Aspectos tecnológicos de los sistemas informáticos

Antoni Martínez Ballesté
Gregorio Robles Martínez

PID_00150272

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Los textos e imágenes publicados en esta obra están sujetos –excepto que se indique lo contrario– a una licencia de Reconocimiento-NoComercial-SinObraDerivada (BY-NC-ND) v.3.0 España de Creative Commons. Podéis copiarlos, distribuirlos y transmitirlos públicamente siempre que citéis el autor y la fuente (FUOC. Fundación para la Universitat Oberta de Catalunya), no hagáis de ellos un uso comercial y ni obra derivada. La licencia completa se puede consultar en http://creativecommons.org/licenses/by-nc-nd/3.0/es/legalcode.es

Índice

Introducción

En este módulo estudiaremos los aspectos tecnológicos de los ordenadores y los sistemas informáticos. El ordenador se ha convertido en el elemento que vehicula el conocimiento dentro de la sociedad de la información. Los sistemas informáticos están formados por uno o varios ordenadores, de diferentes tipos y con diferentes propósitos (ordenadores personales para trabajos de oficina, computadores que forman parte de entornos industriales, etc.). Sobre estos ordenadores se ejecuta una gran diversidad de programas. En muchos sistemas informáticos, el software incluye sistemas de información para el almacenamiento y gestión de datos.

Los términos *computación* e *informática* definen el propósito inicial de los ordenadores: el cálculo y el tratamiento automático de la información. A pesar de ser términos sinónimos, la palabra *informática* tiene hoy día un sentido más generalista que *computación*, ya que la informática ha permitido el uso al público en general de toda una serie de potentes herramientas que facilitan multitud de tareas, y también ha facilitado una comunicación sin barreras y el acceso a cantidades ingentes de información.

Las palabras *computador* y *ordenador* también tienen significados ligeramente distintos. Un computador tendría una utilidad no tan generalista como un ordenador. Normalmente, los computadores designan los primeros ordenadores (destinados a hacer cómputos y cálculos), mientras que el término *ordenador* se aplica a los computadores actuales. Sin embargo, en inglés el término *computer* sirve para definir tanto computadores como ordenadores.

Los ordenadores empezaron como herramientas de cálculo para facilitar tareas rutinarias como operaciones aritméticas o grandes procesos de cálculo, por ejemplo, la gestión de censos. En el apartado 1 del módulo, conoceremos los hitos más importantes en la historia de la informática y los ordenadores. En el apartado 2 del módulo, veremos cómo tratan los ordenadores la información en el mundo digital.

En el apartado 3 del módulo, distinguiremos las diferentes partes que forman un sistema informático. Básicamente, veremos que hay una parte tangible, llamada hardware, que sirve de soporte a la parte intangible de la informática: los programas y la información. También definiremos este componente flexible de los ordenadores: el software. Veremos qué es el software y aprenderemos cómo se crea. Así, conoceremos cómo se transforma un programa desde la especificación hasta un programa que se pueda ejecutar en un ordenador.

Finalmente, en el apartado 4 del módulo, presentaremos los tipos de ordenadores actuales que podemos encontrar, en el último apartado, y los sistemas de información, que se utilizan de manera generalizada en el mundo empresarial y gubernamental.

Objetivos

Los objetivos que el estudiante habrá alcanzado al finalizar este módulo son:

1. Tener una perspectiva histórica de la informática.

2. Comprender las bases del tratamiento de la información por parte de los ordenadores.

3. Distinguir las partes fundamentales de un sistema informático y entender cuáles son sus objetivos.

4. Entender los conceptos más importantes relacionados con el software, incluidas todas sus etapas de creación desde la especificación en algoritmos hasta su implementación con un lenguaje de programación de alto nivel.

5. Conocer la evolución de los lenguajes de programación.

6. Conocer los diferentes tipos de software: el software de sistemas (del que el más significativo es el sistema operativo) y el software de usuario final, con especial énfasis en los sistemas de información.

1. Perspectiva histórica de la informática

Los inicios de la informática son más bien difusos. Lo que hoy conocemos como ordenadores e informática es el resultado del avance de varias disciplinas, en especial las matemáticas, la electrónica y la física. Sin embargo, los esfuerzos de la humanidad por tener artilugios que ayudaran con los cálculos se remontan a épocas antiguas.

Los inicios de la informática van ligados a la invención de diferentes utensilios y artilugios dedicados a facilitar el cálculo matemático.

Unos de los más antiguos es el ábaco, originario de la China del siglo V a. C. Se trata de un marco de madera con ejes sobre los que se deslizan bolas. Los ábacos facilitan la realización de sumas, restas, multiplicaciones y divisiones. Este utensilio fue adaptado y utilizado por muchas otras culturas.

Durante los siglos XVII y XVIII, el avance de la mecánica propicia el diseño e incluso la producción de diferentes máquinas de calcular. Una de las más célebres es la Pascalina, creada en el año 1645 por Blaise Pascal y de la que se produjeron una cincuentena de copias.

La Pascalina, una de las primeras calculadoras. Expuesta en el Musée des Arts et Métiers de París.

En el siglo XIX tiene lugar el primer diseño de lo que se puede considerar un computador programable y no una mera máquina de calcular: la máquina analítica, de Charles Babbage.

> La máquina analítica de Babbage estaba ideada para poder implementar pequeños programas de cálculo, mostrar la salida y guardar datos en una memoria.

Desgraciadamente, su inventor no la pudo ver realizada por una serie de problemas, sobre todo porque la ingeniería de la época aún no estaba preparada para llevarla a la práctica. Ada Lovelace, hija de Lord Byron, estudió el funcionamiento de esta máquina e incluso propuso métodos para definir procedimientos a partir de las operaciones que la máquina permitiría realizar. Ada Lovelace es considerada la primera persona que trabajó en los conceptos de programación de ordenadores.

Aquí empieza lo que se considera la historia de los ordenadores. Acto seguido, veremos sus hitos más importantes. Pero antes describiremos una división clásica de las etapas de la historia de los ordenadores, que pasa por diferentes generaciones:

- La primera generación comprende los grandes ordenadores electromecánicos, en los que la electrónica se controlaba mediante válvulas y los circuitos eran cableados.

- La segunda generación se inicia con la introducción del transistor y la consecuente reducción de tamaño de los equipos. Un módulo basado en componentes electrónicos del tamaño de una bombilla ahora se podría diseñar con componentes no mayores que una mosca.

- La tercera generación supone unos ordenadores aún más pequeños debido a la aparición de los microchips. Así pues, un circuito cableado de dimensiones considerables comparables al tamaño de una mesa de comedor se podía reducir al tamaño de un sello de correos.

- La cuarta generación de los ordenadores comprende los ordenadores personales. La miniaturización iniciada con la tercera generación aún es más evidente, ya que los circuitos integrados pueden incluir diferentes circuitos, incluso sobrepuestos en diferentes capas, para poder realizar varias tareas.

Se considera que la generación actual de ordenadores es la quinta. En esta generación se introducen conceptos como el multiprocesador y la memoria caché.

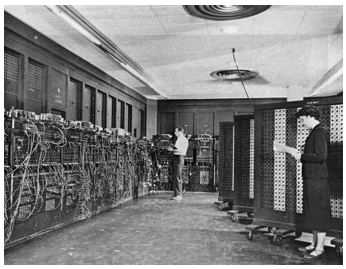
1.1. La electrónica y los primeros computadores

La llegada de la electrónica a mediados del siglo XX propició la materialización de diseños que, con tecnologías puramente mecánicas, no eran realizables. Así pues, durante la década de los años cuarenta vieron la luz los primeros computadores electromecánicos. Estos ordenadores, grandes logros de la ingeniería de la época, eran muy diferentes de los ordenadores que conocemos hoy en día: eran máquinas de gran tamaño, alimentadas por una potencia eléctrica considerable y generadoras de ruido y altas temperaturas. Para controlar los voltajes e intensidades que regulaban los módulos de procesamiento de información se utilizaban válvulas, como las que había dentro de las radios antiguas.

En el año 1941, se construyó el ordenador Z3, que se presentó en Berlín. Desgraciadamente, este equipo quedó destruido en un bombardeo durante la Segunda Guerra Mundial. Por ello, la popularidad la ganó el Mark I, diseñado y construido en la Universidad de Harvard. Fue presentado en el año 1944. Le siguió el ENIAC[1], presentado en el año 1946 en la Universidad de Pennsylvania. Ambas máquinas tenían grandes dimensiones y ocupaban una sala entera. Elevaban la temperatura ambiente y necesitaban una gran cantidad de energía para arrancar. La diferencia entre el ENIAC y el Mark I es que con el primero se interactuaba por medio de cables y tableros de conmutación, mientras que

(1)ENIAC es el acrónimo de *electronic numerical integrator and calculator*, en español, calculadora e integrador numérico electrónico.

el segundo disponía de lectores de cinta perforada. Esta técnica de almacenamiento ya se utilizaba para las partituras de las pianolas automáticas y las primeras calculadoras como la máquina analítica.



El ENIAC, uno de los primeros grandes ordenadores

Las máquinas de calcular analógicas

Aunque la gran mayoría de ordenadores de estas épocas se basan en el tratamiento digital de la información, durante el siglo XX también se desarrollaron máquinas de calcular analógicas. En éstas, por ejemplo, para realizar una operación de suma se sumaban dos voltajes diferentes. El resultado se mostraba en un voltímetro.

Durante los años cincuenta, diferentes empresas fabricaron grandes ordenadores como producto comercial. El primer computador comercial fue el UNIVAC[2] I. De estas máquinas, que ocupaban unos 30 m^2 cada una, se vendieron unas 40 copias a un millón de dólares. IBM entró en el mercado de los ordenadores haciendo máquinas considerablemente más pequeñas y, en consecuencia, más económicas. Uno de los primeros modelos populares a partir del año 1954 fue el IBM 650, que pesaba unos 900 kilogramos de peso y necesitaba un espacio de unos dos metros cúbicos.

(2)En inglés, *universal automatic compute* (computador automático universal).

A finales de esta década, la aparición del transistor y los circuitos integrados facilitaron la reducción de tamaño de los computadores, a la vez que disminuyeron el consumo energético. Paralelamente, van apareciendo equipos con entrada y salida de datos, ya que como hemos mencionado anteriormente los primeros computadores había que programarlos mediante cables y enchufes (como es el caso del ENIAC) y la salida se representaba en un panel de bombillas. Las pantallas de televisión y las máquinas de escribir se adaptaron para facilitar la interacción de los operadores con las máquinas.

En los inicios de la informática, cada fabricante utilizaba sus sistemas de representación de información y sus sistemas de programación de ordenadores. Durante los años sesenta y setenta del siglo XX, se hicieron esfuerzos a fin de que los ordenadores se miniaturizaran y utilizaran estándares tanto en software como en la representación de la información e incluso desde el punto de vista de la comunicación.

Ved también

Las redes de ordenadores se introducirán en el módulo "Aspectos tecnológicos de las redes e Internet".

En este sentido, ven la luz los sistemas Unix, los lenguajes de programación generalistas para crear programas, e incluso aparecen los primeros intentos de interconexión de ordenadores en una incipiente Internet.

A finales de los años sesenta, aparece la ingeniería informática, especializada en el diseño de sistemas informáticos para gestionar los procesos de información. En la misma época, la creciente complejidad de la gestión de la información provoca que se ideen herramientas de ingeniería para el diseño de pro-

gramas, empezando lo que llamamos ingeniería del software. Los conceptos teóricos sobre los modelos matemáticos y de procesamiento de información relacionados con los computadores dieron lugar a las Ciencias de la Computación.

1.2. El ordenador personal

Los años ochenta tienen como estrella la irrupción de la informática personal, gracias a la aparición de la cuarta generación de ordenadores. Hasta entonces, las empresas grandes y medianas ya contaban con ordenadores *mainframes* a los que varios trabajadores podían acceder mediante sus terminales. Los ordenadores personales IBM PC[3] y los Apple hicieron posible que muchas pequeñas empresas y particulares se adentraran en el mundo de la informática, ya sea por afición o para mejorar su productividad.

(3)Del inglés *Personal Computer.*

Los IBM PC utilizaban chips fabricados por Intel y un sistema operativo (lo que permite la ejecución de los programas) creado por Microsoft. Ambas empresas se convirtieron en las más importantes en el mundo de la informática. La compañía Apple se hizo popular con el modelo Macintosh, que aprovechaba unas ideas que tenían su origen en investigaciones de la empresa Xerox en los años setenta: el ratón y las ventanas. Estos conceptos fueron llevados al mundo de los IBM PC por Microsoft mediante el sistema operativo Windows.

La evolución de los ordenadores durante los años ochenta y noventa se produjo a una velocidad vertiginosa: por una parte, aparecen microordenadores como el ZX Spectrum, que popularizaron la informática a coste asequible. Estas máquinas no eran compatibles con los PC de IBM (es decir, los programas creados para estos ordenadores no podían funcionar en los otros). Muchas compañías diseñaron sus microordenadores sin tener en cuenta, en general, la compatibilidad entre productos de diferentes marcas.

Microordenador de la casa Spectrum. Como muchos ordenadores de su época, había que conectarlos al televisor de casa.

Por otra parte, Intel permitió que otras compañías utilizaran sus chips, con lo que centenares de empresas fabricaron ordenadores personales compatibles con el de IBM y sensiblemente más económicos que el original. Así pues, después de unos años en los que multitud de marcas y modelos ocupaban parte importante del mercado de la informática doméstica, los ordenadores compatibles con los IBM PC pasaron a ser los más vendidos.

1.3. La informática, en todas partes y para todo el mundo

La evolución de los ordenadores PC en cuanto a velocidad y capacidad, la aparición de los ordenadores portátiles, las impresoras de calidad fotográfica, las cámaras digitales, etc. son historias bastante conocidas por nosotros, dado que son recientes y podemos hacer memoria de ellas.

