

Estructura y diseño de computadores

LA INTERFAZ SOFTWARE/HARDWARE

TRADUCCIÓN DE LA CUARTA EDICIÓN EN LENGUA INGLESA

Estructura y diseño de computadores

LA INTERFAZ SOFTWARE / HARDWARE

TRADUCCIÓN DE LA CUARTA EDICIÓN EN LENGUA INGLESA

David A. Patterson

University of California, Berkeley

John L. Hennessy

Stanford University

Con contribuciones de

Perry Alexander
The University of Kansas

Peter J. Ashenden
Ashenden Designs Pty Ltd

Javier Bruguera
Universidade de Santiago de Compostela

Jichuan Chang
Hewlett-Packard

Matthew Farrens
University of California, Davis

David Kaeli
Northeastern University

Nicole Kaiyan
University of Adelaide

David Kirk
NVIDIA

James R. Larus
Microsoft Research

Jacob Leverich
Hewlett-Packard

Kevin Lim
Hewlett-Packard

John Nickolls
NVIDIA

John Oliver
Cal Poly, San Luis Obispo

Milos Prvulovic
Georgia Tech

Partha Ranganathan
Hewlett-Packard



EDITORIAL
REVERTÉ

Barcelona - Bogotá - Buenos Aires - Caracas - México

Registro bibliográfico (ISBD)

PATTERSON, DAVID A.

[Computer Organization and Design. Español]

Estructura y diseño de computadores / David A. Patterson, John L. Hennessy; versión española por: Dr. Javier Díaz Bruguera. –Barcelona : Reverté. D.L. 2011

XXV, 703 p., [184] p. : il. col. ; 24 cm

Ed. orig.: Computer organization and design: the hardware/software interface. 4.^a ed. Burlington: Elsevier Inc., cop. 2007. – Índice.

DL B-15281-2011. – ISBN 978-84-291-2620-4

1. Estructura y diseño de computadores. I. Hennessy, John L., coaut. II. Díaz Bruguera, Javier, trad. III. Título.

Título de la obra original:

Computer Organization and Design. The Hardware / Software Interface. Fourth Edition

Edición original en lengua inglesa publicada por:

ELSEVIER INC of 200 Wheeler Road, 6th floor, Burlington, MA 01803, USA

Copyright © 2009 by Elsevier Inc. *All Rights Reserved*

Edición en español:

© Editorial Reverté, S. A., 2011

ISBN: 978-84-291-2620-4

Versión española por:

Prof. Dr. Javier Díaz Bruguera

Catedrático de Universidad en el área de arquitectura y tecnología de computadores

Universidad de Santiago de Compostela

Maquetación: REVERTÉ-AGUILAR, SL

Propiedad de:

EDITORIAL REVERTÉ, S. A.

Loreto, 13-15, Local B

08029 Barcelona – España

Tel: (34) 93 419 33 36

Fax: (34) 93 419 51 89

reverte@reverte.com

www.reverte.com

Reservados todos los derechos. La reproducción total o parcial de esta obra, por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático, queda rigurosamente prohibida, salvo excepción prevista en la ley. Asimismo queda prohibida la distribución de ejemplares mediante alquiler o préstamo públicos, la comunicación pública y la transformación de cualquier parte de esta publicación (incluido el diseño de la cubierta) sin la previa autorización de los titulares de la propiedad intelectual y de la Editorial. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (arts. 270 y siguientes del Código Penal). El Centro Español de Derechos Reprográficos (CEDRO) vela por el respeto a los citados derechos.

Impreso en España - *Printed in Spain*

Depósito Legal: B-15281-2011

Impreso por Liberdúplex, S.L.U.

