a 0. El porqué de ello es relativamente simple: En realidad el conjunto de Mandelbrot no es más que un corte bidimensional (bidimensional en dimensiones reales, pero unidimensional en dimensiones complejas) de un hipercuerpo tetradimensional (o bidimensional si consideramos dimensiones complejas). La variable "*Z*" apunta a los diferentes "cortes" posibles de este superobjeto y, modificando este valor, podemos ir observando distintos Mandelbrots alternativos a cada cual más interesante. El conjunto de Mandelbrot por autonomasia está situado en el centro del hipercuerpo, el (0,0) complejo, de ahí que se inicialice "*Z*" con el valor 0. Los lectores avispados se habrán dado cuenta de que de un objeto de cuatro dimendiones reales se pueden obtener dos familias de "rebanadas" bidimensionales perpendiculares entre sí. Una de estas familias está constituida por los "Mandelbrot", ¿Cuál es la otra? Se trata de los conjuntos de Julia.

Los conjuntos de Julia son visiones de nuestro hipercuerpo, igual que los Mandelbrots, pero vistos "desde otro ángulo". En principio podemos considerar que cada punto del conjunto de Mandelbrot tiene un conjunto de Julia asociado, que representa el "*atractor*" correspondiente a dicho punto. Por decirlo de alguna forma, el conjunto Julia asociado a cualquier punto del conjunto de Mandelbrot constituye un "*mapa*" de la evolución de dicho punto al ir variando la posición del Mandelbrot (la famosa "*Z*"). Es decir, que si la posición (0.17,0.23) de un Julia asociado a un punto del conjunto de Mandelbrot es amarilla, entonces ese punto (el punto del cual es asociado el Julia) será amarillo si calculamos el Mandelbrot localizado en la posición (0.17,0.23), etc. Como puede verse, los Julia y los Mandelbrots está intimamente relacionados: los Julia representan la evolución de cada punto del Mandelbrot al ir variando la "*Z*", mientras que el Mandelbrot canónico cataloga los infinitos Julias según su conexidad o inconexidad (si un punto pertenece al conjunto de Mandelbrot, su Julia asociado es conexo, y si no pertenece, es inconexo). En realidad todo esto es bastante sencillo si pensáis en dimensiones complejas y no en dimensiones reales, dado que para generalizar conceptos a espacios n-dimensionales hace falta bastante imaginación.

Como curiosidad, he probado matemáticamente que TODOS los conjuntos de Julia son simétricos respecto a la diagonal del primer y tercer cuadrante, lo que significa que nuestro hipercuerpo tetradimensional es simétrico también (al menos según las "rebanadas" con las que tomamos los Mandelbrots). Es decir, que los Mandelbrots localizados en (x,y) y en (-x,-y) son EXACTAMENTE IGUALES. Otro dato: la existencia de este hipercuerpo la deduje "a posteriori" estudiando los resultados de las ecuaciones, y tras calcular decenas de pantallas de Mandelbrots y Julias. Personalmente, me parece una explicación perfectamente plausible. Si alguien no está de acuerdo conmigo que me lo diga.

*Bien, con este artículo termina la serie sobre los diferentes tipos de fractales. Espero haberos ayudado a introduciros con buen pié en este apasionante mundillo. Por mi parte nada más, sólo desearos una fecunda investigación a los que haya conseguido interesar por el tema. Y a los que no, que echen otro vistazo a los dibujos de los últimos números y seguro que les pica el gusanillo...*

HARDWARE

TELMO LAGO

# DISEÑO DE CIRCUITOS IMPRESOS

***Muchas veces hubiéramos querido construirnos ese interface, controlador,o simplemente soporte para alguna cosilla electrónica que hayamos diseñado, pero siempre topamos con un problema: ¿dónde o cómo hacer la base donde irán los componentes? La respuesta a la primera pregunta es un tanto sencilla: en nuestra casita. La segunda respuesta es lo que va a intentar abordar este artículo.***

Las placas están compuestas de una base de material plástico (suele ser baquelita, fibra de vidrio o poliester) con una o dos caras de cobre muy finas, que sirven de conductores. El diseño de la placa consiste en hacer las siluetas de las pistas que irán en la placa. En el mercado se pueden encontrar dos tipos de placas: unas, llamadas "Uni-print", son parecidas a las antiguas regletas de terminales, con siluetas y agujeros ya predefinidos. La principal ventaja es que no hace falta una preparación previa, pero el diseño del circuito se hace muy difícultoso debido a la gran cantidad de "puentes" que hay que hacer, además suelen ser bastante caras. El otro tipo de placas son las que estan "en bruto", que se pueden siluetear de la forma que se desee, pero hay que prepararlas previamente, en lo cual vamos a profundizar en seguida. Decir también que su precio es aceptable (una de doble cara de 200x120 mm cuesta alrededor de 500 pts.

Adentrémonos pues en la preparación de este tipo de placas. Como ya mencioné anteriormente, existen placas de una o dos caras. Basicamente, el proceso de realización es el mismo para los dos tipos. Sólo advertir que en las de dos caras hay que tener en cuenta que a ambos lados de la placa tienen que coincidir los agujeros para los terminales (y además ser los correctos).

Después de esta breve introducción vamos a hablar ya de como se hace. Puede hacerse de dos formas: una "casera" y otra fotografica. Lo primero es fijar el tamaño de la placa. Para ello hay que diseñar la silueta del circuito (las pistas). Un método sencillo para llevarlo a cabo puede ser:

1) Trazar dos lineas paralelas, una para + y otra para -. El circuito se pone entre ellas.

2) Se hacen partir de las lineas todos los componentes unidos a ellas.

3) Se pone el espacio que ocupen

esos componentes, no olvidando la polaridad.

4) A partir de cada componente se trazan tantas lineas como componentes estén unidos a él.

5) Se hace lo descrito en 3) hasta que todos los componentes estén unidos según el esquema eléctrico.

6) Darle forma "artistica" con ángulos, ajustando tamaños, etc., hasta obtener el diseño final.

La parte anterior hay que ponerla en práctica sea cual sea el método a utilizar. La fase siguiente es pasar el dibujo del papel a la placa.

## *TECNICA "CASERA"*

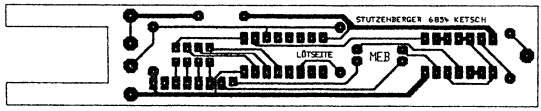
El marcaje de las pistas se hace a lápiz, para pasar después algo que resalte y aguante el ataque de los reveladores. Lo más utilizado es:

*Laca de uñas:* La solución más simple. Pueden usarse incluso si está medio seco. La principal desventaja es que es bastante difícil hacer un trabajo fino.

*Rotuladores:* Lo más recomendable. Los resultados son aceptables y el precio bastante bajo. El tipo adecuado es el Eding 3000, que marca pero no es absorbido totalmente.