Los ordenadores personales han adquirido una popularidad notable, en parte gracias a su abaratamiento y en parte debido a las posibilidades que ofrece Internet como pieza integrante de cualquier sistema informático. El ordenador ya no es una herramienta exclusivamente dedicada a mejorar la productividad: el ocio personal, la gestión de contenidos multimedia, el comercio electrónico, etc., son hechos del día a día vehiculados por los ordenadores personales.

One Laptop Per Child Association

Varias asociaciones, como One Laptop Per Child Association, promueven la fabricación y distribución de equipos informáticos para países en vías de desarrollo.

Los ordenadores portátiles se han convertido en un complemento al ordenador de sobremesa, sustituyéndolo en muchos casos. La miniaturización de los ordenadores ha permitido la fabricación y popularización de teléfonos móviles (que en el fondo son pequeños ordenadores), aparatos digitales de grabación de vídeo (que también lo son), coches controlados y gestionados por un sistema informático de a bordo, etc.

Personas de cualquier nivel cultural tienen acceso a las herramientas informáticas que les permiten formar parte de la sociedad de la información. La adquisición de información y su tratamiento automático se han convertido en universales en todos los ámbitos, al menos en lo que llamamos primer mundo.

2. Tratamiento de la información

Los ordenadores, y la gran mayoría de aparatos electrónicos, trabajan en lo que se llama **mundo digital**. En contraposición, el mundo real es analógico. En el mundo analógico, las fuerzas y señales que encontramos (como la gravedad, la electricidad, el sonido, los colores, etc.) son continuas, en contraposición al mundo digital, en el que las medidas son discretas.

Eludiendo una descripción basada en las matemáticas, pondremos un ejemplo. Supongamos que queremos medir la longitud de una mesa. Si utilizamos un sistema continuo de medida, utilizaremos toda la precisión necesaria para calcular el tamaño exacto de la mesa.

En el mundo analógico, todas las mesas tienen longitudes diferentes, aunque a veces éstas difieran en millonésimas de milímetro.

> En un mundo digital, las medidas y las señales son discretas, es decir, tienen una determinada precisión que especifica un conjunto de valores. Por lo tanto, no puede haber infinitos valores diferentes aparte de los que defina una determinada precisión.

En un mundo digital con una precisión de centímetros, todas las mesas que en el "mundo analógico" tienen 1.200, 1.201, 1.202, 1.203 y hasta 1.209 milímetros se considerarán de 120 centímetros.

En este apartado veremos cómo se trata la información en el mundo digital.

2.1. El mundo binario

Los circuitos electrónicos en los que se basan los computadores funcionan a partir de intensidades de corriente eléctrica. Así pues, un transistor puede estar emitiendo corriente o no hacerlo. De la misma manera, los sistemas primitivos de almacenamiento de información en tarjetas perforadas utilizaban agujeros para guardar los datos: o hay agujero o no lo hay.

Todo esto responde a lo que se llama *Álgebra de Boole*, ideada por el matemático George Boole en el siglo XIX. Las operaciones complejas de cálculo se pueden reducir a **operaciones lógicas binarias**, que trabajan sobre dos valores posibles (por ejemplo, el cero o el uno; el agujero o la ausencia de agujero). En los primeros ordenadores, estas operaciones se llevaban a cabo con diferentes módulos de cálculo que se basaban, en general, en las sencillas operaciones lógicas del álgebra de Boole. La representación de números en el sistema binario,

Los ordenadores cuánticos

Los ordenadores cuánticos, actualmente en fase de investigación, permitirían trabajar con más valores que los dos del Álgebra de Boole.

es decir, en ceros y unos, hizo posible que se pudiera tratar automáticamente la información con estos módulos de cálculo. Hoy en día, los ordenadores tienen la misma base de funcionamiento.

Los ordenadores tratan la información utilizando el sistema binario, formado por unos y ceros. Un **bit** es un dígito binario, es decir, uno de estos dos valores. En consecuencia, un bit es la cantidad mínima de información que puede tratar un ordenador.

Sin embargo, los módulos de cálculo y los actuales chips no realizan los cálculos sólo con un bit. Normalmente se utilizan grupos de bits, por ejemplo, 4, 8, 16, 32 o 64 bits. Lo más habitual históricamente fue trabajar con grupos de 8 bits, llamados *bytes*.

El byte

Como los módulos de cálculo podían trabajar con la información a base de "comérsela a mordiscos de 8 bits", el grupo de 8 bits se llama *byte*, que suena igual que *bite* (mordisco en inglés).

2.1.1. Los números en binario

Cualquier número se puede representar en binario. Por ejemplo, aquí tenéis todos los números naturales que se pueden representar con tres bits:

N° binario	N° natural
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

La cantidad de números naturales que se puede representar dependerá del número de bits. En concreto, con n bits se pueden representar 2^n números diferentes, del 0 al 2^{n-1}.

El número decimal que representa un número en binario

Para obtener el número decimal que representa un número en binario hay que multiplicar el dígito (cero o uno) por un valor que corresponde a la base (en este caso, dos para ser el sistema binario) elevada al número que indica la posición del dígito, entendiendo que la posición de más a la derecha es la posición cero. Para entenderlo mejor, veamos un ejemplo. En el caso de los tres bits, los valores de cada uno de los dígitos de izquierda a derecha son: $4(2^2)$, $2(2^1)$ y $1(2^0)$. Por lo tanto, el número binario 100 corresponde al número decimal 4, ya que $1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 4$. De la misma manera, el número 111 corresponde al número 7, ya que $1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 7$.

Fijémonos en que eso también sucede con el sistema decimal que utilizamos los humanos, en el que la base es 10. Por ejemplo, el número 2.009 se puede descomponer en varias potencias: $2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 9 \times 10^0$.

Origen del sistema decimal

Los antropólogos afirman que el origen del sistema decimal está en los 10 dedos que siempre nos han servido de base para contar.

En el caso de los números enteros (es decir, los que pueden ser positivos o negativos), existen diferentes alternativas para representarlos en binario. Una de ellas sería utilizar uno de los bits para decir si el número es positivo o negativo. En el caso anterior, se necesitaría un bit de más. ¡Fijaos en que aparece el caso extraño de que hay dos representaciones posibles del cero!

Números positivos		Números negativos	
N° binario	N° decimal	N° binario	N° decimal
0000	0	1000	–0
0001	1	1001	–1
0010	2	1010	–2
0011	3	1011	–3
0100	4	1100	–4
0101	5	1101	–5
0110	6	1110	–6
0111	7	1111	–7

Gracias a la representación de números en el sistema binario fue posible crear máquinas digitales de cálculo, desde los primeros grandes ordenadores hasta las calculadoras de bolsillo.

El sistema binario no es el único que se utiliza en informática. También es habitual el llamado sistema hexadecimal, en el que la base es 16. El objetivo de este sistema es representar números binarios de manera más compacta, utilizando menos dígitos que si los representamos con ceros y unos.

Número decimal	Número binario	Número hexadecimal	Número decimal	Número binario	Número hexadecimal
0	0000	0	8	1000	8
1	0001	1	9	1001	9

Número decimal	Número binario	Número hexadecimal	Número decimal	Número binario	Número hexadecimal
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

2.1.2. Los caracteres

Los primeros ordenadores estaban dedicados a realizar grandes cálculos, pero la transmisión y el almacenamiento de información digital precisaron de la necesidad de codificar esta información. Por ejemplo, los textos enviados mediante un sistema de transmisión se debían codificar con bits.

Código Morse

El uso de la codificación es anterior a los propios ordenadores. Un ejemplo lo encontramos en el código Morse, en el que diferentes símbolos (letras y cifras, básicamente) se representan en un código de puntos y rayas.

Uno de los sistemas de codificación de caracteres, y el más importante en la actualidad, es el ASCII[4]. Un código ASCII de 8 bits puede representar 256 símbolos diferentes. Por ejemplo, la letra "A" tiene asignado el código 65 (que en binario es 01000001), la cifra "1" tiene el código 49, el interrogante tiene el código 63, etc.

(4)Del inglés *american standard code for information interchange*, código estándar americano para el intercambio de información.

La codificación de caracteres

La frase "la vida es bella" ocupa en el ordenador un total de 16 bytes (13 letras más los tres espacios en blanco). El número 2009, si se trata como una cadena de caracteres, ocupará 4 bytes (32 bits), uno para cada cifra, pero si el número se trata como un número entero, ocupará 11 bits, ya que se utilizará su valor en binario: 11111011001.

Con el aumento de la transmisión de información con diferentes lenguas con el código ASCII aparece un problema: no se pueden representar todos los sistemas de escritura. Actualmente, se utiliza el código UNICODE (del inglés *universal code*) de 16 bits, donde se pueden representar incluso los símbolos de las lenguas asiáticas.

2.1.3. Los múltiplos del byte y del bit

Los sistemas informáticos de tercera y cuarta generación son capaces de tratar y almacenar millones de bytes de información. De esta manera, se han definido múltiplos del byte:

- Un **kilobyte** o KB son 1.024 bytes. También se llama "ka".
- Un **megabyte** o MB son 1.024 KB o 1.048.576 bytes. También se llama "mega".
- Un **gigabyte** o GB son 1.024 MB o 1.073.741.824 bytes. También se llama "giga".

- Un **terabyte** o TB son 1.024 GB o 1.099.511.627.776 bytes. También se llama "tera".

Ejemplos sobre cuánto pueden ocupar algunos elementos habituales

Una carta hecha con un procesador de texto puede ocupar 25 KB. No sólo se guardan los caracteres que la forman, sino también información adicional como el tipo de letra, el color del texto, posibles imágenes integradas, etc. Una foto de alta calidad puede ocupar 2 MB. Una canción para ser escuchada en un reproductor MP3 ocupa unos 4 MB. Una película en DVD puede ocupar unos 6 GB.

Ved también

Los formatos multimedia (imagen, vídeo y sonido) se explicarán en el módulo "La World Wide Web".

¿Por qué múltiplos de 1.024 y no de 1.000? Pues por razones estrictamente relacionadas con el sistema binario y la construcción de los ordenadores. Sin embargo, la especificación de capacidad de soportes de almacenamiento, como los discos duros y los DVD, utilizan múltiplos de 1.000. Así pues, uno esperaría poder guardar 5.046.586.572 bytes en un DVD de 4,7 GB, cuando realmente caben 4.700.000.000 bytes.

Cuando se habla de **velocidades de transferencia de información**, hablamos de múltiplos de bit por segundo. Por ejemplo, tendremos los kilobits por segundo (kbit/s) o los megabits por segundo (Mbit/s). En estos casos, se utilizan múltiplos de 1.000.

Ved también

Trataremos las velocidades de transferencia de información en el módulo "Aspectos tecnológicos de las redes e Internet".

2.2. El tratamiento lógico de la información

Una vez que hemos visto cómo se representan los valores numéricos y los caracteres en los sistemas digitales y/o informáticos, veremos algunos conceptos básicos sobre cómo se diseñan los módulos de cálculo que ya se encontraban en los primeros computadores.

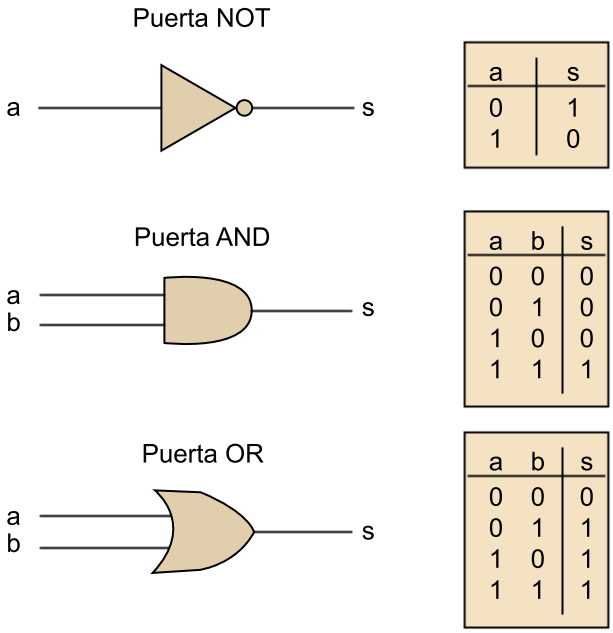
Hemos visto que el Álgebra de Boole se fundamenta en una serie de funciones lógicas que trabajan con bits. Usando estas funciones, se pueden hacer módulos sumadores, restadores, multiplicativos, etc. Las tres funciones lógicas más importantes son:

- La función **no**, llamada NOT, cambia el valor del bit que le entra. Por ejemplo, el valor 0 se convierte en 1 y el 1 en 0.
- La función **y**, llamada AND, tiene el valor 1 de salida cuando todos los valores de entrada son 1, y 0, en caso contrario.
- La función **o**, llamada OR, tiene valor 1 cuando alguna de las entradas vale 1, y 0, si todas las entradas valen 0.

Estas funciones tienen un símbolo que las representa y una **tabla de verdad** asociada. En la figura siguiente, se muestra el símbolo para las tres funciones lógicas que se acaban de definir, así como sus tablas de verdad.

Tabla de verdad

Una tabla de verdad especifica, por las diferentes combinaciones de los bits de entrada, cuál es la salida de una función lógica o de un circuito de cálculo.



Arriba, la puerta lógica NOT; en el centro, la puerta AND, y abajo la tiene la puerta OR.

Otras funciones utilizadas son la XOR, la NAND o la NOR. La función XOR tiene 1 como salida sólo si una de las dos entradas es 1. La NAND y la NOR son los resultados de "negar" las salidas de las funciones AND y OR, respectivamente. Todas estas funciones se pueden implementar físicamente en un circuito mediante la combinación de dispositivos electrónicos, como transistores. Al encadenar estos dispositivos, se consiguen funciones más complejas diseñadas a partir de las básicas y, en consecuencia, se pueden hacer módulos para cálculos complejos.

Los módulos de cálculo están diseñados para trabajar con un número concreto de bits, por ejemplo, 8 bits. Al encadenar módulos de éstos, se consiguen módulos capaces de procesar 16, 32 o 64 bits.