Sant Llorenç d'Hortons (Barcelona)


Contenido

Prefacio xiii

CAPÍTULOS

1

Abstracciones y tecnología de los computadores 2

- 1.1 Introducción 3
- 1.2 Bajo los programas 10
- 1.3 Bajo la cubierta 13
- 1.4 Prestaciones 26
- 1.5 El muro de la potencia 39
- 1.6 El gran cambio: el paso de monoprocesadores a multiprocesadores 41
- 1.7 Casos reales: fabricación y evaluación del AMD Opteron x4 44
- 1.8 Falacias y errores habituales 51
- 1.9 Conclusiones finales 54
-  1.10 Perspectiva histórica y lecturas recomendadas 55
- 1.11 Ejercicios 56

2

Instrucciones: el lenguaje del computador 74

- 2.1 Introducción 76
- 2.2 Operaciones del hardware del computador 77
- 2.3 Operandos del hardware del computador 80
- 2.4 Números con signo y sin signo 87
- 2.5 Representación de instrucciones en el computador 94
- 2.6 Operaciones lógicas 102
- 2.7 Instrucciones para la toma de decisiones 105
- 2.8 Apoyo a los procedimientos en el hardware del computador 112
- 2.9 Comunicarse con la gente 122
- 2.10 Direcciones y direccionamiento inmediato MIPS para 32 bits 128
- 2.11 Paralelismo e instrucciones: sincronización 137
- 2.12 Traducción e inicio de un programa 139
- 2.13 Un ejemplo de ordenamiento en C para verlo todo junto 149

Nota importante: En la presente edición en castellano, los contenidos del CD incluidos en la edición original son accesibles (en lengua inglesa) a través de la página web www.reverte.com/microsites/pattersonhennessy. Aunque en la presente edición no se proporciona un CD-ROM físico, a lo largo de todo el texto se menciona el CD y se utiliza el icono que lo representa para hacer referencia a su contenido.

1


*La civilización avanza
extendiendo el
número de operaciones
importantes que se
pueden hacer sin
pensar en ellas*

Alfred North Whitehead

An Introduction to Mathematics, 1911

Abstracciones y tecnología de los computadores

1.1	Introducción	3
1.2	Bajo los programas	10
1.3	Bajo la cubierta	13
1.4	Prestaciones	26
1.5	El muro de la potencia	39
1.6	El gran cambio: el paso de monoprocesadores a multiprocesadores	41

1.7	Casos reales: fabricación y evaluación del AMD Opteron x4	44
1.8	Falacias y errores habituales	51
1.9	Conclusiones finales	54
	1.10 Perspectiva histórica y lecturas recomendadas	55
1.11	Ejercicios	56

Nota importante: En la presente edición en castellano, los contenidos del CD incluido en la edición original (en inglés) son accesibles a través de la página web www.reverte.com/microsites/pattersonhennessy. Aunque en la presente edición no se proporciona un CD-ROM físico, a lo largo de todo el texto se menciona el CD y se utiliza el icono que lo representa para hacer referencia a su contenido.

1.1

Introducción

¡Bienvenido a este libro! Estamos encantados de tener esta oportunidad de transmitir el entusiasmo del mundo de los computadores. Éste no es un campo árido y aburrido, donde el progreso es glacial y donde las nuevas ideas se atrofian por negligencia. ¡No! Los computadores son el producto de la increíblemente vibrante industria de las tecnologías de la información, que en su conjunto es responsable como mínimo del 10% del producto nacional bruto de los Estados Unidos y cuya economía se ha vuelto, en parte, dependiente de las rápidas mejoras en tecnologías de la información prometidas por la ley de Moore. Esta insólita industria impulsa la innovación a una velocidad asombrosa. En los últimos 25 años aparecieron varios computadores nuevos cuya introducción parecía que iba a revolucionar la industria de la computación; estas revoluciones duraban poco tiempo simplemente porque alguien construía un computador aun mejor.

Esta carrera por innovar condujo a un progreso sin precedentes desde el inicio de la computación electrónica en los últimos años de la década de 1940. Si los medios de transporte hubiesen ido a la par con la industria de la computación, por ejemplo, hoy en día se podría viajar de Nueva York a Londres en aproximadamente un segundo por unos pocos céntimos. Pensemos un momento cómo tal adelanto habría cambiado la sociedad (vivir en Tahití y trabajar en San Francisco, ir a Moscú para ver el Ballet Bolshoi) y podremos apreciar las implicaciones de tal cambio.