*Plantillas:* Bueno, pero caro. Son similares a los "calquitos", que se pasa un lápiz o similar por encima de ellas y las lineas quedan impresas en la placa. Es necesario un poco de experiencia para conseguir buenos resultados.

Proseguimos con el proceso de revelado. Para ello se usan ácidos con poder oxidante. En la placa, el cobre está en oxidación cero. Si el estado de oxidación es positivo, el cobre se disuelve, y esto es lo que hace el revelador. Cuando se marcaron las pistas, se taparon las zonas que tienen que




quedar (las pistas), y se disuelve el cobre innecesario.

El agente revelador utilizado mas comunmente es el cloruro férrico. Es un sólido de color amarillento. Tiende a captar agua del ambiente, por lo que a veces puede tener un color un poco mas anaranjado, pero sirve igual. Como se usa: se coge un recipente ***NO METALICO*** (el cloruro férrico actuaría sobre él) del tamaño de la placa (vale un plato de cerámica). En un vaso de cristal se echa agua caliente como para llenar el plato. Se echa cloruro férrico hasta que ya no se disuelve más. Se coloca la placa en el plato y se deja un rato. Si el líquido va tomando un color verdoso, es que la cosa va bien. Cuando sólo se vea lo que se usó para marcar, se quita la placa, se lava con agua corriente y se seca.

Otros ácidos que se pueden utilizar (el resultado práctico es que se tarda menos que con el cloruro, pero el índice de peligrosidad es bastante mayor) son el clorhídrico, sulfúrico, nítrico, mezcla clor-hídrico y HO2, etc.

El último paso es eliminar los restos de lo que se haya utilizado para cubrir las pistas (el rotulador, los calquitos, o lo que fuera). Puede hacerse con alcohol o cualquier disolvente normal y un algodón.

Estas placas son especiales, y hay que conseguirlas en sitios un tanto especializados (en una tienda de electrónica medianamente decente las hay). Dentro de este tipo las hay en "positivo", o "directas" y en "negativo", o inversas. Actúan según si lo que se sensibiliza es la zona expuesta a la luz o no. En las positivas se sensibiliza las zonas expuesta a la luz, de forma que lo que se tapa es lo que interesa que quede (las pistas). En las negativas es todo lo contrario, ya que la zona expuesta no reacciona al revelador. De ello se deduce que en este tipo de placas lo que hay que hacer es la imagen en negativo de lo que sería el circuito. Reseñar que las placas fotosensibilizadas precisan de las mismas precauciones que las habituales películas fotográficas. De hecho, las placas vienen en bolsas negras para preservarlas del efecto de la luz, y para su manejo es necesario trabajar con luz roja.

Puede haber placas que reaccionan a la luz visible y a la ultravioleta. Para estas últimas es necesario tener una lámpara especial. Por experiencia personal, puedo decir que las que dicen sensibilizadas a la luz visible son un fiasco. Precisan de una dosis de UVA lo suficientemente elevada como para ser necesario usar igual la lámpara.

Para pasar el circuito a la placa se dibuja el circuito en un trozo de papel vegetal (lo ideal es papel para transparencias) y se pone sobre la placa, fijado firmemente a los bordes, y se procede a dejar todo bajo la luz durante cierto tiempo. Seguidamente se procede a revelar la placa (suele hacerse con productos especiales que se pueden conseguir donde se compró la placa), y posteriormente se pasa por ácido para eliminar el cobre sobrante. Por último se lava con agua corriente.

Por ultimo, queda hacer los agujeros donde se soldarán los terminales de los componentes, y si se quiere, barnizar (con producto especial) las pistas para proteger la placa de agentes externos.

*Nada más, esperando de que os haya gustado este artículo, os emplazo para el siguiente número en el cuál hablaremos más de hardware.*

TECNICA

NACHO AGULLO

# EL Z80 DE ZILOG

***Esta nueva sección tratará sobre los chips que controlan nuestros microordenadores. Mientras que en los grandes ordenadores los procesadores están constituidos por varios chips - incluso varias placas -, en los microordenadores todas esas funciones están concentradas en el microprocesador, siendo de gran utilidad conocerlo a fondo, por su versatilidad y posibilidades.***

Los primeros microprocesadores fueron creados casi accidentalmente. Fue Datapoint quien, en 1972, encargó a Intel un chip capaz de reemplazar los numerosos chips TTL empleados por las terminales de ordenador de la época. Intel desarrolló el 4004 y el 8008, de 4 y 8 bits respectivamente, que respondían a las características deseadas por Datapoint excepto en la velocidad: eran demasiado lentos.

No obstante este fracaso inicial, los microprocesadores no desaparecieron. Tras el rechazo de Datapoint, ingenieros y aficionados a la electrónica descubrieron el potencial del 8008 como Unidad Central de Proceso para un ordenador y lo utilizaron para diseñar y construir pequeños ordenadores de mesa, en contraste con las voluminosas consolas y unidades del momento.

Pronto se hicieron evidentes las limitaciones del 8008; sin embargo, los prometedores comienzos en el terreno de los microordenadores decidieron a Intel a producir un chip más potente; así nació el 8080. Al propio tiempo, la empresa rival Motorola decidía entrar en conpetición produciendo su propio microprocesador: el 6800. Ambos chips, muy diferentes, eran igualmente potentes y adecuados para servir de base al diseño de un microordenador.

La competencia por desarrollar un microprocesador más potente y rápido que se estableció desde entonces no ha parado hasta la fecha, produciendo chips cada vez mejores: 6502, 8086, 68000...

## Aparición del Z80

En 1974, la industria de los microordenadores va a experimentar un significativo ade-lanto. La aparición del sistema operativo CP/M, diseñado por Gary Kildall y John Torode para los ordenadores basados en el 8080, va a representar una gran simplificación en el tratamiento de ficheros: este sistema operativo estaba preparado para manejar las recién aparecidas unidades de disco flexible. El sistema, creado en principio para Intel, fué rechazado por ésta, por lo que sus autores decidieron distribuirlo por sí mismos.

El CP/M no sólo tuvo éxito, sino que se convirtió en el sistema predominante de manipulación de discos para ordenador por muchos años, y su éxito corrió paralelo al de los microordenadores basados en el 8080. El 6800 quedaba atrás, mientras despunta-

ba el alba de la era del ordenador personal.

Fué en estas circunstancias cuando Frederico Faggin y Masatoshi Shima, empleados de Intel que habían participado en el desarrollo del 8080, se independizaron para formar su propia empresa: Zilog.