Para realizar operaciones más complejas que la suma o la resta, por ejemplo, una división o una raíz cuadrada, existen dos alternativas:

- Diseñar bloques específicos para llevar a cabo estas operaciones. Esto implicaría tener muchos módulos que a menudo no se estarían utilizando. Algunos de los primeros grandes ordenadores seguían esta alternativa.
- Hacer un único módulo capaz de realizar operaciones "complejas" a partir de la definición de procedimientos que utilicen un conjunto reducido de módulos de cálculo sencillos. Por ejemplo, hacer la resta a partir del módulo suma con el segundo operando cambiado de signo, hacer una división mediante restas, etc. Ésta es la tendencia que ha marcado el diseño de los ordenadores modernos. Este módulo de cálculo se llama *procesador*.

3. Componentes del sistema informático

Una vez hemos visto cómo los ordenadores y los sistemas digitales tratan la información, estudiaremos los componentes del sistema informático. Un sistema informático está formado por dos tipos de componentes:

- **Componentes de hardware**. Se trata de todos los componentes físicos y electrónicos que configuran los sistemas informáticos. Dentro de estos componentes encontramos, entre otros, el procesador, el teclado, el lector de DVD, etc.

- **Componentes de software**. Forman parte de él todos los programas que funcionan en el ordenador y los datos que éstos manejan. Se trata, pues, de la parte "intangible" del sistema informático.

Un elemento importante del software es el **sistema operativo**, que crea una "capa" intermedia entre el hardware y el software con el objetivo de facilitar la interacción del software y el usuario con el hardware. Un caso concreto de software es el que se encuentra registrado en los propios chips del hardware, por ejemplo, en los dispositivos móviles como teléfonos o GPS, que recibe el nombre de **firmware**.

3.1. La estructura del hardware

La estructura del hardware del ordenador se basa en la llamada arquitectura Von Neumann. John Von Neumann, un matemático prolífico, propuso que los ordenadores deberían estar gobernados por una **unidad de control**, que se encargaría de transportar información desde la memoria a una **unidad de cálculo** (llamada unidad aritmético-lógica) y devolverla a la **memoria** una vez acabadas las operaciones. Definió, además, unos **módulos para la entrada y salida** de información, así como tres caminos (o **buses**) por los que circularían diferentes tipos de información: los datos con los que trabajar, el nombre de la instrucción que debía ejecutar la unidad **aritmético-lógica** y la posición en la memoria en la que se ubica la información y se guardarán los resultados. Las instrucciones que conforman el procedimiento de operaciones que se deben realizar también se almacena en la memoria.

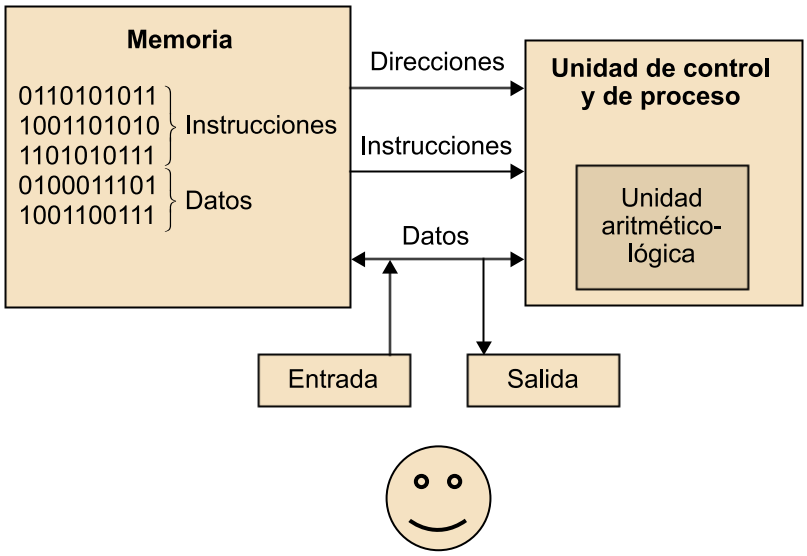
Ved también

Se definirá con más detalle el software en el subapartado 3.2 de este módulo.

Mediante las instrucciones, los programadores definen lo que deben hacer los ordenadores durante la ejecución de los programas.

La siguiente figura muestra un esquema de esta arquitectura.

Arquitectura Von Neumann



Incluso los ordenadores actuales se basan en esta propuesta, a pesar de haber evolucionado y haberse añadido nuevos elementos.

3.1.1. El microprocesador

La unidad de control y de proceso[5] es el elemento central de los **microprocesadores**. El microprocesador es el encargado de la preparación del hardware para la ejecución de tareas, así como de controlar la ejecución y ubicar los resultados en el lugar especificado en las instrucciones. Uno de los factores a la hora de valorar el rendimiento de un ordenador es medir las prestaciones del microprocesador.

(5)En inglés, *central processing unit* (CPU).

Entre los factores que permiten evaluar el **rendimiento de un microprocesador** destacamos la velocidad, el número de bits con los que es capaz de trabajar o el número de operaciones sobre números decimales capaz de hacer en un segundo.

A continuación, haremos una descripción más detallada. La **velocidad del microprocesador** depende de la frecuencia de funcionamiento del reloj que éste incorpora. Esta velocidad se mide en hercios y durante decenios no se llegó a superar el megahercio (o MHz, es decir, el millón de hercios). En la última década del siglo XX, se superó la barrera del gigahercio (o GHz) y las frecuencias actuales tienen dificultades tecnológicas para ser superadas.

El **número de bits de trabajo** también es un factor determinante del rendimiento.

Ejemplo sobre el número de bits de trabajo

Un procesador de 8 bits podrá hacer operaciones con números de 8 bits (por ejemplo, números enteros entre –127 y 128). Si se quiere llevar a cabo una operación con un número mayor, habrá que dividir éste en dos números de 8 bits, y en consecuencia el microprocesador empleará el doble de tiempo que al realizar la operación con un único número de 8 bits. Los procesadores actuales trabajan con grupos de hasta 64 bits.

Finalmente, el **número de operaciones con números decimales por segundo**[6] también es un factor importante. Hoy en día, los microprocesadores llegan a rebasar los gigaflops (miles de millones de operaciones por segundo).

(6)En inglés, *floating point operations per second* (flops).

La siguiente tabla muestra algunos de los microprocesadores de la casa Intel usados en los ordenadores personales, con el año de introducción en el mercado. También se muestra el número de transistores que lo forman, para dar idea de cómo se ha incrementado la complejidad en el diseño de los microprocesadores.

Año	Nombre	Bits de trabajo	Megahercio[7]	Transistores
1978	8086	16	8	29.000
1982	80286	16	12,5	134.000
1988	80386	32	25	275.000
1989	80486	32	50	1.200.000
1993	Pentium	32	66	3.100.000
1997	Pentium II	32	300	7.500.000
1999	Pentium III	32	800	28.000.000
2003	Pentium IV	32	2400	42.000.000

(7)De cada procesador ha habido versiones con diferentes velocidades; mostramos un valor medio.

A partir del 2003, al tener dificultades tecnológicas para aumentar la velocidad de reloj, se pensó en diseñar microprocesadores que incluyen, en realidad, más de un microprocesador.

3.1.2. La memoria

Si el microprocesador es un elemento importante a tener en cuenta en los ordenadores, también lo es la memoria. Acto seguido, se detallan los tipos de memoria que se pueden encontrar en un ordenador.

La **memoria ROM**[8] es una memoria que no se borra. Este tipo de memorias sirven, por ejemplo, para almacenar procedimientos y datos fundamentales para el funcionamiento de determinados componentes del ordenador o aparatos digitales. Notad que en este tipo de memoria no se puede modificar su contenido, únicamente se puede consultar.

(8)En inglés, *Read-Only Memory*, es decir, memoria sólo de lectura.

La **memoria RAM**[9] es la más importante. Se trata del conjunto de chips que sirven para que el ordenador almacene los procesos que se ejecutan (es decir, las instrucciones que conforman los programas que se están ejecutando) y los datos que utilizan estos programas. Cuanta más memoria RAM haya y más rápido sea el acceso a ésta, mejor podrá ser el rendimiento del ordenador.

(9)Del inglés, *Random Access Memory*, memoria de acceso aleatorio.

Esta memoria RAM se vacía cuando se apaga el ordenador. Por lo tanto, es necesario disponer de memorias capaces de retener la información que contienen. El **disco duro** es el dispositivo de almacenamiento de memoria en el que se almacena el sistema operativo, el software que se utiliza, y los documentos con los que trabajaremos. Cuando se inicia el ordenador, el sistema operativo se carga desde el disco duro a la memoria RAM. Sin embargo, algunos equipos utilizan memorias tipo ROM para guardar este software (por ejemplo, los teléfonos móviles o los ordenadores ligeros).

Memoria principal y memoria secundaria

La memoria principal del ordenador está normalmente formada por módulos de memoria RAM. Los discos duros forman la denominada memoria secundaria.

La velocidad del disco duro también es un factor importante para evaluar el rendimiento de un sistema informático: cuanto más rápido se pueda ir a leer o escribir la información, mejor.

En general, la memoria RAM tiene mucha menos capacidad que el propio disco duro. Por ejemplo, el disco duro ofrece capacidades un centenar de veces superior a la memoria RAM. Uno podría pensar por qué no utilizar discos duros en vez de memorias RAM si aquéllos tienen más capacidad. La razón es el tiempo de acceso. La tecnología utilizada en los discos duros provoca que su velocidad de acceso sea mucho menor que la velocidad de la memoria RAM.

Una solución utilizada para aumentar la memoria principal del ordenador es emplear parte del espacio del disco duro como **memoria virtual** y guardar aquella información que se está utilizando pero no cabe en la RAM.

Ejemplo de uso de la memoria virtual

Si tenemos un gestor de correo abierto pero estamos retocando una fotografía, lo más probable es que el programa de retoque y la imagen estén en la RAM, mientras que el programa de correo está "descansando" en el disco duro. Una vez activamos el gestor de correo, éste pasará automáticamente a la memoria RAM (y, en consecuencia, quizás la imagen que estamos retocando se guardará en la memoria virtual del disco duro).

Finalmente, la **memoria caché** permite tener más próxima al microprocesador aquella información que se utiliza más a menudo. Se encuentra ubicada dentro de la propia CPU, al lado del procesador, de manera que acceder a información de la memoria caché lleva menos tiempo que acceder a la información que se encuentra en la RAM (y, evidentemente, ¡menos tiempo que para acceder a la memoria virtual del disco duro!).

Ejemplo de uso de la memoria caché

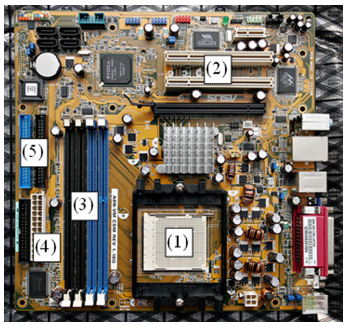
Siguiendo con el ejemplo anterior, la porción de la imagen que estamos retocando podría estar en la memoria caché, mientras que el resto de la imagen se encuentra en la RAM.

También existen otros dispositivos de almacenamiento como los DVD o las memorias flash.

3.1.3. La placa base

El microprocesador y la memoria se encuentran acoplados a la placa base[10]. Esta placa contiene una ubicación para cada uno de los elementos básicos del sistema informático: bancos para ubicar chips de memoria RAM, conexiones para el disco duro, puertos para los dispositivos de entrada y salida, etc.

[10]También conocida como placa madre, según el nombre en inglés *motherboard*.



Esquema de una placa base habitual. **1)** Espacio para el microprocesador. **2)** Ranuras para placas de extensión para conectar dispositivos. **3)** Espacio para chips de memoria RAM. **4)** Conector para la fuente de alimentación eléctrica. **5)** Conectores para el disco duro.

3.1.4. Dispositivos de entrada y salida de información

Nombramos ahora los dispositivos más importantes de entrada y salida de información. Gracias a ellos, podemos introducir información en el sistema informático (en el caso de los dispositivos de entrada) o ver la información (en el caso de los de salida).

El **teclado** es un dispositivo de entrada que permite introducir texto, por ejemplo, en un documento, en el navegador, etc. Dispone de teclas distribuidas en áreas (área numérica, alfanumérica, teclas de navegación, etc.), y algunos teclados más modernos disponen de otros botones y teclas para hacer diferentes tareas (subir o bajar el volumen del sonido, apagar el ordenador, etc.). Aunque lo más habitual es disponer de un teclado que se conecta a la unidad central con un cable, podemos optar por un teclado sin hilos. La idea es que lo que va conectado al ordenador no es el propio teclado, sino un receptor.

El **ratón** es un dispositivo de entrada que se inventó a finales de 1960 y fue olvidado en un rincón, hasta que la casa Apple lo popularizó en sus ordenadores. El ratón permite desplazar un puntero por la pantalla al ser movido por una superficie plana. Podemos activar acciones haciendo clic con sus botones o moviendo una ruedecita de desplazamiento que contiene. Podemos encontrar dos tipos de ratón, como en el caso del teclado: ratones con cable y ratones inalámbricos. Por otra parte, según la tecnología que utilizan para detectar el movimiento, tenemos los ratones mecánicos (de bola) y los ratones ópticos. La ventaja de éstos respecto a los primeros es que no se ensucia su mecanismo interno y siempre disfrutaremos de un movimiento preciso con ellos, mientras que con los ratones de bola a menudo deberemos sacar la bola y limpiar los ejes internos.

Un **escáner** es un dispositivo de entrada que permite digitalizar (pasar al ordenador) cualquier documento en papel, como una fotografía, un dibujo, una diapositiva, etc.

El último dispositivo de entrada que describimos es la webcam o **cámara web**, que va conectada al ordenador, con lo que su campo de alcance de toma de imágenes es bastante limitado. Se utiliza básicamente en comunicaciones de videoconferencia.