Los computadores nos han llevado a una tercera revolución de la civilización, la revolución de la información, que se sitúa a la par de las revoluciones agrícola e industrial. El resultado de la multiplicación de la potencia y el alcance intelectual de la humanidad ha afectado profundamente a nuestras vidas cotidianas y también ha cambiado la manera de obtener nuevos conocimientos. Hay ahora una nueva forma de investigación científica, en la cual científicos informáticos trabajan junto a científicos teóricos y experimentales en la exploración de nuevas fronteras en astronomía, biología, química y física entre otras.

La revolución de los computadores continúa. Cada vez que el coste de la computación se mejora en un factor 10, las oportunidades para los computadores se multiplican. Aplicaciones que eran económicamente inviables repentinamente se convierten en factibles. Hasta hace muy poco, las siguientes aplicaciones eran “ciencia ficción computacional”.

- *Computadores en los coches*: hasta que los microprocesadores mejoraron drásticamente en precio y prestaciones a principios de la década de 1980, el control por computador en los coches era risible. Hoy en día los computadores reducen la contaminación y mejoran las prestaciones del combustible vía controles en el motor, y también mejoran la seguridad al prevenir peligrosos patinazos e inflando los *air bags* para proteger a los ocupantes en caso de colisión.
- *Teléfonos portátiles*: ¿quién habría podido soñar que los avances en sistemas de computación llevarían al desarrollo de teléfonos móviles, permitiendo una comunicación persona-a-persona casi en cualquier parte del mundo?
- *Proyecto genoma humano*: el coste del equipamiento informático para cartografiar y analizar las secuencias del ADN humano es de centenares de millones de dólares. Es improbable que alguien hubiese considerado este proyecto si los costes de los computadores hubiesen sido entre 10 y 100 veces mayores, tal y como eran hace 10 o 20 años. Además, el coste continúa bajando; podríamos ser capaces de adquirir nuestro propio genoma, permitiendo que los cuidados médicos se adapten a nosotros.
- *World Wide Web (La telaraña mundial)*: la World Wide Web, que no existía en la primera edición de este libro, ha cambiado nuestra sociedad. Para muchos, la *www* ha reemplazado a las bibliotecas.
- *Motores de búsqueda*: Dado que el contenido de la WWW ha crecido en tamaño y valor, encontrar información relevante es cada vez más importante. En la actualidad, mucha gente confía tanto en los motores de búsqueda que se verían en apuros si no pudiesen utilizarlos.

Claramente, los avances en esta tecnología influyen hoy en día en casi todos los aspectos de nuestra sociedad. Los avances en el hardware han permitido a los programadores crear programas maravillosamente útiles, y explican por qué los computadores son omnipresentes. Lo que hoy es ciencia ficción, serán las aplicaciones normales en el futuro: ya hay mundos virtuales, reconocimiento de voz y asistencia médica personalizada.

Tipos de aplicaciones de computador y sus características

Aunque se usan un conjunto común de tecnologías hardware (presentadas en las secciones 1.3 y 1.4) que van desde los electrodomésticos caseros inteligentes a los teléfonos móviles o celulares o los mayores supercomputadores, estas aplicaciones diferentes tienen diversos requerimientos de diseño y utilizan las tecnologías hardware de manera diferente. Grosso modo, los computadores se utilizan en tres clases diferentes de aplicaciones.

Los **computadores de sobremesa** son posiblemente la forma más conocida de computación y están representados por el computador personal, que muchos lectores de este libro habrán usado extensamente. Los computadores de sobremesa se caracterizan por dar buenas prestaciones a bajo coste a único usuario, y a menudo se usan para ejecutar programas de terceros, también llamado software estándar. La evolución de muchas tecnologías es impulsada por este tipo de computadores, ¡que sólo tiene 30 años de antigüedad!