Mediante la utilización de códigos de operación de dos bytes y otras técnicas, Faggin y Shima consiguieron ampliar el juego de instrucciones del 8080, con lo que se conseguía un rendimiento mucho mayor, manteniendo al mismo tiempo una compatibilidad total con el software ya existente, incluyendo - cómo no - el CP/M. Además, integraron todo el conjunto del microprocesador en un sólo chip (el 8080, a pesar de describirse como "un ordenador en una sola pastilla", necesitaba, aún en su configuración mínima, de dos chips auxiliares: el 8224, excitador de reloj; y el 8228, excitador de buses y controlador).

Fué así como nació el Z80, en 1977. Muchos ordenadores han aparecido desde entonces basados en él, alcanzando ventas de millones y millones de unidades.

## ¿Qué es un microprocesador?

Se define microprocesador cómo «*circuito integrado capaz de ejecutar programas, operar datos, y controlar las unidades de comunicación con el exterior y la memoria dónde se almacenan los da tos*»

Las funciones de un microprocesador son:

- Controlar el flujo de información.
- Operar los datos.
- Gestionar la memoria.
- Gobernar toda actividad del ordenador de acuerdo a las instrucciones recibidas.

## El microprocesador en el ordenador

Seguidamente veremos como se relaciona el microprocesador con los restantes componentes del ordenador.

En principio, podemos englobar estos componentes en tres categorías en función de como se relacionan con el microprocesador:

a) La memoria interna.

b) Los dispositivos de entrada.

c) Los dispositivos de salida.

El microprocesador se va a comunicar

con estos componentes a través de los *buses*, que son conjuntos de varias pistas que permiten mandar tantas señales binarias (utilizando la tensión para simbolizar los unos, y la ausencia de la misma para simbolizar los ceros) como pistas posee el bus. De hecho, al hablar del bus de datos, por ejemplo, se dice por extensión que tiene ocho, dieciséis, treinta y dos *bits*, en vez de *pistas*.

Los buses que podemos encontrar en un ordenador son tres: bus de datos, bus de direcciones y bus de control. Usualmente se identifica la cantidad de bits que puede mandar de una sola vez el bus de datos con la capacidad del microprocesador, hablándose así de microprocesadores de ocho, dieciséis,

treinta y dos bits... según los buses de datos correspondientes.

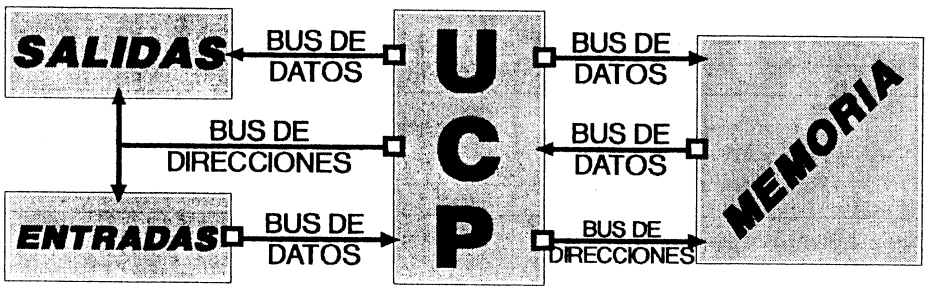
En la próxima descripción de la comunicación del microprocesador con los demás dispositivos del ordenador, hablaremos de la función de los buses de datos y direcciones; en cuanto al bus de control, se define como el conjunto de pistas a través de las cuales el microprocesador controla a las unidades complementarias.

La operación a) se desarrolla del siguiente modo: el ordenador envía por el bus de direcciones la dirección de la celda de memoria cuyo valor se desea sustituir, y simultáneamente envía por el bus de datos el valor nuevo. Ambas señales confluyen en la memoria, realizándose la operación.

La operación b) se desarrolla de forma similar. El ordenador envía a través del bus de direcciones la dirección de la celda de memoria cuyo valor se desea tomar, pero no envía nada por el bus de datos. Al llegar la señal del bus de direcciones a la memoria, la celda correspondiente es excitada y su contenido se copia en el bus de datos, yendo a parar al microprocesador.

**Figura 1.**

*Estructura Interna de un Ordenador con los Buses de Datos y de Direcciones.*




## Relación microprocesador y memoria

Las operaciones que el microprocesador realiza con relación a la memoria se pueden agrupar en dos tipos:

a) Sustituir el valor de una celda de memoria por otro.

b) Tomar el valor de una celda de memoria.

A continuación veremos el modo en que se desarrollan estas operaciones. Tomaremos como ejemplo tipo un ordenador con buses de datos y de direcciones de 8 y 16 bits respectivamente.

## Relación microprocesador y dispositivos entrada

Los dispositivos de entrada de datos son todos aquellos que proporcionan datos al ordenador, sin requerir que éste les suministre

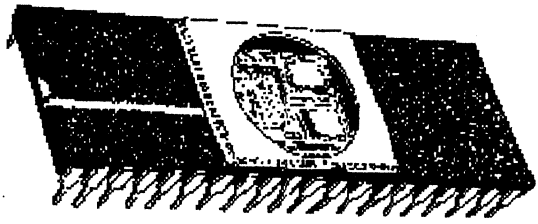
a su vez datos. Ejemplos son el teclado, el joystick y la tableta gráfica.

Tomando como ejemplo un ordenador con soporte para doscientos cincuenta y seis puertos de entrada diferentes, se requiere un bus de direcciones de al menos 8 bits. La operación de entrada de datos se realizaría entonces de la siguiente manera: El microprocesador envía a través del bus de direcciones el número del puerto de entrada que quiere comprobar. La señal llega al puerto de entrada, excitándolo y provocando que el valor que entra a través del mismo salte al bus de datos, yendo a parar al ordenador.

## Relación mcroprocesador y dispositivos de salida

Los dispositivos de salida de datos son todos aquellos que toman los datos que el ordenador les suministra para ejecutar una tarea, sin suministrar a su vez datos al ordenador. Ejemplos son el chip de vídeo, la impresora, y el chip de sonido.

Tomando como ejemplo, al igual que para los dispositivos de entrada, un ordenador con soporte para doscientos cincuenta y seis puertos de salida diferentes y un bus de direcciones de al menos 8 bits, el procedimiento es el siguiente: El microprocesador envía a través del bus de direcciones el número del puerto de salida a través del cuál quiere enviar el dato, y este se envía a su vez a través del bus de datos. Ambas señales confluyen en el puerto correspondiente, saliendo el dato a través de él.



## Dispositivos de entrada-salida

Son aquellos que proporcionan datos al ordenador, requiriendo además que éste a su vez les suministre datos. Un ejemplo lo constituye una unidad de almacenamiento de datos.

A la hora de explicar su relación con el microprocesador, basta considerarlos dispositivos de entrada mientras envían datos al microprocesador, y dispositivos de salida mientras reciben datos del mismo.