La **pantalla** o monitor es el dispositivo de salida más básico e importante del ordenador. Su dimensión se mide, como todos los televisores, con el número de pulgadas en diagonal. Así pues, encontramos monitores de 15", 17", etc. La pantalla es una parrilla que contiene miles de píxeles para representar imágenes y textos. El número de píxeles que pueden representar es un factor a tener en cuenta: cuantas más filas y columnas tenga, más elementos al mismo tiempo se podrán visualizar y más cómodo será trabajar con determinadas tareas.

Ved también

Entraremos más en detalle sobre cómo representar las imágenes en formato digital en el módulo "La World Wide Web".

La **impresora** es un dispositivo de salida que permite sacar en papel todo aquello que hacemos con el ordenador. Existen diferentes tecnologías de impresión, entre las que conviene destacar la de inyección de tinta (la más habitual en el mercado doméstico), la láser (para grandes volúmenes de impresión) y la matricial (ruidosa, pero utilizada en casos específicos).

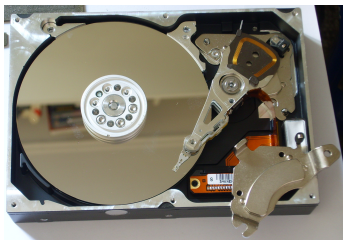
Los píxeles

Una imagen digital se dibuja en la pantalla a base de píxeles, del inglés *picture element*, en español, elemento de imagen.

3.1.5. Dispositivos de almacenamiento de información

Finalmente, describiremos algunos de los dispositivos dedicados al almacenaje de la información.

El **disco duro**, generalmente ubicado en el interior del ordenador, permite el acceso rápido a la información que se almacena. Está formado por varios discos cerrados en una carcasa que los protege.



Interior de un disco duro, en el que se muestra el brazo lector/escritor.

Los discos duros externos suelen utilizarse como extensión del disco duro, o bien para guardar los documentos que no nos caben, o bien como soporte para hacer copias de seguridad.

Las **unidades ópticas**, como los lectores/grabadores de CD-ROM y DVD, permiten la lectura y grabación de este tipo de discos. Los CD-ROM[11] tienen capacidades de grabación de unos 600 MB, mientras que con los DVD llegamos a los varios GB de espacio de almacenamiento. Inicialmente, estos dispositivos se han utilizado, aparte de como soporte de audio y vídeo digital, como soporte para las copias de seguridad de los documentos y programas. Sin embargo, la proliferación y el abaratamiento de los discos duros externos han provocado que los CD y los DVD se releguen a la multimedia y a la distribución de software.

(11)Del inglés *Compact Disc Read Only Memory*, memoria sólo de lectura en un disco compacto.

Los disquetes

Los **disquetes** son unos dispositivos muy utilizados en los años ochenta y noventa del siglo XX. Aunque algunos ordenadores aún incluyen de serie una disquetera para cubrir las funcionalidades de los disquetes, ésta ha quedado relegada con las memorias flash e incluso el envío de información en Internet.

Finalmente, las **memorias flash** son un dispositivo utilizado para transportar información de un equipo a otro. Permiten el almacenamiento de una cantidad considerable de información (actualmente del orden de los GB), pero no se suelen utilizar como soporte de copia de seguridad por su baja fiabilidad a largo plazo y porque la cantidad de información que se puede llegar a almacenar en el disco duro es considerablemente más elevada.

3.2. El software

El software es un **conjunto de instrucciones** que indican al procesador lo que debe hacer. Estas instrucciones se utilizan para manejar los datos, codificados en binario como hemos visto, de manera conveniente para resolver problemas.

Es interesante hacer notar que el software también se encuentra codificado en binario y almacenado en memoria junto a los datos.

En realidad, no puede ser de otra manera, ya que como hemos explicado, los ordenadores sólo entienden conceptualmente de unos y ceros. Por lo tanto, al igual que en código ASCII se codifica un carácter en una serie de unos y ceros para almacenarlo en memoria, se hace lo propio con una instrucción que se debe ejecutar (por ejemplo, la instrucción sumar).

Si todo es binario y todo está almacenado en memoria, ¿cómo sabe un ordenador qué es una instrucción y qué es un dato? Se trata de una buena pregunta, porque en realidad no lo sabe. En un dispositivo de almacenamiento, por ejemplo un disco duro, sólo se puede identificar unos y ceros y no se puede distinguir cuáles corresponden a instrucciones y cuáles a datos. La clave está en el orden en el que se le pasan los bits al procesador.

El procesador se construye de manera que cada instrucción se representa con un conjunto de bits. Además, cada instrucción acepta un conjunto determinado de argumentos después de la instrucción que identifican los datos.

Imaginemos una situación en la que queramos sumar dos números con un procesador que tiene la siguiente instrucción:

SUMAR DATO1, DATO2, RESULTADO

Siguiendo el orden predeterminado, el procesador espera primero el código binario de la instrucción y, a continuación, la dirección de memoria en el que está almacenada la primera cifra seguida de la segunda. Finalmente, recibe la dirección en la que se guardará el resultado. De esta manera, el procesador, dependiendo del orden en el que llegan los bits, algunas veces los interpretará de una manera (instrucciones) y otras veces, de otra (datos).

Si, por la razón que fuera, no se le envían al procesador las cosas en el orden correcto, dará un comportamiento erróneo o un error. Nótese que cada procesador normalmente tiene su conjunto de instrucciones codificadas de una manera concreta. Por ello, el software creado para una arquitectura específica

(de las que el procesador es una parte importante) no funciona en otra arquitectura. Un software para la arquitectura i386 de la familia Intel (la de los IBM PC tradicionales) no funcionará en una arquitectura Motorola, usada en los antiguos Macintosh de Apple. La solución consiste en adaptar el software para que siga las pautas de cada arquitectura.

La característica más importante que diferencia el software del hardware es su **flexibilidad**. El hardware, y no sólo el de ordenador, es específico para cada tarea. Por ejemplo, la Pascalina que ya conocemos permitía hacer (algunos) cálculos, pero nada más, al igual que una máquina exprimidora de fruta sólo permite exprimir fruta. Se puede tratar de hacer otras cosas con la máquina exprimidora, como freír un huevo, pero el resultado no será satisfactorio.

El software es muy versátil, debido a que su tarea es manejar (tomar, transformar, guardar) datos binarios. En otras palabras, con un mismo ordenador y diferente software podemos resolver un gran conjunto de problemas. Y es que una vez que un problema se ha conseguido transformar en datos binarios, podemos resolver el problema aplicando una serie de instrucciones sobre ese conjunto de datos.

Los edificios inteligentes

Los edificios inteligentes cuentan con ordenadores que toman datos del edificio (personas que entran y salen del edificio, temperatura, humedad, luminosidad, etc.) mediante sensores y otros dispositivos. Estos datos son convertidos en binarios, y posteriormente almacenados y tratados. Así, se puede conocer fácilmente el número de personas que han entrado y salido, por ejemplo, en las horas punta o ajustar la temperatura o la luminosidad que debe haber en cada espacio para ahorrar energía. Y esto es sólo el principio, ya que se nos podrían ocurrir muchas tareas más, incluso cuando los diseñadores originales del software ni las pudieran imaginar. Por ejemplo, se podría añadir funcionalidad para que el ordenador central, diseñado en la década de los ochenta, cuando todavía no existía la tecnología de la telefonía móvil, permitiera reservar salas de reuniones mediante el envío de un mensaje SMS.

Pero no todo son ventajas. La flexibilidad también conlleva que el software sea cada vez más complejo, ya que cada vez se quieren programas que realicen tareas más complicadas.

La complejidad del software

En el caso de los edificios inteligentes, si al principio sólo se quería conocer el número de personas que entraban y salían de un edificio, más adelante se desarrolló un sistema para ajustar de manera inteligente la temperatura según la ocupación del edificio, para terminar queriendo un sistema que acepte reservas de salas mediante mensajes de telefonía móvil.

Todo esto provoca que el software se vuelva cada vez más complejo: el número de factores que hay que tener en cuenta crece, así como las situaciones y las reglas... Con tantos factores que considerar, el hecho de crear software se ha convertido en una de las actividades que requieren mayor ingenio humano. Si no fuera así, probablemente lo crearían máquinas de manera automática.

3.2.1. Algoritmos

¿Cómo se crea el software? Ya hemos indicado que el software está pensado principalmente para resolver problemas. La manera en la que podemos resolver estos problemas es mediante el uso de **algoritmos**.

> Un **algoritmo** es un conjunto finito de pasos que sirven para resolver un problema o llevar a cabo una tarea.

Un algoritmo es como una especie de receta, como las de la cocina, pero en este caso informáticas. Así, imaginemos que quisiéramos cocinar unas sabrosas torrijas para degustarlas a la hora de la merienda. Al ir a un libro de recetas de cocina cualquiera, nos encontraríamos con una receta como la que sigue:

En la imagen, se pueden ver unas torrijas, listas para ser degustadas, también conocidas como tostadas francesas. Fuente: Wikipedia. Licencia: Creative Commons Atribución 2.0.

1) Mezclar 2 huevos levemente batidos con una cucharadita de vainilla, media de canela y una taza de leche.

2) Mojar 6 rebanadas de pan en la mezcla.

3) Freír en un poco de mantequilla hasta que se doren.

4) Servir el pan con un poco de sirope de caramelo.

Los algoritmos comparten con las recetas de cocina el hecho de que están escritas con un **lenguaje natural**. Esto es una ventaja, porque una vez que se ha aprendido a leer y se tienen los conocimientos básicos de los ingredientes que se suelen encontrar en la cocina tradicional, se puede entender una receta sin mayores problemas. Pero, a su vez, el uso del lenguaje natural resulta problemático. Es un problema, porque el lenguaje natural es **ambiguo**. Así, cuando se pide batir levemente dos huevos, ¿a qué se refiere con "levemente"? ¿A qué velocidad significa "levemente"? También están esos detalles relacionados con las cantidades que todos conocemos de nuestras experiencias culinarias, como el "agréguese una cucharada de sal... ". ¿Se trata de una cucharada sopera o de una de café?

En general, los humanos tenemos cierta capacidad (aunque limitada) para dirimir estos problemas asociados a la ambigüedad del lenguaje. Y esta capacidad se ve aumentada en el caso de la cocina, porque se suele degustar el producto mientras se cocina o justo al terminar. Pero las máquinas no tienen esta capacidad. Las máquinas no entienden bien *levemente* o *un poco*, salvo que se sea más concreto. Y ahí es donde, en el software, entran los programas. Los programas deben verse como una **descripción formal de un algoritmo**. Para

tal fin, en vez de valernos del lenguaje natural, utilizaremos una serie de lenguajes pseudomatemáticos llamados **lenguajes de programación**, que explicaremos más adelante.

3.2.2. Diseño del software

Cuando se crea un software, el primer paso suele corresponder a la descripción del algoritmo en lenguaje natural. Generalmente, las personas que quieren utilizar el software no saben de lenguajes de programación. Por otro lado, los ingenieros de software saben programar, pero muchas veces no entienden el problema al que se enfrentan.

Para aclarar este punto, imaginemos a un chef que quiere una máquina de hacer torrijas. Es evidente que el chef sabrá hacer torrijas, pero es improbable que tenga conocimientos avanzados en programación de software. Para tener un programa que haga torrijas llamaría a alguien con conocimientos de programación. Sin embargo, es muy probable que esta persona experta en programación no haya hecho una torrija en su vida.

La manera que el chef y el ingeniero de software tienen para entenderse será mediante un algoritmo como el que hemos visto con anterioridad, especificado mediante lenguaje natural. El chef y el ingeniero de software quedarán para indicar las especificaciones y requisitos del programa. El nombre específico que recibe el ingeniero de software en este caso suele ser el de **analista de software**, ya que su tarea es realizar un análisis de los requisitos y de cómo afrontarlos para codificarlos convenientemente en el programa que quiere el chef.

En general, la siguiente etapa suele consistir en tomar los requisitos del análisis del software (plasmados en un documento) y crear un documento de diseño del programa. Básicamente, esto consiste en tomar los algoritmos de la etapa de análisis y organizarlos de manera que el software sea sencillo de realizar, sin perder a su vez flexibilidad para que en el futuro se puedan realizar mejoras. Y es que cuanto mejor diseñado, menos errores tendrá el programa y más fácil será modificarlo en el futuro, incluyendo nuevas funcionalidades. Como consecuencia de este trabajo, se creará otro documento en el que los algoritmos estén organizados convenientemente.

Será tarea de un programador transformar estos algoritmos en un programa de software, utilizando para ello un lenguaje de programación. Si la tarea de elaborar un algoritmo no es sencilla, tampoco lo es la transformación a un lenguaje de programación. Se debe eliminar la ambigüedad, ya que los programas deben ser precisos. En el caso de que no se haga correctamente, el software contendrá errores. Imaginemos las trágicas consecuencias de haber indicado mal el tamaño de las cucharadas de azúcar para nuestras torrijas. ¡Sería un desastre!

Errores del software

Errores informáticos se pueden encontrar en todos los sitios, pero los aeroespaciales son de los más conocidos. Por ejemplo, en 1972, la sonda Mariner 1 se desvió del curso establecido y tuvo que ser destruida debido a que una fórmula escrita a mano fue trasladada erróneamente al lenguaje de programación. Otro caso, éste en 1999, tuvo como protagonista a la Mars Climate, que se estrelló contra el suelo de Marte debido a un error en la conversión de millas a kilómetros.

Como vemos, es común que el software tenga errores, y para aplicaciones algo complejas –casi todas las que utilizamos a día de hoy– es imposible asegurar que un software esté libre de ellos. Por este motivo, la creación del programa también consiste en **probarlo** –comprobar si tiene errores– y **validarlo** –asegurar que al final del proceso se obtiene lo que queríamos al principio y no otra cosa diferente–. Antes de que un software se publique definitivamente, suele pasar por un período en el que se comprueba que hace lo que debe hacer. En este sentido, la programación se parece mucho al quehacer habitual de la cocina, ya que toda receta se prueba una vez elaborada, y generalmente las recetas más innovadoras se suelen probar en círculos íntimos antes de ser presentadas en grandes ocasiones.