Los **servidores** son la versión moderna de lo que fueron los computadores centrales, los minicomputadores y los supercomputadores, y generalmente solo se accede a ellos vía una red. Los servidores están pensados para soportar grandes cargas de trabajo, que pueden consistir en una única aplicación compleja, generalmente científica o de ingeniería, o en muchos trabajos pequeños, como ocurre en un servidor Web. Estas aplicaciones están basadas en programas de otras fuentes (tales como una base de datos o un sistema de simulación), pero frecuentemente se modifican o adaptan a una función concreta. Los servidores se construyen con la misma tecnología básica que los computadores de sobremesa, pero permiten una mayor ampliación de su capacidad tanto de computación como de entrada/salida. En general, los servidores también ponen gran énfasis en la confiabilidad, puesto que un fallo es generalmente más costoso que en un computador de sobremesa de un único usuario.

Los servidores abarcan la mayor gama de costes y capacidades. En el extremo más bajo, un servidor puede ser un poco más que una máquina de sobremesa sin pantalla ni teclado y con un coste un poco mayor. Estos servidores de gama baja se usan típicamente para almacenar archivos o ejecutar pequeñas aplicaciones de empresas o un servicio web sencillo (véase la sección 6.10). En el otro extremo están los **supercomputadores**, que en la actualidad disponen de cientos de miles de procesadores, y generalmente **terabytes** de memoria y **petabytes** de almacenamiento, y cuestan de millones a cientos de millones de dólares. Los supercomputadores se utilizan para cálculos científicos y de ingeniería de alta calidad, tales como predicción del clima, prospección petrolífera, determinación de la estructura de proteínas, y otros problemas de gran envergadura. Aunque estos supercomputadores representan el pico de la capacidad de computación, en términos relativos constituyen una pequeña fracción del número total de servidores y también del total del mercado de los computadores en términos de facturación.

Aunque no se les llama supercomputadores, los **centros de datos** de internet utilizados por compañías como eBay y Google disponen también de miles de procesadores, terabytes de memoria y petabytes de almacenamiento. Habitualmente se consideran como grandes clústeres de computadores (véase capítulo 7).

Computador de sobremesa: computador diseñado para un único usuario, y que incorpora una pantalla, un teclado y un ratón.

Servidor: computador que se utiliza para ejecutar grandes programas para muchos usuarios, a menudo simultáneamente; generalmente sólo se accede a él vía una red.

Supercomputador: computador con la capacidad de computación y coste más altos; se configuran como servidores y generalmente su coste es de millones de dólares.

Terabyte: originalmente son 1 099 511 627 776 (2^{40}) bytes, aunque algunos sistemas de comunicaciones y de almacenamiento secundario lo han redefinido como 1 000 000 000 000 (10^{12}) bytes.

Petabyte: 1000 o 1024 terabytes.

Centro de datos: una habitación o edificio diseñado con todo lo que se necesita para un número elevado de servidores: alimentación y potencia eléctrica, aire acondicionado y red.

Computadores empotrados: computador que se encuentra dentro de otro dispositivo y que se utiliza para ejecutar una aplicación predeterminada o un conjunto de aplicaciones relacionadas.

Los **computadores empotrados** constituyen la clase de computadores más amplia y son los que tienen la gama más amplia de aplicaciones y prestaciones. Los computadores empotrados incluyen los microprocesadores que se encuentran en los coches, los computadores de los teléfonos móviles o de los computadores de los videojuegos o de los televisores digitales, y las redes de procesadores que controlan los aviones modernos o los barcos de carga. Los sistemas de computación empotrada se diseñan para ejecutar una aplicación o un conjunto de aplicaciones relacionadas, que normalmente están integradas con el hardware y se proporcionan como un único sistema; así, a pesar del gran número de computadores empotrados, ¡muchos usuarios nunca ven realmente que están usando un computador!