## Configuraciones dispositivos / buses

Con todo lo visto anteriormente, trataremos de hacernos una idea de la configuración de un ordenador, combinando los dispositivos ya vistos y uniéndolos mediante los buses.

En principio, visto lo anterior y realizándolo paso por paso, obtendríamos una configuración como la de la figura 1, para un ordenador tipo de 8 bits, con soporte para doscientos cincuenta y seis puertos de salida.

No obstante, podemos observar que las características que requieren los dos buses de datos para meter y sacar datos de memoria son idénticas, pudiéndolos sustituir por un sólo bus de datos que realice ambas funciones. Se trata pues de un bus de datos de dos sentidos: las señales fluyen del microprocesador a la memoria o a la inversa, según lo decida el microprocesador. Este tipo de bus se conoce como bidireccional.

Observando que lo mismo es aplicable a los dispositivos de entrada y de salida, podemos practicar la misma simplificación con respecto a ellos.

Por último, considerando que los dispositivos de entrada y salida pueden tomar sólo ocho de los dieciséis bits que transmite el bus de direcciones para conocer el puerto al que se refiere la operación en curso, podemos hacer que se alimenten del mismo bus de direcciones que la memoria.

Así, el resultado es la configuración de la figura 2, que es típica de muchos ordenadores con procesador de 8 bits y que representa una gran simplificación con respecto a la figura 1, gracias a la utilización de buses bidireccionales.

Y así termina esta primera parte de la serie. En la próxima parte veremos las componentes del Z80 (que hay en su interior) y los datos técnicos concretos.

## Bibliografía


- *Gran Enciclopedia Informática.* Ediciones Nueva Lente.

- *Enciclopedia Mi Computer.* Editorial Delta.

- *Construya su propia computadora basada en el Z80.* Steve Ciarcia. Bytes books. Editorial McGraw Hill.


Algunos ordenadores basados en el Z80:

*Sinclair ZX80, ZX81*

*ZX Spectrum*

*Tatung Einstein*

*Osborne 1*

*Memotech MTX512 y RS128*

*Spectrvídeo 318/328*

*Sega SC3000H*

*Research Machines 380Z*

*Link 480Z*

*Linx*

*Colour Genie*

*AMSTRAD CPC 464/6128*

*Júpiter Ace*

*Yamaha CX5M*

*Toshiba H10*

*Sord M5*

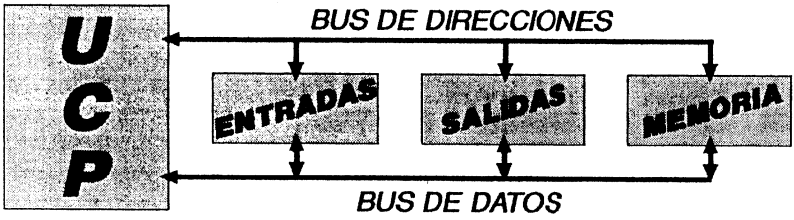
*Sharp MZ 711*

*Coleco Adam*

*Enterprise 64.*

Con posterioridad apareció el Z80B, sobre el que se han diseñado ordenadores como el Sam Coupé y el Wren Executive.

**FIGURA 2.** *Estructura Interna de un Ordenador con Buses Bidireccionales.*



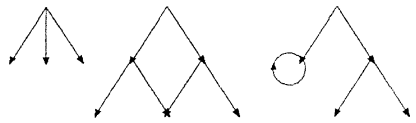
ALGORITMOS

JESUS CEA

# ALGORITMO GENERADOR DE HUFFMANN

***En el número anterior hice varias referencias al Algoritmo Generador de Huffmann para crear árboles binarios. Lo siento, pero no dispuse de suficiente espacio físico para poder incluirlo en el mismo. Espero que el retraso no haya perjudicado a nadie.***

**Figura 1:**



*Arboles no expresables mediante el algoritmo generador de Huffmann. El primer árbol puede generarse realizando una pequeña modificación al método.*

Como su nombre indica, el Algoritmo Generador de Huffmann sirve para crear árboles binarios óptimos (luego veremos que esto no es del todo cierto) para aquellas aplicaciones que los utilicen. Esto incluye desde sistemas de compresión, clasificación y de búsqueda, hasta optimización de recursos, maximización de beneficios, etc. Aunque este algoritmo es facilmente generalizable a árboles n-dimensionales vamos a reducir nuestro estudio a árboles binarios simples.

Para que una estructura arborescente sea realizable utilizando Huffmann se necesita que sea expresable mediante un árbol binario simple. Es decir, que a cada nudo llegue una única rama y que partan de él 0, 1 o 2 ramas (2 ramas como máximo). Esto quiere decir que los árboles de la Figura 1 no pueden ser creados utilizando este algoritmo sin más, lo cual no significa que el método no sea facilmente ampliable para incluir los árboles de cuyos nudos partan más de dos ramas siempre y cuando **SOLO** llegue una rama a cada nudo (árboles n-arios simples). En el caso de que a cada nudo pueda llegar más de una rama, el algoritmo de Huffmann no se puede utilizar. En contra de lo que os pueda parecer este tipo de árboles (árboles n-arios complejos) es bastante frecuente: Toma de decisiones (inteligencia artificial), sistemas expertos, clasificaciones múltiples, optimización de tiempos de acceso en redes de ordenadores, etc. Existen formas de aplicar Huffmann en estos casos, pero a costa de sacrificar algunas ventajas del algoritmo. En casos como estos os recomiendo utilizar otros algoritmos "más apropiados". Como ya he dicho, aquí sólo vamos a estudiar los casos de árboles binarios simple.

Para ver como se crea un árbol binario siguiendo el método Huffmann vamos a utilizar un ejemplo de gran utilidad: La compresión de textos (ver número anterior). Para simplificar, vamos a comprimir sólo las 26 letras. (ni la eñe, ni los acentos, ni las mayúsculas, ni puntuación, ni nada...). En la Figura 2 podéis ver el número de apariciones de estas 26 letras en mi artículo de curiosidades informáticas de este mes. Lo que nosotros deseamos obtener es que los caracteres más utilizados aparezcan más "al principio" para que tengan una expresión más corta. En la Figura 3 podéis observar el árbol óptimo final obtenido. ¿Cómo he obtenido dicho árbol? Muy fácil.