Y aun así, es común que con el paso del tiempo aparezcan errores que no se habían detectado o se quiera que el software tenga nuevas funcionalidades que no se habían incluido anteriormente. Es por ello que la última etapa de la vida de un programa es la del **mantenimiento**. Durante toda esta fase, los ingenieros de software gestionan el programa eliminando errores y añadiendo nuevas funcionalidades. Se trata de la etapa más larga, ya que mientras las anteriores se suelen llevar a cabo en cuestión de meses o de unos pocos años, un software puede estar en fase de mantenimiento durante años, en algunos casos, incluso décadas.

Ejemplo de software longevo

El caso más conocido de software longevo se puede encontrar en el sector de la banca, en la que existe una gran cantidad de software de la década de los setenta. Un ejemplo de funcionalidad que se debió añadir a este software es la conversión de las antiguas pesetas al euro, la nueva moneda única europea. Para tal fin, los programadores encargados del mantenimiento tuvieron que modificar los programas, y probar otra vez las nuevas versiones del programa para constatar que no tuvieran errores.

El modelo de desarrollo del software recibe el nombre de **modelo de desarrollo en cascada**, porque las fases se realizan una detrás de otra sin solapamientos. Éste es el modelo que se utiliza de manera tradicional en la gran industria del software.

- La "definición de requerimientos" es lo que en nuestro caso ha hecho el analista de software con el chef en un algoritmo.

- Tomando los requerimientos como punto de partida, un diseñador de software crea un diseño sobre papel, como un arquitecto dibuja unos planos. Generalmente, es una buena práctica dividir el problema general en sub-

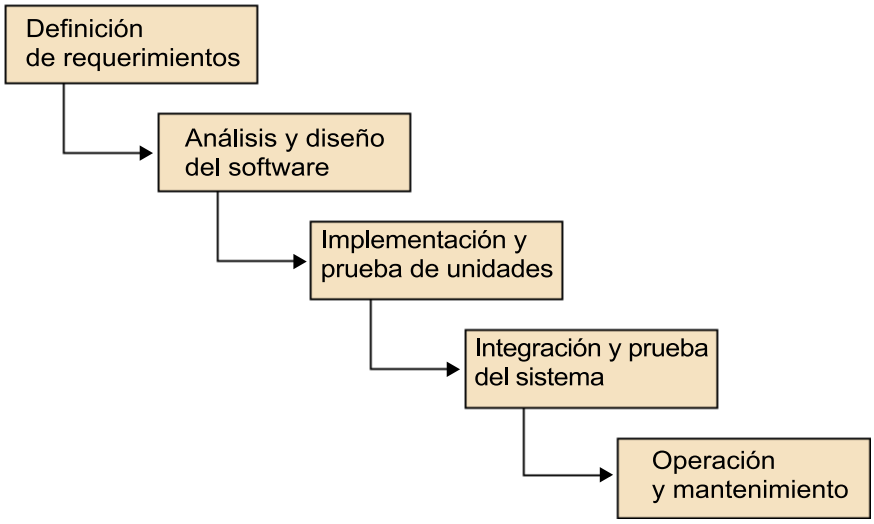
problemas, más manejables y abordables. Estos subproblemas reciben el nombre de unidades.

- En la tercera etapa, se tomará este diseño y se implementará mediante el uso de un lenguaje de programación. La división en unidades permite probar cada unidad de manera aislada y comprobar que cada una de ellas hace lo que queremos.

- En la cuarta etapa, se toman todas las unidades y se integran en un único programa. Es ahora cuando hay que probar el sistema completo para comprobar si es correcto.

- Después de probarlo, se entrega al usuario (entra en operación) y se corrigen los errores que éste pueda notificar.

Modelo cascada puro o secuencial para el ciclo de vida del software



Fuente: Wikipedia. Licencia: Creative Commons Atribución CompartirIgual.

El modelo en cascada es sencillo de llevar a cabo y de entender, pero tiene también sus aspectos negativos. El primero es que desde la primera fase hasta que se obtiene el programa, puede pasar mucho tiempo. Y sólo entonces es cuando le podemos entregar al cliente el programa definitivo. Si se da la circunstancia de que en cualquier eslabón de la cadena la información no ha fluido de manera conveniente o que se ha desviado del propósito inicial, se podría tener un software que funciona pero que no hace lo que el cliente quería. Para evitar esta situación, existen modelos de desarrollo más flexibles que obtienen resultados intermedios con anterioridad, de manera que el cliente pueda ver la evolución y dictaminar si ésta es de su agrado o no.

3.2.3. Evolución de los lenguajes de programación

En un principio, el trabajo de implementación de los programadores era muy arduo, porque las máquinas sólo entienden unos y ceros. Así, su trabajo consistía en tomar el algoritmo deseado y transformarlo en **código máquina**, que es el nombre que recibe la secuencia de unos y ceros de los programas.

Ejemplo de código máquina

El código máquina de dos números que se suman podría ser el siguiente: 1100 0100 0010 1001 0100 0100.

Trabajar de esta manera con unos y ceros directamente es, como os podéis imaginar, una tarea tediosa. Es difícil ver qué es lo que se está haciendo, y es difícil identificar si hay algún error. En programas escritos directamente en código máquina, la tarea de encontrar un error (conocida como **depuración**) se convierte en una tarea titánica. Al hecho de trabajar con unos y ceros se lo conoce como trabajar al nivel de máquina; ya que es como si estuviéramos conversando a su misma altura.

Para no tener que trabajar al nivel de máquina, se inventó un **nivel de abstracción** superior. Este nivel de abstracción permite que la tarea de programación sea más sencilla: en vez de unos y ceros se trabaja con representaciones más abstractas de las instrucciones. Así, surgió el **lenguaje ensamblador**.

Ejemplo de lenguaje ensamblador

La instrucción para el código máquina de dos números que se suman (1100 0100 0010 1001 0100 0100) podría ser la siguiente: *add $1 $2*.

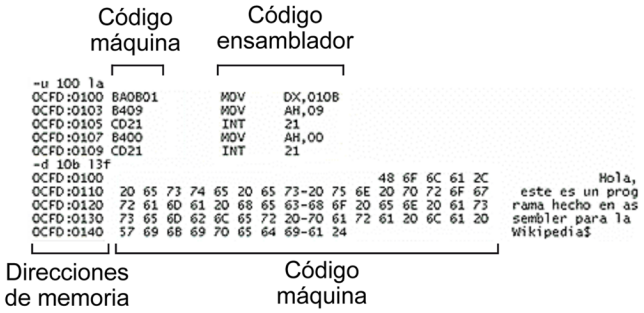
Habréis podido ver que se trata de un pseudolenguaje mitad matemático, mitad inglés, que parece indicar que se trata de una operación de suma (*add*) de dos números. En vez de indicar los dos números directamente, se indican las posiciones en las que están almacenados esos dos números en memoria (en 1 y 2). Cabe notar que no se indica la dirección en la que se debe guardar el resultado. Algunos lenguajes en ensamblador guardan el resultado de la operación en el primer operando. También habrá otras instrucciones, como una operación para introducir en una posición de memoria un número *store $1, valor*. Mediante esta instrucción, por ejemplo, se podría introducir en la dirección $1, el valor –4.

Aun así, el lenguaje ensamblador tiene una filosofía similar al código máquina; simplemente es más legible. Los informáticos dirían que se trata de un lenguaje muy cercano a la máquina, ya que la transformación en binario es inmediata. Esto tiene sus puntos positivos, pero también sus aspectos negativos. En el lado positivo cabe mencionar el hecho de que es muy eficiente, ya que se está usando directamente el conjunto de instrucciones que utiliza el procesador. Al trabajar en un nivel de abstracción tan cercano a la máquina, se pueden optimizar las operaciones, obviar las redundancias y aprovechar el conocimiento de los pormenores del procesador. Sin embargo, aunque sea mejor que trabajar al nivel de unos y ceros, no deja de ser una manera muy tediosa de indicar secuencias de instrucciones; sigue siendo muy fácil cometer errores. Además, diferentes procesadores –por ejemplo, de diferentes fabricantes– pueden tener un conjunto de instrucciones distintas, por lo que el programa en ensamblador puede que sólo sirva para un procesador específico.

Programa en ensamblador

Para aclarar un poco más los aspectos relativos al lenguaje ensamblador y sin entrar en detalles, en la siguiente figura se muestra una parte de un programa en ensamblador, con su correspondiente código máquina. El código máquina se representa en hexadecimal para simplificar el ejemplo. Podéis observar que las primeras cinco líneas corresponden a instrucciones en lenguaje ensamblador, pero de la séptima a la undécima los unos y ceros no se corresponden con instrucciones, sino con datos. Se trata de un texto codificado en ASCII.

Lenguaje máquina y lenguaje ensamblador para la arquitectura Intel 8088



Fuente: Wikipedia. Licencia: GFDL y Creative Commons Attribution ShareAlike 3.0.

Debido a la laboriosidad del código ensamblador, empezaron a surgir con el paso del tiempo **lenguajes de programación** de mayor nivel de abstracción, es decir, más cercanos al humano (al programador). Notad que cuanto más se alejan de la máquina (de los unos y ceros) y más se acercan al lenguaje natural, mayor es el nivel de abstracción. Por un lado, se gana en facilidad de implementar los algoritmos; pero, por otro, se pierde la eficiencia que se ganaba al ajustar los algoritmos a la máquina. Hoy en día, casi todas las aplicaciones, salvo cuestiones muy concretas que requieran de mucha eficiencia, como las tareas del sistema operativo, se realizan en lenguajes de alto nivel.

Ejemplos de lenguaje de programación

Existen muchos lenguajes de programación; los más populares a día de hoy son Java, C, C++, COBOL o Pascal, pero hay otros miles más.

Los lenguajes de programación permiten al programador centrarse en la implementación del algoritmo y olvidarse de las particularidades de la máquina.

Mediante estos lenguajes se crean programas menos complejos de leer y de escribir, y a su vez, menos propensos a errores, lo que permite tener aplicaciones más potentes –en el sentido de que pueden realizar tareas más complejas–. Sin embargo, son programas que no están optimizados tal y como podríamos hacer si se hubiera programado en lenguaje ensamblador. Pero la optimización no suele ser una prioridad en la mayoría de los casos para los programadores. Esto es debido a que en los últimos treinta años la potencia de los ordenadores (la velocidad de procesamiento, la capacidad de almacenamiento, etc.) se ha duplicado aproximadamente cada dieciocho meses. Eso significa que no tiene mucho sentido optimizar un programa utilizando el tedioso código ensamblador teniendo en cuenta que el coste de comprar una máquina más potente es mucho menor que el de tener a programadores optimizando los programas.

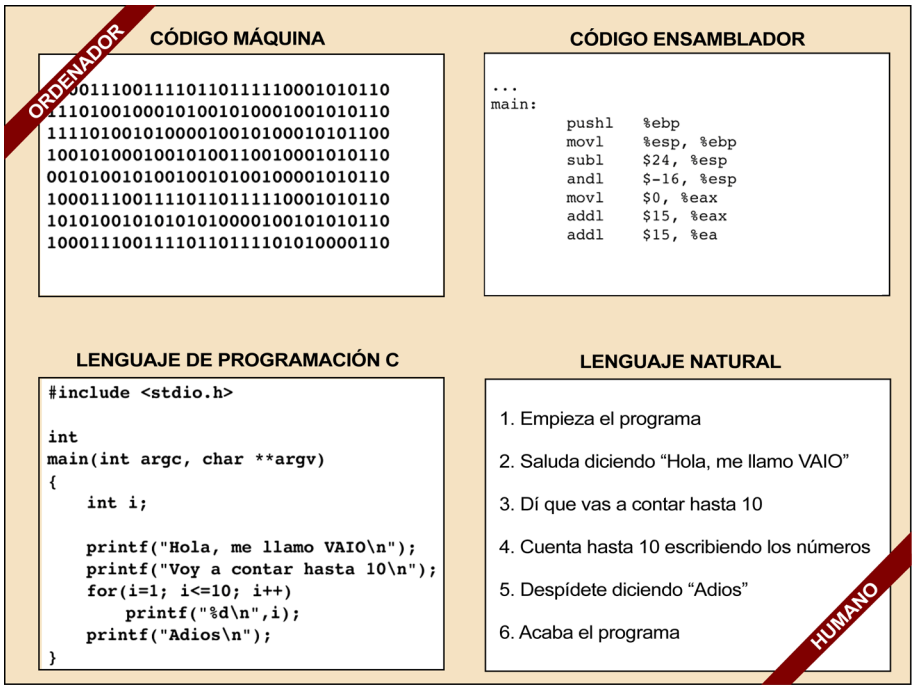
Para mostrar la conveniencia de los lenguajes de programación, se muestra a continuación un programa ejemplo, tomado de la Wikipedia, e implementado en el lenguaje de programación Pascal. Este programa realiza una suma entre dos números enteros dados por el usuario. Además de instrucciones y datos,

el programa incluye comentarios (se pueden identificar fácilmente, porque en Pascal los comentarios se encuentran entre llaves). Estos comentarios no son ni instrucciones ni datos; simplemente ayudan al programador actual (y a los futuros) a entender qué es lo que hace el programa.

```
program suma;
var x,s,r:integer; {declaración de las variables}
begin {comienzo del programa principal}
   writeln('Introduzca 2 números enteros'); {imprime el texto}
   readln(x,s); {lee 2 números y los coloca en las variables x y s}
   r:= x + s; {suma los 2 números y coloca el resultado en r}
   writeln('La suma es ',r); {imprime el resultado}
   readln;
end. {termina el programa principal}
```

En un lenguaje de programación de alto nivel, el texto que el programador genera se conoce como **código fuente**. Para poder estudiar un programa es indispensable contar con el código fuente. Esto es debido a que con la complejidad de los programas actuales, el código binario tiene una extensión tal que haría falta muchísimo tiempo y esfuerzo para descifrarlo. Sin embargo, si se ha utilizado un lenguaje de alto nivel, esto es al menos más sencillo.

En la figura siguiente exponemos la evolución que han sufrido los lenguajes de programación al crearse varios niveles de abstracción. Partiendo de nuestro algoritmo hasta llegar al código máquina que entienden los ordenadores, podemos ver los niveles intermedios (código fuente, en este caso en el lenguaje de programación C, y código ensamblador). Al proceso de transformar el algoritmo en código fuente se lo conoce como **implementación** y, como hemos comentado, lo realiza un humano. La transformación de código fuente en código máquina, sin embargo, es un proceso que se puede automatizar, incluso obviando el paso intermedio de transformar el código fuente en código ensamblador. Así, esta tarea es responsabilidad de un programa especial conocido como **compilador** y recibe el nombre de **compilación**. Cada lenguaje de programación tiene un compilador específico que transforma las instrucciones de la sintaxis del lenguaje de programación en el código máquina final.



Niveles de abstracción. (c) Enrique Soriano. Licencia: GFDL y Creative Commons Attribution ShareAlike 3.0.

3.3. Tipos de software

Llegados a este punto, tenemos unas nociones básicas suficientes de cómo desarrollar un programa nuevo. Así, si quisiéramos tener un juego propio de buscaminas, empezaríamos por aprender un lenguaje de programación de alto nivel, estudiaríamos el problema para obtener los algoritmos del juego y empezaríamos con la implementación.

Sin embargo, hay una serie de detalles que no se han abordado y que son de gran importancia, como la interacción con el usuario. Volviendo al ejemplo del buscaminas, además de los algoritmos del juego en sí, habría que incluir algoritmos para capturar las decisiones del usuario, en particular si éste ha pulsado ciertas teclas, si ha movido el ratón, etc.

Esto, en realidad, no debería ser mayor problema, ya que no dejan de ser más algoritmos que se pueden implementar con el mismo lenguaje de programación recién aprendido. Imaginemos que quisiéramos que nuestro buscaminas funcionara para muchos tipos de ratón diferentes, idealmente para todos. ¡Deberíamos implementar las instrucciones para todos y cada uno de los ratones existentes! O lo que es peor, mantener y modificar el juego para todos los ratones habidos y por haber.

Todo esto no tiene mucho sentido, sobre todo porque la idea es realizar un buen juego de buscaminas y no andar liados con el hardware de los usuarios. Es por esta razón, y por otras que veremos a continuación, que el software no se hace encargándose de todo, desde la funcionalidad que brinda al usuario hasta la interacción con los dispositivos conectados al ordenador. También

existen niveles de abstracción a la hora de crear funcionalidad y por ello el software se divide en dos grandes familias: el software de sistemas y el software para usuario final.

3.3.1. El software de sistemas: el sistema operativo

El **software de sistemas** tiene como objetivo interactuar con el hardware y proveer servicios y funcionalidad a otras aplicaciones. De esta manera, se trata de un programa que está más centrado en el ordenador que en el usuario. Aunque existen muchos software de sistemas, como los antivirus o los servidores, el software de sistemas más conocido es el sistema operativo.

Los **sistemas operativos** son programas que crean un nivel de abstracción superior sobre el hardware.

Dicho en otras palabras: el sistema operativo se encarga de comunicar los programas para usuarios y otro software de sistema con el hardware del ordenador. Siguiendo con el ejemplo del juego del buscaminas del subapartado anterior, el sistema operativo se encargará de la interacción con todos los posibles ratones que existan o puedan existir. Los programadores de aplicaciones para usuario final sólo deben tener en cuenta que se interactúa mediante un ratón genérico (y abstracto). Los programadores conocerán la posición del puntero y si se han pulsado los botones, independientemente del ratón usado.

El caso de las impresoras

Los creadores de los procesadores de textos como Microsoft Office u OpenOffice.org Writer no se preocupan de la impresora instalada y de cómo se envían los trabajos. Estas aplicaciones simplemente indican al sistema operativo el fichero que se debe imprimir, y será tarea del sistema operativo que eso suceda.

Además, el sistema operativo es el encargado de gestionar un gran número de otras tareas y elementos, como veremos a continuación. Por esta razón, los sistemas operativos se han convertido desde hace un par de décadas en un componente esencial de todo ordenador. Entre sus responsabilidades podemos encontrar:

- La **comunicación con los periféricos**. Es el caso que hemos visto hasta ahora; el sistema operativo tiene como objetivo que los programas de usuario puedan trabajar de manera independiente a las especificidades del hardware que se tenga. Por ello, cuando se compra un nuevo dispositivo, se deben instalar los controladores[12].

(12)En inglés, *drivers*.

Los controladores

Los **controladores** son los programas que permiten la comunicación del sistema operativo con los dispositivos.

- La **gestión de la memoria**. Es tarea del sistema operativo gestionar las diferentes memorias que podemos encontrar en un ordenador, desde la memoria principal (RAM) hasta la secundaria (el disco duro), pasando por la memoria virtual y la memoria caché que hemos visto con anterioridad. La gestión de la memoria es una tarea muy importante si recordamos que la memoria principal es muy rápida, pero muy cara y volátil, mientras que la memoria secundaria es lenta, pero barata y persistente. Es precisamente por la velocidad, que nos interesa que tanto las aplicaciones que se ejecutan como los datos se encuentren en la memoria principal. Sin embargo,

debido a su coste, generalmente no se tiene memoria principal suficiente para almacenar todo en una memoria principal y es necesario que el sistema operativo use la memoria secundaria.

- La **concurrencia de procesos**. Aunque el procesador sólo puede ejecutar una instrucción cada vez, se puede conseguir concurrencia virtual mediante el sistema operativo. En otras palabras, el sistema operativo puede conseguir que el usuario tenga la impresión de que se están ejecutando varios programas a la vez. Imaginemos que se está escribiendo una carta con el procesador de textos mientras se descarga un documento con el navegador web y se calcula mediante un explorador de archivos cuánto espacio libre queda en el disco duro. En la pantalla, se ve la ventana del procesador de textos, por lo que aparentemente se podría pensar que ésta es la tarea que tiene entre manos el sistema operativo, dejando las otras para después. Y, sin embargo, se tiene la impresión de que todas ocurren a la vez. En realidad, esta impresión es falsa: el procesador sólo puede ejecutar una instrucción a la vez, así que no puede realizar múltiples tareas de manera simultánea. Lo que sucede es que el procesador ejecuta las instrucciones a una velocidad tan elevada (lo hace a la velocidad de miles de millones de instrucciones por segundo) que nos parece que las realiza todas a la vez. Cuando las tareas del usuario dejan ocioso al procesador, éste realiza otras tareas. En nuestro ejemplo, mientras no se escribe nada en el procesador de textos, el procesador puede calcular el espacio libre en el disco duro, seguir descargando un documento con el navegador web y probablemente otras tareas de las que ni siquiera se es consciente, como ver cuánta batería nos queda en el portátil. Cabe notar que aunque actualmente la mayoría de los ordenadores tienen más de un procesador, se sigue utilizando la misma técnica explicada. El sistema operativo expulsa las tareas ociosas de los procesadores para ejecutar otras tareas pendientes. En este segundo caso hay concurrencia real entre procesos, ya que se ejecutan múltiples tareas al mismo tiempo en procesadores diferentes.

- La **monitorización y seguridad**. Los sistemas operativos modernos son multiusuarios. Esto introduce un problema de seguridad. Hay documentos personales en los que se debe restringir el acceso a otros usuarios del ordenador. El sistema operativo se encarga de esta función añadiendo metadatos a los ficheros, como el del usuario propietario. Por otro lado, para saber qué es lo que ha pasado en la ejecución de alguna tarea en especial, existen ficheros en los que se registran los acontecimientos para su posterior estudio. Es lo que se conoce como registros[13] y permite conocer qué ha sucedido y las tareas ejecutadas en un ordenador. Esto es útil, por ejemplo, para detectar si una máquina ha sido comprometida (alguien ha entrado en ella sin tener permiso de acceso).

(13) En inglés, *logs*.

- La **gestión de almacenamiento**. Los ordenadores contienen programas y datos. Ambos se guardan (almacenan) como ficheros en el disco duro. Es

tarea del sistema operativo hacer esto de manera ordenada (por ejemplo, en directorios o carpetas).

- La **gestión de comunicaciones**. Los ordenadores rara vez se usan hoy en día de manera aislada; generalmente se conectan a una red, como podría ser Internet. Para ello, se comunican mediante una serie de reglas establecidas, normalmente conocidas públicamente, que reciben el nombre de protocolos. El sistema operativo es el que se encarga del envío y la recepción de información utilizando protocolos de red y permitiendo que las aplicaciones (en realidad, los que las programan) no deban ocuparse de los detalles de la Red.

Debido a todas estas capacidades que tienen los sistemas operativos modernos, se puede entender rápidamente que son de gran provecho para crear nuevas aplicaciones de manera más rápida y sencilla.

Todo esto tiene un precio, que es el de la compatibilidad. Hemos comentado que los programadores realizan su trabajo creando código fuente para que posteriormente un programa conocido como compilador lo transforme en código máquina. Bien, pues esta etapa de compilación depende del sistema operativo, ya que el código máquina que "entiende" un sistema operativo no tiene por qué ser "comprendido" por otro. ¿Cómo se consigue esto? Principalmente mediante dos cuestiones:

- Por un lado, el lenguaje de programación usado cuenta con varios compiladores diferentes, uno para cada sistema operativo soportado. Por ello, a la hora de compilar se elige el compilador para nuestro sistema operativo, de manera que obtenemos el código máquina correspondiente.

- Por otro lado, los programadores han tenido suficiente cuidado para, además de utilizar un lenguaje multiplataforma (esto es, que se pueda compilar en diferentes sistemas operativos), no utilizar secuencias de programación o métodos que son exclusivos de un sistema operativo. Cabe notar que hay algunas funcionalidades o métodos que son específicos de un sistema operativo. La inclusión de estos métodos en el programa produciría errores en la compilación en otros sistemas operativos.

Ejemplos de dependencia del sistema operativo

El programa PADRE para la realización de la declaración del impuesto sobre la renta en España, la *suite* ofimática libre OpenOffice.org o la del navegador Mozilla Firefox tienen diferentes versiones dependiendo del sistema operativo usado.

Con anterioridad, habíamos visto que el compilador depende de la arquitectura del ordenador. Así, la transformación de código fuente a código máquina sería diferente para una máquina Intel o para una Mac. Ahora, además, vemos que el sistema operativo también es una cuestión que tener en cuenta en esa transformación.

En definitiva, los compiladores deben ser específicos para una arquitectura y para un sistema operativo.

Ejemplo de compiladores

Para que un programa creado en el lenguaje de programación Pascal pueda ejecutarse en un ordenador de la arquitectura Intel y con el sistema operativo GNU/Linux, se necesitará el compilador para Pascal para la arquitectura Intel y el sistema operativo GNU/Linux. El código máquina que generará este compilador no funcionará si sólo coincide la arquitectura (por ejemplo, en una máquina Intel con sistema operativo Microsoft Windows) o el sistema operativo (por ejemplo, en una máquina con arquitectura PowerPC con sistema operativo GNU/Linux).

3.3.2. Software para usuario final

El **software para usuario final** tiene como finalidad resolver problemas del usuario, ya sea dando soporte, ya sea mejorando el trabajo que se quiere realizar. Su funcionalidad está más relacionada con la lógica de la tarea a realizar por el usuario que con el hardware del sistema o de los dispositivos con los que el usuario está interactuando, lo que, como hemos visto, es gestionado por el sistema operativo.

Ejemplo de software para usuario final

Tomemos una analogía del mundo urbano para comprender la diferencia entre el software de sistemas y el de usuario final. En las ciudades, tenemos el servicio de limpieza y mantenimiento que realiza tareas más cercanas a la infraestructura, mientras que las tiendas y los supermercados ofrecen a los ciudadanos servicios finales. Así pues, el software para usuario final es como las tiendas o supermercados, ya que ofrecen servicios y productos directamente al ciudadano, mientras que el software de sistemas se asemeja a los servicios de limpieza y mantenimiento, que se encargan de otras tareas con las que el ciudadano no suele interactuar directamente, pero sin las que sería difícil vivir (al menos cómodamente) en una ciudad.

Generalmente, los programas de usuario final realizan un objetivo específico, como puede ser la manipulación de imágenes, la grabación de CD-ROM, la navegación por la web, etc. La lista es larga y ha habido múltiples intentos de crear una taxonomía de programas para usuario final. A continuación, ofrecemos una posible clasificación de este tipo de software:

- **Software de infraestructura empresarial**: se trata de software como bases de datos para almacenar grandes volúmenes de datos, software de flujos del proceso de negocio que ayudan a la gestión del producto o servicio que ofrece la empresa, o software de sistemas de información geográfica que permite enlazar datos con cartografía, por ejemplo para conocer la localización exacta de cada uno de los camiones que componen la flota de la empresa.

- **Software de acceso a contenidos**: mediante este software, se puede acceder a información, ya sea textual o multimedia. Tal es el caso de navegadores web para navegar por Internet o de programas de reproducción

multimedia. El software de entretenimiento como juegos o salvapantallas también se suelen incluir en esta categoría.

- **Software educativo**: como tal se conoce al software con finalidad educativa. En esta categoría, encontramos software de gestión de aulas, software de aprendizaje o de entrenamiento o software de consulta (como las antiguas enciclopedias que vendían en CD-ROM, hoy en día más en desuso debido a Internet y la Wikipedia).

Edutainment

El *edutainment* es un software de entretenimiento diseñado para educar.

- **Software de simulación**: en esta categoría, se encuadrarían los conocidos juegos de simulación (como los simuladores aéreos o los de motor), pero también hay una gran gama de simuladores científicos que se utilizan en investigación.

- **Software de desarrollo multimedia**: aquí tenemos el software de gestión de imágenes, y de creación y edición de contenidos multimedia (vídeo, imágenes, sonido, etc.). Este campo está en gran apogeo en los últimos tiempos y está consiguiendo introducir grandes cambios en la industria cinematográfica.