Si lo que queremos es que los caracteres más frecuentes aparezcan al princi-

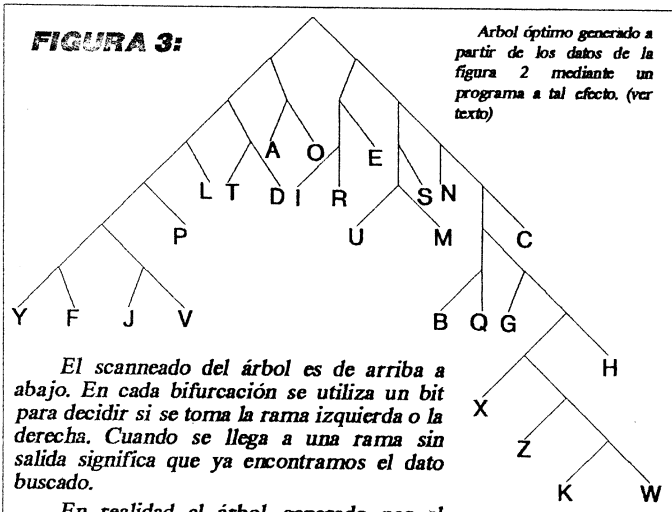
pio, es evidente que estamos relegando a los caracteres menos utilizados a los últimos puestos. Podemos aprovechar esta aparente trivialidad para empezar a generar nuestro árbol. Para ello cogemos los dos caracteres menos utilizados y creamos un nudo que llamaremos *Nudo1*. En nuestro caso son "**W**" y "**K**". Ahora eliminamos ambos caracteres y ponemos en su lugar al *Nudo1*, que posee un número de apariciones de 1 (la suma de apariciones de "**W**" y "**K**"). *En este* momento tenemos una tabla de 25 datos (24 letras y un nudo). Ahora repetimos el proceso de forma idéntica: En este caso, el *Nudo2* se crea a partir del *Nudo1* y de la "**Z**", y poseerá un número de apariciones de 29. Se eliminan *Nudo1* y "**Z**" de la tabla y se añade *Nudo2*, por lo que ahora estaremos trabajando con una tabla de 24 datos (23 letras y 1 nudo), etc.

El proceso continúa hasta que sólo tengamos un nudo, al que llamaremos **RAIZ**. Como podéis comprobar, independientemente de la estructura del árbol en sí SIEMPRE vamos a necesitar n-1 nudos para poder acceder a n elementos. La profundidad de un árbol (el número máximo de nudos por los que hay que pasar para acceder a un elemento) varía a medida que vamos añadiendo datos. El rango de valores posibles cubre desde un mínimo de $LOG_2(n)$ hasta un máximo de $n/2$ niveles. Si trabajamos con árboles K-arios simples, los intervalos son desde $LOG_k(n)$ hasta $n/k$. Como podéis ver el algoritmo generador de Huffmann es bastante sencillo y facilmente programable.

Bien, ya hemos generado el árbol; ahora "sólo" queda almacenarlo de la forma más práctica posible. Y pongo el "sólo" entrecomillado porque no es tan fácil como puede parecer. El formato de almacenamiento del árbol debe ser una solución de compromiso entre la velocidad que deseamos obtener y la memoria que nos podemos permitir gastar, algo bastante normal por otro lado. También es indudable que puede variar según las características del lenguaje de programación que utilicemos e, incluso, según el micro-

**FIGURA 3:**

*Arbol óptimo generado a partir de los datos de la figura 2 mediante un programa a tal efecto. (ver texto)*




*El scanneado del árbol es de arriba a abajo. En cada bifurcación se utiliza un bit para decidir si se toma la rama izquierda o la derecha. Cuando se llega a una rama sin salida significa que ya encontramos el dato buscado.*

*En realidad el árbol generado por el programa fue otro equivalente. El árbol original fue rediseñado para simplificar su aspecto (el original era bastante más "liado"). Dos árboles se dicen equivalentes si cualquier dato está representado por el mismo número de bits en ambos, por lo que se comportan de forma idéntica en la práctica. Cuando dos árboles son equivalentes son intercambiables, pero en ciertos casos se utiliza uno en particular porque su forma es más cómodo (como cuando hay que representarlo graficamente, como hemos hecho aquí). En este tipo de estructuras, el número de árboles k-arios equivalentes es de $(k!)^{n-1}$, donde k es la dimensión y n el número de elementos. En nuestro caso sería $2^{n-1}$.*

**Figura 2:**

| | | | |
|---|---|---|---|
| A | 1145 | N | 856 |
| B | 124 | O | 1015 |
| C | 485 | P | 305 |
| D | 561 | Q | 120 |
| E | 1461 | R | 684 |
| F | 84 | S | 852 |
| G | 103 | T | 591 |
| H | 54 | U | 460 |
| I | 700 | V | 71 |
| J | 61 | W | 1 |
| K | 0 | X | 31 |
| L | 633 | Y | 74 |
| M | 333 | Z | 28 |

*Esta tabla muestra el número de apariciones de las 26 letras del alfabeto en mi artículo de curiosidades informáticas de este mes. No se tienen en cuenta los acentos ni la letra eñe.*

procesador que poseamos si escribimos las rutinas de manejo del árbol en ensamblador. En principio, existen dos "estilos" completamente diferentes de gestionar un árbol:

Como su propio nombre indica, ***el método del árbol*** crea una "copia" del árbol generado en memoria. A la hora de localizar un dato, partimos de la raiz y vamos recorriendo los diferentes nudos hasta que llegamos al dato buscado. ¿Cómo sabemos cuando debemos coger la rama izquierda o la derecha, o cómo sabemos cuando hemos llegado a un nudo terminal? (un nudo terminal es aquel del que no salen más ramas, por lo que es el dato buscado). Como ya he dicho, se puede almacenar el árbol de muy diversas formas, por lo que sólo voy a hacer una ligera aproximación en el supuesto de que estuviéramos utilizando Código Máquina. Si estamos trabajando con menos de 128 datos podríamos almacenar cada nudo en dos bytes, uno para cada rama. Si el bit superior es 1, se trata de una rama y los 7 bits inferiores indican a qué nudo debemos saltar (la posición de ese nudo será RAIZ+nudo*2). Si el bit superior es 0, es un dato y los bits inferiores nos indican cual es. Este formato es, al mismo tiempo, manejable, compacto y muy rápido.

El sistema se puede ampliar a más datos, sin más que utilizar palabras o palabras largas en vez de bytes pero, en general, se desperdiciará memoria. En general, utilizando esta técnica se precisan ***2*TRUNC (LOG2(n))+2*** bits para expresar n datos. Es evidente que de esta forma los parámetros de cada rama pueden quedar repartidos de una forma un tanto incómoda entre octetos adyacentes. Dependiendo de nuestra experiencia y del lenguaje utilizado (el ensamblador es el más conveniente por darnos total libertad sobre la forma de manejar los datos), podemos obtener una velocidad de acceso aceptable, dependiente en gran medida de la aplicación a la que se destine el árbol. Si queremos velocidad no tendremos otra opción más que utilizar una forma de almacenamiento menos compacta, organizando los datos de forma más práctica. Obviamente, el precio a pagar es la memoria...