- **Software de ingeniería**: ayudan en la creación ingenieril e incluyen el diseño –los famosos programas CAD (*computer aided design* o diseño asistido por ordenador), utilizados por ingenieros mecánicos, ingenieros aeronáuticos, ingenieros industriales y arquitectos–. También hay que incluir dentro de esta categoría el software de creación de programas software, entre los que se encuentran algunos de los que ya hemos hablado con anterioridad en este módulo, como el compilador.

- **Software de sistemas de información.**

Ved también

El software de sistemas de información se verá con más detalle en el apartado 5.

Un subconjunto del software para usuario final es el software integrado, como los paquetes ofimáticos. Cuentan con una infraestructura propia y una manera de interactuar con el usuario homogénea. Esto permite que, una vez que se conozca un programa de la *suite*, podamos utilizar los otros de manera más sencilla, ya que los menús, la manera de ser usado, etc., es parecida. Un ejemplo sencillo es el uso de iconos similares, mientras que entre las posibilidades más avanzadas podemos encontrar la posibilidad de integrar lo que hacemos con una aplicación en otra. Así, podríamos incluir una tabla de nuestra hoja de cálculo en nuestro documento en el procesador de textos.

Finalmente, existe un tipo de software para usuario final que también podemos hallar con bastante frecuencia: se trata de las aplicaciones a medida. Estos programas los solemos encontrar en el mundo empresarial, como por ejemplo en programas de facturación, o en la aplicación que manejan los empleados de banca, incluso el que se utiliza en un hospital. Este software es diferente

al anterior, ya que se suele diseñar y crear con la vista puesta en un usuario específico, y no para el gran público. Podríamos decir que son programas *ad hoc* creados específicamente para el usuario.

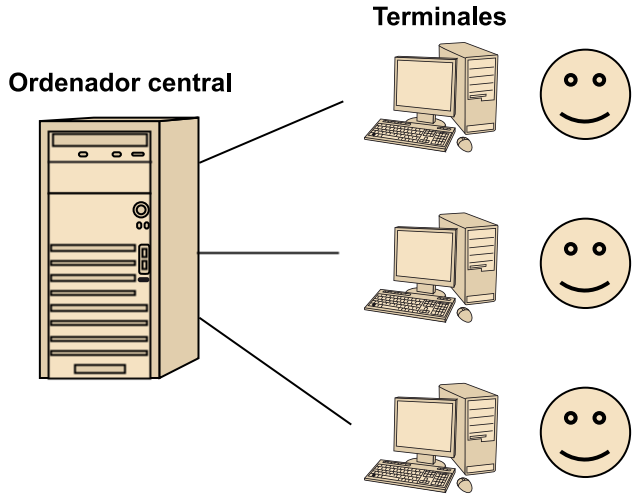
4. Tipo de ordenador

Los ordenadores personales no son extraños para nosotros. Cuando hablamos de ordenador personal nos viene a la mente la pantalla, el teclado, el ratón y una torre, grande o pequeña, que contiene el lector de DVD, las conexiones de dispositivos externos, los altavoces, etc. Sin embargo, durante la historia de la informática han ido variando los tipos de ordenador existentes. Algunos de ellos se consideran históricos y otros son muy populares en la actualidad. En este apartado, describiremos algunos de los sistemas informáticos más importantes.

4.1. Los ordenadores principales

Los primeros ordenadores tenían como objetivo la realización de cálculos sobre grandes volúmenes de datos. Durante décadas, ésta fue la principal actividad de los computadores. Llegó un momento en el que la tecnología permitió no sólo la miniaturización, sino también el hecho de poder ejecutar varias tareas a la vez mediante los sistemas operativos. Gracias a ello, un mismo microprocesador era capaz de ejecutar varios programas al mismo tiempo, sencillamente repartiendo el tiempo de proceso entre los diferentes programas que se estaban ejecutando. De la misma manera, los sistemas multi-usuario permitieron que diferentes trabajadores accedieran a un mismo ordenador principal o *mainframe* para poder utilizarlo. Para trabajar con el ordenador había que utilizar un terminal que nos conectaría remotamente al equipo, como muestra la figura siguiente.

Un ordenador principal o *mainframe* con tres terminales conectados



Actualmente, este modelo continúa siendo válido, en cierta manera: muchos de los sistemas de información que utilizamos son grandes servidores accesibles en Internet. Para acceder, sólo es necesario utilizar el navegador web (que hace de terminal).

4.2. Los microcontroladores

Un microcontrolador es un chip que contiene todas las funcionalidades básicas descritas en la arquitectura Von Neumann, es decir, tiene una CPU, una memoria y una unidad capaz de gestionar la entrada y la salida de información. Estos chips pueden ejecutar programas. Los microcontroladores son ordenadores tan pequeños que es posible integrarlos en una gran variedad de aparatos: lavadoras, móviles, máquinas industriales, hornos microondas digitales, cámaras de vídeo, etc.

Hay microcontroladores genéricos que se pueden programar y se pueden utilizar en diferentes ámbitos. Por otra parte, existen microcontroladores específicamente diseñados para un determinado entorno.

4.3. Los superordenadores

Durante los años setenta y ochenta del siglo XX, varias compañías sacaron a la luz ordenadores muy potentes, si se comparaban con los ordenadores principales habituales en las grandes empresas, y evidentemente mucho más potentes que los microordenadores personales. Estos ordenadores, llamados supercomputadores, se utilizaban para aplicaciones específicas que requerían gran potencia de cálculo: desde simulaciones atómicas hasta participar contra humanos en competiciones de ajedrez. Sin embargo, hoy en día estos supercomputadores serían poco potentes si los comparáramos con los ordenadores portátiles actuales.

Ved también

Recordad que hemos tratado los microordenadores personales en el apartado 1 de este módulo dedicado a la historia de los ordenadores.

Deep Blue

Ha habido varios ordenadores diseñados específicamente para jugar al ajedrez. Uno de los más famosos es el Deep Blue, que en el año 1997 ganó al campeón de ajedrez Gari Kasparov.

De esta manera, actualmente la idea superordenador se alcanza no con un único ordenador muy potente, sino con el trabajo colaborativo de centenares o miles de ordenadores "normales" conectados a redes de gran velocidad y rendimiento. Este concepto lo explicamos acto seguido.

4.4. Multicomputadores

Hemos visto anteriormente que se ha llegado a límites en cuanto a velocidad de reloj de los procesadores. Así pues, los nuevos ordenadores ya no aumentan su rendimiento a base de aumentar la velocidad de reloj, sino que se utilizan técnicas como la "multicomputación" para producir equipos más potentes.

Los **multicomputadores** propiamente dichos son los sistemas formados a partir de una **granja de ordenadores**, habitualmente con las mismas capacidades y características de rendimiento. De esta manera, las tareas se pueden distribuir entre los diferentes ordenadores que integran la granja y hacer el trabajo más

rápido. Por ejemplo, en una productora de cine de animación por ordenador, una escena se podría dividir en pequeños trozos, y cada uno de ellos pasaría a ser tratado por un ordenador. Por otra parte, con sistemas multicomputador se pueden llegar a atender peticiones de miles de usuarios. Por ejemplo, los grandes servicios de búsqueda de Internet disponen de decenas de miles de ordenadores para atender los millones de peticiones que reciben cada minuto.

Ejemplo de multicomputación

Los ordenadores personales con procesadores de varios núcleos son un ejemplo de multicomputación. Un microprocesador de doble núcleo contiene duplicados muchos de los elementos que forman un procesador, es decir, es como tener dos microprocesadores en uno. De esta manera, si se están realizando dos tareas al mismo tiempo (por ejemplo, trabajando con el campus virtual de la UOC y generando un pase de fotografías para ser registrado en un DVD), cada uno de los procesadores se dedicará a una de las tareas. En un sistema con un único procesador, al igual que los primeros ordenadores principales, se tardaría mucho más tiempo que con dos núcleos.

Finalmente, hay que hablar de los modelos de **computación grid**. El objetivo es también el procesamiento en un tiempo factible de tareas costosas por medio de la división del trabajo en tareas que realizarán los diferentes computadores que integran el grupo de trabajo. A diferencia del caso anterior, en la computación grid los equipos no son iguales, ni están conectados por una red de alta velocidad, ni se usan en exclusiva para la tarea: los equipos que se emplean suelen ser ordenadores personales de usuarios particulares que reciben las pequeñas tareas que realizar por medio de su conexión a Internet. Estas pequeñas tareas se van haciendo en períodos de inactividad del equipo.

Ejemplo de computación grid

Un ejemplo de computación grid es el *Search for Extra Terrestrial Intelligence* (SETI), que se basa en el análisis de señales procedentes de radiotelescopios para descubrir si alguna de éstas puede haber sido generada por seres inteligentes.

4.5. Los ordenadores portátiles

La miniaturización de los equipos permitió la realización de equipos portátiles. Históricamente, los **ordenadores portátiles** o dispositivos electrónicos, como agendas digitales o **teléfonos móviles avanzados**, han sido equipos poco potentes si se los compara con los de sobremesa. Sin embargo, la tecnología ha permitido hacer ordenadores portátiles lo bastante potentes o lo bastante pequeños como para ser considerados ultraportátiles. Estos productos están siendo habituales, por lo que conviene citarlos en este material.

Un ordenador portátil es actualmente tan potente como un ordenador de sobremesa. De todas maneras, este tipo de ordenadores no es muy cómodo de llevar siempre encima, dado su peso. Los ordenadores **ultraportátiles** son equipos más cómodos de transportar. Incluso existe una variante que permite utilizarlos como si se tratara de una libreta para tomar nota (los llamados **tablet PC**): tienen una pantalla táctil para poder escribir directamente sobre la pantalla. Los ordenadores **netPC** son ordenadores aún más portátiles (normalmente tienen dimensiones más reducidas), a cambio de tener un rendimiento menor que los portátiles y los ultraportátiles. Sin embargo, son lo bastante potentes para permitir el acceso a Internet, la visualización de fotografías y vídeos y las tareas ofimáticas más habituales.

5. Sistemas de información

Un sistema de información, en términos generales, está constituido por una serie de elementos que se relacionan entre sí para apoyar en las actividades de una institución o empresa. Los sistemas de información se encuentran dentro del software para usuario final. Los elementos de los que está compuesto un sistema de información son:

- Las personas que interactúan con el sistema.
- Los datos.
- Las actividades para procesar la información y los datos de una organización.
- El componente informático, que incluye tanto hardware como software.

En un sentido más restringido del término, a veces muchos se refieren como sistema de información a la aplicación software que realiza las tareas de almacenamiento y algunas de manipulación y procesamiento de datos e información.

Ejemplo de sistema de información

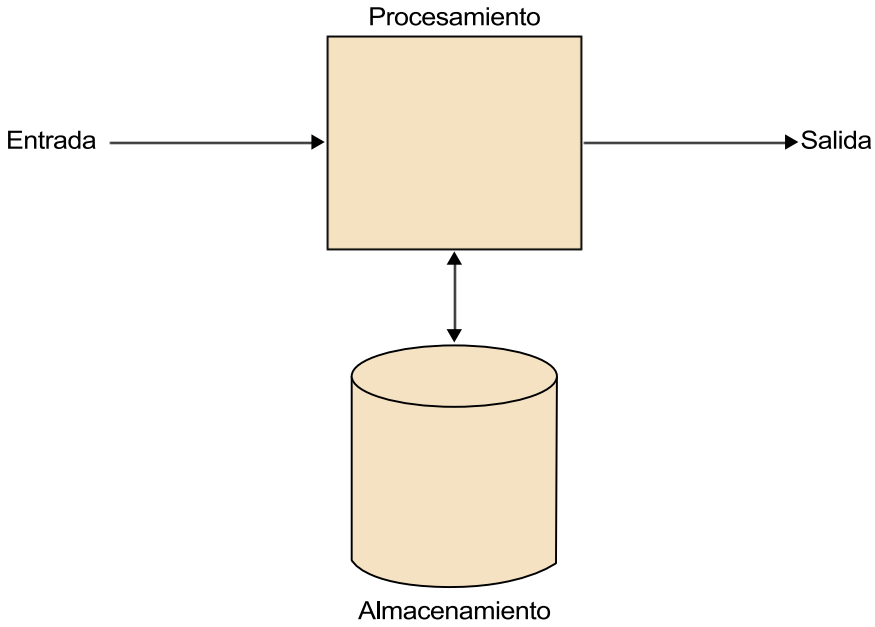
El ejemplo más paradigmático de sistema de información es el censo de una población. Herman Hollerith creó en 1890 un sistema de información para gestionar de manera automática una gran cantidad de datos que debían ser tratados. La diferencia principal de este primer sistema de información complejo con los actuales es que aquél era completamente mecánico, en vez de basarse en la electrónica. Pero el sistema de Holerith sirvió de inspiración para la computación moderna, por lo que no deja de ser lógico que el primer gran sistema de información fuera instalado en la Oficina del Censo norteamericana en 1951, el UNIVAC I, uno de los primeros ordenadores.

En la actualidad, los sistemas de información se utilizan básicamente con tres objetivos:

- Automatizan procesos.
- Tratan información para la ayuda de decisiones.
- Buscan ventajas competitivas.

Independientemente de la tecnología que se utilice, los sistemas de información constan de cuatro actividades: entrada, procesamiento, almacenamiento y salida de información, como se puede observar en la figura siguiente.

Actividades básicas de un sistema de información



Ateniéndonos a la aplicación del sistema de información, tenemos un gran número de programas dentro de este ámbito, como:

- **Programas de gestión de tiempo y recursos**. Aquí es donde podemos encontrar los famosos ERP[14] utilizados para la gestión empresarial. Sin embargo, hay otros programas como los de contabilidad o los de gestión de tareas y agenda que también se pueden englobar en esta categoría.
- **Gestión de datos**. Se trata de software que permite gestionar cantidades de datos importantes, aunque no tan abundantes ni tan variadas como para tener una base de datos específica. Así, tenemos aplicaciones de gestión de contactos, las hojas de cálculo o las pequeñas bases de datos personales.
- **Documentación**. Los más conocidos dentro de este apartado son los procesadores de texto, aunque también existe software para publicaciones, software para la creación de diagramas y software para realizar presentaciones.
- **Software analítico**. Son programas que permiten el cálculo o la manipulación de datos numéricos para su análisis, con carácter estadístico, financiero o de otro tipo.
- **Software cooperativo**. Software utilizado para comunicarse con otras personas, tanto de manera asíncrona (correo electrónico, blogs o wikis) como síncrona (sistemas de mensajería instantánea o software de telefonía IP).