La otra técnica utilizada se llama ***desarrollo en tablas***. La idea en la que se inspira es bastante diferente. Se basa en crear unas tablas que informan sobre el código asociado a cada carácter. Aquí la búsqueda no consiste en ir recorriendo un árbol binario, sino en comparar códigos hasta llegar a uno que coincida. Este sistema sólo es práctico con árboles pequeños (unos pocos cientos de datos), dado que suele ser bastante más lento que el algoritmo anterior. Sus ventajas consisten en una muy reducida ocupación de memoria y, sobre todo, en una extremadamente sencilla codifi-

cación. Estas ventajas suelen hacerlo preferible, aunque es mucho menos eficaz. Para que os hagáis una idea apoximada de como funciona este algoritmo, vamos a verlo en acción:

Para empezar, cada entrada de la tabla precisa una bandera especial para saber cuanto mide. Recordemos que esto es así porque cada dato es expresado mediante una cadena binaria de longitud variable. Para que el método funcione correctamente necesitamos conocer el tamaño de dicha cadena, y la forma más compacta de hacerlo consiste en incluir un flag dentro de la propia cadena que indique donde empieza o acaba. Supongamos un dato expresado como 3 ceros. El generador de la tabla crearía la entrada siguiente: *1000*. El uno inicial es el flag, cuya importancia se verá siguiendo el algoritmo.

Al empezar a buscar un caracter, se inicializa un registro con en valor uno. A continuación se realiza un desplazamiento a la izquierda (un ***LSL*** en ensamblador), introduciendo como nuevo bit cero, el primer bit del dato; en nuestro caso, un cero. Así obtenemos la cadena *10*. Ahora la comparamos con nuestra tabla, realizando una operación ***XOR***. Si durante la búsqueda se obtiene un cero, significa que ya hemos encontrado el dato buscado. Si no es así, se vuelve a realizar el ***LSL***, introduciendo los bits siguientes del dato como nuevos bits cero, y se repite el proceso hasta realizar un ***XOR*** cuyo resultado sea cero. Si hemos generado la tabla de forma correcta, este hecho se producirá de forma ineludible.

Como se deduce de lo explicado, este método puede ser bastante lento, pero es muy simple y funciona. Y lo que es más importante: Aunque no suele ser un sistema recomendable, resulta bastante interesante es algunas aplicaciones específicas. La idea de la bandera inicial está tomada del los sistemas de transmisión asíncronos de datos, de lo que podréis deducir sus aplicaciones. El tema de la trasmisión de datos será analizado en profundidad en un próximo número.

Evidentemente, estas no son las únicas formas de organizar un árbol obtenido aplicando HUFFMANN de una forma adecuada para trabajar con él. En el número anterior ya comentábamos algunos métodos en mi artículo sobre los algoritmos de compresión de textos. Os recomiendo releerlo para refrescaros la memoria. Existen también multitud de formatos de almacenamiento adaptados a necesidades específicas en determinadas aplicaciones que no son apropiados para otras. Los métodos estudiados aquí (y en el número anterior) son de propósito general general, por lo que son utilizables en cualquier aplicación. Sin embargo, siempre debemos tener en cuenta que en cada caso se pueden optimizar en gran medida los resultados tanto utilizando variantes y/o combinaciones de dichos métodos, como diseñando algoritmos totalmente nuevos. Yo sólo os he abierto el camino: La experimentación es vuestra.

*Al principio de este artículo doy a entender que el uso del algoritmo generador de Huffmann NO produce árboles óptimos. El caso es que durante mi investigación sobre este tema realicé una optimización a dicho algoritmo que mejora lor árboles en un 12.5% en el mejor de los casos (en el peor, crea un árbol idéntico al que generaría Huffmann). De momento no voy a contaros nada sobre esta modificación. Daré una explicación de este nuevo método en el proximo número. Hay que ahorrar páginas...*

*Por cierto, quien lo desee puede utilizar el árbol calculado en este artículo en sus propios programas. Dicho árbol fue calculado sobre un Atari STE mediante un programa escrito a tal efecto en GFA BASIC, y su tiempo de cálculo fue de 1.7 segundos. (tiempo de lectura de ficheros no incluido y con una versión sin compilar y programada "deprisa y corriendo" en una mañana de sábado...). En algún número próximo daré un árbol más completo que incluya todos los códigos ASCII imprimibles.*

TECNICA

# UNIDADES DE ENTRADA/SALIDA

por Pablo Lobariñas

***En este articulo vamos a ver la importancia de estos componentes del ordenador. Sabremos que son , para que sirven y algunos de sus tipos. Por supuesto intentaremos utilizar un lenguaje no muy tecnico para poder facilitar al profano en la materia la comprensión del mismo.***

En esta primera parte del articulo , daremos una introducción al tema , al mismo tiempo que explicamos en que consisten y para que utilizan el ordenador estas unidades. Posteriormente hablaremos de su funcionamento y de los tipos que existen.

Asi pues , y despues de esta pequeña introducción vamos a empezar por explicar que es una unidad de entrada\salida.

Para ello nada mejor , a mi entender , que compararlas con nuestro propio cuerpo , cuyo funcionamiento es conocido por todos mas o menos , proporcionandonos asi una base para poder entender en que consisten , que son estas partes del ordenador.

Pues bien,todos estamos familiarizados con las distintas sensaciones que nos llegan de distintas partes del cuerpo , desde las imagenes que percibimos a traves de nuestros ojos , pasando por los olores que captamos a traves de nuestras fosas nasales , hasta el gusto de los alimentos y bebidas que ingerimos todos los días. Todo ello llega a nuestro cerebro mediante unos canales que traducen y trasladan esta información desde el lugar en que es captada. Sin estos canales no podriamos diferenciar lo que nos rodea , no podriamos saber si estamos o no hablando con alguien o si caminamos en la dirección correcta , o si llueve o hace el sol,no sabriamos nada de nuestro entorno , estariamos aislados. Por esta razón se creó el sistema nervioso en nuestro organismo , para poder procesar la información que nos llega del exterior en forma de señales luminicas, sonoras, tactiles, transformandola en señales comprensibles y interpretables por nuestro cerebro , de forma que este sea capaz de sacar una conclusión , una idea mediante esa información que llega hasta el.

Por supuesto nosotros también podemos relacionarnos con el exterior , utilizando para ello el habla , o incluso otro tipo de señales , quizas no tan convencionales (mímica , señales luminosas , etc...). Pero para ello primero hemos de traducir el lenguaje empleado por nuestro cerebro a un lenguaje comprensible por el exterior , a parte de llevar este al lugar adecuado para ser enviado. Como podemos comprobar nuestro sistema nervioso cumple una función muy importante de relación sin el cual esta seria imposible.

En el ordenador va a pasar exactamente lo mismo . Los dispositivos de entada\salida se van encargar de comunicar al ordenador , con los distinctos perifericos a los que esta conectado. Es decir , nuestro ordenador va a

contar con una nueva serie de dispositivos externos (perifericos) mediante los cuales va a realizar distintas tareas como impresión , visualización de mensajes en la pantalla , carga de programas a través de la unidad de disco , etc... Al mismo tiempo , tambien va a recibir información del o de los usuarios que lo esten utilizando a traves del teclado , las palancas de mando ("joystick") , las propias impresoras que comunican la falta de papel o los errores producidos en la grabación de un fichero , etc...

Como es de suponer , todos los perifericos van a tener , normalmente , sus propios lenguajes , sus propias formas de comunicación , que como es lógico no siempre van a coincidir con la forma de hacerlo del ordenador. Por lo tanto , es necesario un traductor que se encargue de adecuar el lenguaje empleado por el periferico al lenguaje empleado por el ordenador , y viceversa.

Ademas de esto , hemos de tener en cuenta que normalmente la UCP (el cerebro del ordenador ) trabaja a una velocidad muy superior a la lo hace cualquier periferico , con lo que si la UCP tuviera que encargarse de la transferencia de información con las unidades externas , además sus otras muchas tareas , perdería muchísimo tiempo , tiempo que precisamente no suele sobrarle . De ahi , que delegue esta función en los canales de entrada / salida , que se encargaran de realizar esta función , liberando así a la UCP de trabajo extra y consiguiendo de esta manera un mejor rendimiento de la misma.

Por estas mismas razones en muchas ocasiones a este tipo de unidades o componentes se les suele llamar "ports" (puertos) , pues al igual que estos son lugares en que se recibe información , y desde los cuales parte hacia el exterior.

Veamos ahora los distintos tipos de unidades de e/s que se emplean para realizar las tareas de intercambio. Son dós:

-*Canales:* el ordenador puede realizar un mayor número de operaciones por segundo , que transferencias de información (como ya dijimos anteriormente ) en ese mismo tiempo , lo que obligaría a este a mantener bloqueada su UCP mientras se realizán estas tareas de transferencia. Si el ordenador tuviera que esperar , por ejemplo , a que la impresora terminara de imprimir una linea para continuar con la siguiente, tardaría tanto tiempo en llevar a cabo esta tarea que uno podría dormirse en la silla mientras lo hace. Con los canales el ordenador puede enviar toda la información requerida al canal , que a partir de ese momento va a gestionar esa información y a realizar todas las operaciones necesarias para enviarla al periferico correspondiente. Mientras tanto , el ordenador puede realizar otro tipo de operaciones . Normalmente este tipo de dispositivos no suelen tenerlo lo microordenadores , que suelen realizar el intercambio de información directamente.

-*Controladores de perifericos:* gestionan varias unidades del mismo tipo , y para ello interpretan las instrucciones que entregan o reciben del ordenador , además han de decodificar la operación que se le ordena ejecutar y emitir en sentido inverso información sobre el estado del periferico.

La complejidad de los controladores puede ser muy alta normalmente , dependiendo del tipo de periferico que gestionan y así algunos ordenadores tienen coprocesadores dedicados exclusivamente a la labor de controlar los dispositivos perifericos a los que esta conectado.

En ocasiones , varios de los procesadores a los que esta conectado el ordenador pueden solicitar al mismo tiempo el uso de uno de sus perifericos, con lo cual el canal habra de decidir a cual de ellos va a conceder el primer puesto de la lista . Esta operación se puede realizar siguiendo varios procesos dependiendo del tipo de canal de e/s .

Estos procesos pueden ser de tres tipos :

1. **FIFO** ( First In , First Out ) o lo que es lo mismo , el primero en llegar es el primero en salir . Así a medida que los dispositivos van

solicitando su acceso al periferico , el canal los va conectando a este.

2. **PRIORIDAD EXTERIOR:** Desde fuera del ordenador se van a asignar una serie de prioridades a los distintos dispositivos , y el canal siguiendo estas va a conectar antes a uno u a otro . Este método tiene el problema de que si un procesador con máxima prioridad tiene muchas transacciones pendientes , el sistema se queda bloqueado.

3.**ASIGNACION CICLICA:** Con este método se atiende de una forma ciclica y constante a todos los dispositivos que hayan solicitado su intervención en un periferico . Por desgracia solo se puede utilizar en determinados sistemas.

### *TIPOS DE TRANSMISION*

Las unidades de e/s han de actuar de intermediarias entre dispositivos externos y UCP , siendo su principal misión intercambiar información entre ellos ( como hemos visto antes ). Para realizar adecuadamente esta misión utilizan distintos formas de comunicación o transmisión . Estas pueden ser de dos tipos:

- en paralelo : mediante este método el ordenador va a recibir y transmitir un número determinados de unidades de información (bits) al mismo tiempo. Es decir , en vez de enviar la información una detrás de la otra lo va a hacer todo de una vez ( hasta cierto limite , claro ).

- En serie : justamente lo contrario de lo anterior. En este caso las unidades de información van a ir tanto en la transmisión como en la recogida , una detrás de la otra en lugar de hacerlo todo de un golpe , es lo que se llama transmisión secuencial.

Por supesto es mucho más rapida la transmisión en paralelo que en serie , pero ciertos dispositivos utilizan solamente esta ultima , de ahi que ciertos ordenadores dispongan de una serie de registros que se encargan de transformar un tipo de transmisión en otra.

### *LOS INTERFACES RS-232 Y CENTRONICS*

La industria informática , y dentro de esta la que se dedica especialmente al hardware , ha intentado en los ultimos años realizar una

normalización dentro de los distintos sistemas de transmisión y conexión entre ordenadores y perifericos, de forma que en sus respectivas categorias , han surgido dos tipos específicos de interface , que són el RS-232 ( para transmisiones en serie ) y el CENTRONICS ( para transmisiones en paralelo ). Y aqui hemos de aclarar lo que es un interface o de lo contrario quien no este familiarizado con los ordenadores podria perderse un poco.

Existen dos formas de entender lo que es un interface , es decir , se denomina interface al conjunto de canales y circuitos de control asociados que permiten la conexión entre un procesador y sus unidades periféricas pero también puede denominarse interface a las distintas especificaciones de conexión de dos unidades cualesquiera.

Normalmente el interface suele formar parte del propio ordenador (es decir , del hardware del mismo ) aunque también existen ciertas rutinas ( software ) encargadas de realizar tareas de comunicación.

La conexión entre periferico y procesador se realiza mediante una serie de conectores multiples provistos del cableado necesario para que puedan fluir a traves de ellos , las señales de control y los datos a transmitir al periferico o al ordenador.

De esta forma se pueden ir acoplando poco a poco al ordenador nuevos sistemas que amplien su capacidad , y siempre teniendo en cuenta el número de canales de que disponga el ordenador para poder administrar estas unidades perifericos.