(14)Del inglés *Enterprise Resource Planning*, planificación de recursos de la empresa.

La importancia que ha cobrado Internet en los últimos tiempos ha provocado que los sistemas de información se estén adaptando a un entorno distribuido, en el que la información y los procesos no se encuentran localmente.

Ved también

En el módulo "La World Wide Web" trataremos la importancia de Internet.

Resumen

En este módulo hemos visto las bases de funcionamiento de los sistemas informáticos. Hemos empezado con una visión histórica de los hitos más importantes de la historia de la computación y hemos comprobado que los sistemas informáticos tienen una historia relativamente reciente y que en los últimos años han entrado en casi todos los ámbitos de las actividades humanas.

También hemos estudiado cómo la informática trata la información. Hemos aprendido que los números y los caracteres se codifican en una serie de bits. De esta manera, la información se puede guardar utilizando diferentes tecnologías, a la vez que se permite su procesado mediante circuitos lógicos.

Una vez que hemos visto las bases del tratamiento de la información, hemos tratado las diferentes partes que forman el ordenador, a la vez que hemos hecho énfasis en aquellos aspectos del hardware que caracterizan el rendimiento y la potencia de la máquina. Finalmente, hemos clasificado los ordenadores en función de su tamaño y complejidad.

La segunda parte del módulo se ha dedicado a presentar el software. Hemos visto cómo el software tiene como objetivo resolver problemas. Así, para crear un programa, primero se especifica un algoritmo en lenguaje natural que, posteriormente, se traducirá. El siguiente paso consta del proceso de compilación, mediante el que se transforma el código fuente en código máquina (o binario), que es lo que ejecuta un ordenador.

Finalmente, hemos presentado los diferentes tipos de software que existen. Lo que se ha visto con más detenimiento es el sistema operativo, el software que gestiona gran cantidad de los servicios del ordenador. Se ha mostrado cómo el sistema operativo ofrece soporte a otros programas para los usuarios finales que, como su nombre indica, son con los que habitualmente interactúan las personas. Para acabar, se ha presentado con un poco más de detalle un tipo de software de usuario final como es el de los sistemas de información. Éstos se encuentran en todas partes en entornos corporativos y domésticos, y con el apogeo de Internet están ganando una nueva dimensión.

Actividades

1. Un amigo nuestro nos ha comentado que quiere aprender a programar. Lo primero que se le ha ocurrido es abrir con el editor de textos un archivo ejecutable de una aplicación (por ejemplo, word.exe). ¿Qué es lo que verá? ¿Podrá programar así? ¿Por qué no?

Solución

Solución de la actividad 1
Al abrir un archivo ejecutable, verá el contenido en binario del archivo. De esta manera, es casi imposible programar, ya que es ininteligible para los seres humanos.

2. ¿Qué le aconsejaríamos a nuestro amigo si quisiera programar?

Solución

Solución de la actividad 2
Para programar, lo mejor es que haga uso de un lenguaje de programación en el que podrá realizar las instrucciones pertinentes de manera más sencilla. Los lenguajes de programación permiten utilizar un lenguaje semiestructurado que es fácil de entender para los humanos.

3. Una vez que le hemos convencido, nuestro amigo ha escrito unas cuantas líneas de código fuente. Ahora quiere ejecutarlas, ¿qué debe hacer?

Solución

Solución de la actividad 3
Compilarlo, de manera que las instrucciones en lenguaje de programación del código fuente sean transformadas en instrucciones en binario entendibles por el computador.

4. Estudiad la vida de Ada Lovelace, la primera programadora de la historia. A un lenguaje de programación le han dado su nombre; investigad en qué ámbitos se suele utilizar este lenguaje.

5. Queremos incluir en binario la información de nuestro nombre (en ASCII), nuestra fecha de nacimiento (una cifra para el día, otra para el mes y una tercera para el año) y nuestro lugar de nacimiento (en ASCII). ¿Cuántos bits ocuparía toda esta información?

6. Buscad en Internet una foto del primer ratón.

7. Tenemos dos puertas lógicas AND y NOT, encadenadas una detrás de otra. Primero se encuentra la puerta AND y, a continuación, su salida, que es la entrada de la puerta NOT. ¿Qué obtendríamos a la salida de la segunda puerta si en la entrada de la primera puerta tenemos los valores "Verdadero" y "Falso"?

8. Nombrad diferentes sistemas operativos que conozcáis. Investigad en la Wikipedia las familias que existen.

9. Enumerad una docena de programas que funcionen tanto en sistemas operativos Windows como en Linux.

10. Cread un algoritmo en lenguaje natural para llegar desde vuestro lugar de trabajo hasta la cafetería más cercana.

11. Buscad en el periódico una oferta actual de un ordenador de sobremesa y de un portátil. Comparad en ambos casos los diferentes tipos de memorias que incluyen, así como su precio.

Ejercicios de autoevaluación

Indicad cuáles de la siguientes afirmaciones son correctas y cuáles son falsas.

1. Los antecesores de los ordenadores, las máquinas analíticas, fueron ideadas mucho antes del siglo XX, pero por problemas ingenieriles no pudieron ser construidas.

a) Verdadero
b) Falso

2. La primera generación de ordenadores modernos vino de la mano de los transistores.

a) Verdadero
b) Falso

3. Los entornos de ventanas fueron ideados en la década de los setenta por Xerox, aunque tardaron bastantes años en llegar al gran público.

a) Verdadero
b) Falso

4. El byte 00000111 significa exclusivamente el número 7 en binario.

a) Verdadero
b) Falso

5. El año 2009 ocupa en ASCII en total 4 bytes.

a) Verdadero
b) Falso

6. Cuando apagamos el ordenador, se borra la memoria principal (RAM).

a) Verdadero
b) Falso

7. Los programadores crean sus programas mayoritariamente en código binario.

a) Verdadero
b) Falso

8. El sistema operativo se encarga de gestionar los recursos de un ordenador.

a) Verdadero
b) Falso

9. Un algoritmo debe estar escrito en un lenguaje de programación de alto nivel.

a) Verdadero
b) Falso

10. Un *mainframe* es un ordenador central al que se le pueden conectar múltiples ordenadores denominados terminales.

a) Verdadero
b) Falso

Solucionario

Ejercicios de autoevaluación

1. a

2. b

3. a

4. b

5. a

6. a

7. b

8. a

9. b

10. a

Glosario

algoritmo *m* Conjunto finito de pasos que sirven para resolver un problema o llevar a cabo una tarea.

analista de software *m* Persona, generalmente de perfil técnico, que realiza un análisis de los requisitos que ha de cumplir un futuro programa de ordenador.

arquitectura (de ordenadores) *f* Se trata de un concepto que engloba el diseño y el funcionamiento de los ordenadores que especifican, entre otros aspectos, el formato y el conjunto de instrucciones. Existen muchas arquitecturas, pero la más popular es i386, porque es la utilizada en los PC.

byte *m* Grupo de 8 bits.

cobol *m* Lenguaje de programación. Acrónimo de Common Organization Business Oriented Language (lenguaje común orientado a la organización de negocios), aunque también se puede encontrar como Common Business Oriented Language (lenguaje común orientado a negocios). Se trata de un lenguaje de programación de finales de 1960 cuyo propósito era poder ser utilizado en cualquier ordenador. Actualmente, su utilización es bastante limitada, aunque se encuentra con frecuencia en aplicaciones bancarias y programas antiguos.

código binario *m* Véase código máquina. Sistema de numeración basado en la utilización de dos valores o bits (típicamente '0' y '1'), en los que a su vez se basa el procesamiento automático de la información. Todo lo que se procesa con un ordenador acaba siendo transformado a código binario.

código fuente *m* (También conocido como fuentes.) Se trata de las instrucciones de ordenador escritas en un lenguaje de programación. En la fase de compilación se transforma en código máquina. Para que el software sea libre, el código fuente debe ser accesible, ya que, en caso contrario, la posibilidad de realizar modificaciones, aunque no sea imposible, se dificulta sobremanera.

código máquina *m* (También conocido como código binario.) Se trata del código que los ordenadores pueden ejecutar. Consta de unos y ceros, aunque existen otros modos de representación como octal o hexadecimal. El código máquina es difícilmente comprensible para los humanos –y la creación de complejas aplicaciones casi imposible–, por lo que se crearon los lenguajes de programación de alto nivel.

compilador *m* Se encarga principalmente de traducir los ficheros escritos a lenguajes de programación (comprensibles para los humanos) en código máquina (unos y ceros, comprensibles generalmente sólo por los ordenadores).

computador *m* Artilugio capaz de hacer cálculos y mostrar resultados sobre los datos numéricos que se introducen. Su complejidad ha ido aumentado y su finalidad ha ido variando, hasta convertirse en aparatos como las calculadoras de bolsillo o los ordenadores personales actuales.

C/C++ *m* Lenguajes de programación. C fue creado a principios de los años setenta para el desarrollo de Unix; se trata de uno de los primeros lenguajes de programación que permiten la programación de alto nivel. C++ es una versión posterior de C que añade técnicas modernas de programación. La mayoría del código libre está escrito en C o C++.

depuración *f* Proceso mediante el que se buscan (y corrigen) errores en el software. Véase también *depurador*.

depurador *m* Programa de ordenador utilizado a la hora de crear software para encontrar y corregir errores de programación.

drivers *m* Software que se encarga de interactuar entre el sistema operativo y los dispositivos (hardware).

hardware *m* Conjunto de dispositivos físicos que componen el ordenador: la pantalla, el teclado, el ratón, etc.

i386 Arquitectura de ordenador típica de los ordenadores personales (PC).

Java *m* Lenguajes de programación. Lenguaje de programación creado en los años noventa con muchas similitudes sintácticas con C y C++, pero con características de programación de más alto nivel de abstracción.

kernel *m* Núcleo del sistema operativo. Es el que se encarga de las labores de más bajo nivel (el nivel más cercano al hardware), como la gestión de la memoria, la entrada/salida de dispositivos, etc. El kernel más popular en el mundo del software libre es Linux, aunque existen muchos más (por ejemplo, los sistemas BSD tienen uno propio).

lenguaje ensamblador *m* Lenguaje de bajo nivel de abstracción que permite crear programas de ordenador. Se trata del lenguaje de programación más cercano al código máquina.

lenguaje de programación *m* Conjunto de reglas semánticas y sintácticas utilizadas para dar instrucciones a un ordenador. Los lenguajes de programación permiten trabajar a un nivel de abstracción superior que con código máquina, lo que facilita la creación y mantenimiento de programas informáticos. Existen miles de lenguajes de programación. Algunos ejemplos son C, C++, ADA, Java, Pascal y COBOL.

lenguaje natural *m* Lenguaje escrito o hablado por los humanos con fines de comunicación.

microprocesador *m* Elemento del hardware del sistema informático encargado de procesar la información, así como de gestionar la ejecución de los pedidos que conforman los programas y el traspaso de datos entre los diferentes componentes del computador.

modelo de desarrollo en cascada *m* Procedimiento para crear software, basado en dividir el proceso de desarrollo en una secuencia de tareas que se realizan una detrás de otra. Las tareas se llevan a cabo siguiendo el orden siguiente: definición de los requisitos, análisis y diseño de software, implementación y prueba de unidades, integración y pruebas del sistema y operación y mantenimiento.

multicomputador/multiprocesador *m* Nombre genérico que recibe un sistema informático basado en el uso de más de un computador o procesador.

Pascal *m* Lenguaje de programación de la década de los setenta escasamente utilizado hoy en día, aunque algunos de sus sucesores sí que cuentan con amplio eco en la industria del software.

sistema operativo *m* Programa responsable de la gestión y coordinación de actividades y del reparto de recursos de un ordenador.

software *m* Componente intangible en la informática. Generalmente, se trata de una serie de instrucciones elaboradas por humanos en lenguajes de programación de alto nivel (código fuente) que luego son traducidas por un compilador a código máquina (unos y ceros comprendidos por las máquinas). El software se divide en software de sistema, parte que corresponde a los sistemas operativos, o de aplicación, que agrupa los programas que el usuario suele utilizar. Estrictamente, el software también incluye la documentación del programa, aunque ésta se encuentre en un manual.

tabla de verdad *f* Una tabla de verdad especifica, para las diferentes combinaciones de los bits de entrada, cuál es la salida de una función lógica o de un circuito de cálculo.

tiempo de ejecución *m* Se trata del espacio temporal en el que el ordenador está ejecutando las instrucciones correspondientes a un programa de software.

Bibliografía

Angulo, J. M. et al. (2003). *Fundamentos y estructura de computadores*. Thomson Paraninfo.

Barceló, M. (2008). *Una historia de la Informática*. Barcelona: Editorial UOC.

Beekman, G. (2004). *Computer Confluence. Standard*. (6.ª ed.) Addison-Wesley.

Beekman, G. (2004). *Introducción a la informática*. Pearson Prentice Hall.

Leiva Olivencia, J. L. (2009). *Informática para Estudiantes*. Editorial Abecedario. ISBN 9788492669011.

Pareja, C.; Andreyo, Á.; Ojeda, M. (1993). *Introducción a la Informática*. ISBN: 84-7491-489-2.

Reina, Pedro. *Curso de Informática*.

Reynolds, G. W. y otros (2000). *Principios de sistemas de información*. (4.ª edición) Thomson Paraninfo.

Recurso: Módulo 1. Aspectos tecnológicos de los sistemas informáticos. Descripción: Este es el primer módulo del recurso "Fundamentos tecnológicos de la sociedad de la información" Idioma: ES Categoría: Humanas y Sociales Fecha de alta: 2010-06-17 00:00:00.0