Como dijimos anteriormente , el RS-232 es el sistema de interface en serie mas extendido , aunque si bien es cierto que existen otros tipos de sistemas de transmisión en serie estos son menos utilizados y no muy comunes.

Las unidades empleadas para medir la velocidad es el baudio y podemos decir que cada diez baudios se corresponden

aproximadamente a un caracter . Las velocidades más usuales de transferencias son 50, 100, 300, 600, 1200, 2400 y 9600 baudios por segundo.

Por último , diremos que el interface en paralelo por excelencia es el CENTRONICS cuya utilización se ha generalizado increiblemente en el terreno de las impresoras.

*Hasta aqui nuestro articulo, espero que os haya gustado y hayais disfrutado leyendolo . Hasta la proxima ocasión.*

HISTORIA

# HISTORIA DE LA INFORMATICA

NACHO AGULLO

## PARTE IV

***En la parte anterior hemos visto como el mundo contable empezaba a acudir a las máquinas para simplificar sus cálculos. Las máquinas representaban una ayuda cada vez más imprescindible para la gestión de las empresas: cada vez se manejaban más datos y pronto el volumen de éstos llegó a hacerse imposible de manejar. La alternativa aparece en forma de máquinas para el proceso de datos: amanece la era de la informática de gestión.***

Fué en la Oficina de Censos de los Estados Unidos donde por primera vez se encontraron con el límite de la capacidad humana. El proceso de los datos de los censos, que se celebraban cada diez años, se alargaba durante años enteros.

En 1886 se calculó que, con el sistema manual de recopilación y proceso de los datos vigente, no se terminaría con el censo de 1890 hasta la fecha del siguiente censo.

Por ello, en 1889 se convocó un concurso para proveerse de equipos de proceso de datos, probándolos sobre datos del censo anterior. El equipo ganador fué el del ingeniero estadístico de origen alemán Hermann Hollerith.

### *Las máquinas de Hollerith*

Hollerith ya había trabajado durante cuatro años en la Oficina de Censos; allí se había dado cuenta de que muchos de los datos eran del tipo "sí" o "no". Hollerith, conocedor de los trabajos de Babbage y Torres Quevedo (del cual hablaremos en la siguiente parte de nuestra Historia de la informática), y conocedor, por haberlo visto en Europa durante un viaje, del sistema de automatización en la industria textil de Jacquard, sintetiza todos estos avances en una máquina estadística, que utilizaba una cinta de papel perforado, no para codificar el programa de la máquina, sino únicamente para almacenar datos. Esta máquina la patentó en 1884.

Pronto apareció salida para la máquina de Hollerith. Se utilizó en la administración militar, y para estadísticas de sanidad, en ciudades como Baltimore.

En 1889, Hollerith mejoró su máquina mediante la sustitución de la cinta perforada por tarjetas perforadas, del tamaño exacto de los billetes de un dólar (posiblemente, debido a que los únicos equipos que Hollerith pudo adaptar habían sido construidos para manipular dinero), con agujeros redondos que se realizaban con los punzones que utilizaban los conductores de autobuses para perforar los billetes. Posteriormente se pudo contar con punzones especiales que

cortaban cuadrados de seis milímetros. Este fué el equipo que ganó el concurso de la Oficina de Censos en 1889.

El censo de 1890 fué procesado en tan sólo dos años y medio, la tercera parte del tiempo empleado para el censo anterior.

### *Una breve mirada al sistema de Hollerith*

Básicamente, el sistema constaba de una máquina clasificadora, una reproductora y una tabuladora.

La clasificadora lee las tarjetas mediante una escobilla y un tambor de latón sobre el que se desliza la tarjeta; al pasar una perforación bajo la escobilla, esta establece con-tacto eléctrico con el tambor, activándose un electroimán que dirige la tarjeta al cajetín correspondiente.

La reproductora consta de dos unidades diferenciadas: una lectora, que funciona según la misma técnica que la clasificadora, y otra perforadora. Esta perfora las tarjetas vírgenes según el modelo que se introduzca en la unidad lectora; una vez leído el modelo, puede realizarse perforación en serie de cuantas tarjetas se fijen, funcionando solamente la unidad perforadora. Con esta máquina puede realizarse también comparaciones entre dos juegos de tarjetas perforadas, utilizándose ambas unidades en el proceso.

La tabuladora computa e imprime los totales de los diferentes procesos de datos. En posteriores modelos, Hollerith le incorporó la suma, la resta y otras operaciones aritméticas sencillas. Con dos juegos de escobillas, la tabuladora puede comparar tarjetas; y con el dispositivo de impresión se escribían los resultados en papel. Las primeras tabuladoras imprimían cien líneas por minuto.

### *La explosión de la Informática de Gestión*

Para el censo de 1900, Hollerith presentó una nueva generación de máquinas más eficaces. Su sistema comenzaba a utilizarse en la empresa privada además. En 1901 se utilizaron estos nuevos equipos para el censo agrícola.

Pero fué en 1905 cuando comenzaron a caducar las patentes de Hollerith y la Oficina de Censos, estimando demasiado caro a Hollerith, subvenciona la construcción de máquinas alternativas, apareciendo así el estímulo de la competencia para la Imformática. Es el comienzo de la Informática de Gestión.

### *BIOGRAFIA*

*Hermann Hollerith ( 1860 - 1929). Cursó estudios de ingeniero estadístico en la Universidad de Columbia, al término de los cuales entró a trabajar en la US National Census Office, en 1879. En 1882 se matriculó en el MIT (Massachusetts Institute of Technology) para realizar investigación; al año, abandonó su trabajo en la Oficina de Censos para proveer equipos para procesar los datos del censo. Al contar con las patentes, cargó al gobierno más de tres millones y medio de dólares de la época de factura. En 1900 presentó una nueva generación de equipos... y una factura similar. En 1890 obtuvo el doctorado en filosofía por la Universidad de Columbia por su trabajo en el campo del procesamiento de la información.*

*Cuando, a partir de 1905, comenzaron a caducar las patentes, Hollerith tuvo que hacer frente a la competencia. Fundó la Tabulating Machines Company, que en 1911 pasó a ser la CTR (Consultating Tabulating Recording). A partir de*

*1914, contó con el prestigioso Thomas J. Watson en la dirección, antiguo director general de NCR. En 1924, el nombre de la empresa se cambió por IBM (International Business Machines), nombre que se ha mantenido hasta nuestros días.*

## ***Bibliografia.***


*Historia de la Informática.* Amparo Gil Orihuel e Ignacio Rieiro Marín. Ed. Alhambra.

*Enciclopedia Monitor.* Ed. Salvat.

*Enciclopedia Mi computer.* Ed. Delta.

*Gran Enciclopedia Informática.* Ed. Nueva Lente.