La figura 1.1. muestra que durante los últimos años el incremento de los teléfonos móviles, que dependen de los computadores empotrados, ha sido mucho más rápido que el incremento de los computadores de sobremesa. Obsérvese que las televisiones digitales, coches, cámaras digitales, reproductores de música, videojuegos y otros muchos dispositivos de consumo incorporan también computadores empotrados, lo que incrementa aún más la diferencia entre el número de computadores empotrados y computadores de sobremesa.

Las aplicaciones empotradas a menudo tienen un único requisito de aplicación que combina unas prestaciones mínimas con fuertes limitaciones en coste o consumo de potencia. Por ejemplo, pensemos en un reproductor de música: El procesador necesita solamente tener la velocidad suficiente para llevar a cabo esta función limitada, y a partir de aquí, los objetivos más importantes son reducir el coste y el consumo de potencia. A pesar de su coste reducido, los computadores

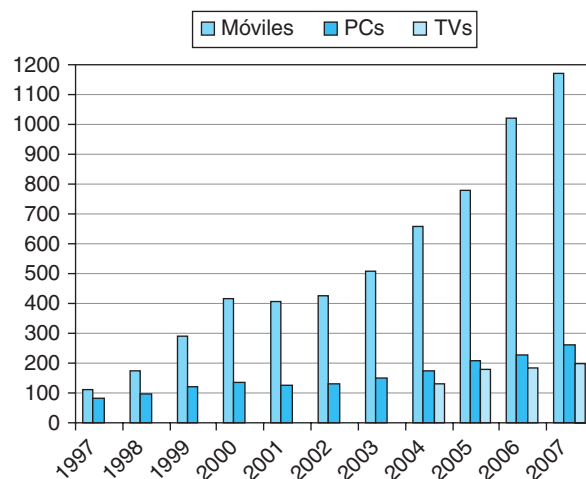


FIGURA 1.1 El número de teléfonos móviles, computadores personales y televisores fabricados cada año entre 1997 y 2007. (Solo hay datos de televisores del año 2004). Más de mil millones de nuevos teléfonos móviles se vendieron en 2006. En 1997, la venta de teléfonos móviles superaba a la de PC en un factor 1.4; y este factor creció hasta 4.5 en 2007. En 2004, se estimaba que había 2000 millones de televisores en uso, 1800 millones de teléfonos móviles y 800 millones de PCs. Cómo la población mundial era 6400 millones de personas, había aproximadamente 1 PC, 2.2 teléfonos móviles y 2.5 televisores por cada 8 habitantes del planeta. En 2006, un estudio estimó que en Estados Unidos había una media de 12 dispositivos por familia, incluyendo 3 televisiones, 2 PCs y otros aparatos como consolas para videojuegos, reproductores de MP3 y teléfonos móviles.

empotrados tienen a menudo menor tolerancia a fallos, porque las consecuencias de un fallo pueden variar desde ser molestas (cuando nuestra televisión nueva se estropea) a devastadoras (algo que puede ocurrir cuando falla el computador de un avión o de un barco de carga). En aplicaciones empotradas orientadas al consumo, como por ejemplo electrodomésticos digitales, la funcionalidad se obtiene principalmente a través de la sencillez; se intenta hacer una función tan perfectamente como sea posible. En grandes sistemas empotrados, se emplean a menudo técnicas de redundancia desarrolladas para el mundo de los servidores (véase sección 6.9). Aunque este libro se centra en los computadores de propósito general, la mayoría de los conceptos son aplicables directamente, o con pocas modificaciones, a los computadores empotrados.

Extensión: Las extensiones son secciones cortas que se usan a lo largo del texto para proporcionar un mayor detalle sobre algún tema en particular, que puede ser de interés. Los lectores que no están interesados en una extensión, pueden saltársela, ya que el material siguiente nunca dependerá de su contenido.

Muchos procesadores empotrados se diseñan usando *núcleos de procesador*, una versión de un procesador escrito en un lenguaje de descripción hardware, tal como Verilog o VHDL (véase capítulo 4). El núcleo permite a un diseñador integrar otro hardware específico de la aplicación con el núcleo del procesador para fabricar un único circuito integrado.

Qué puede aprender

Los programadores de éxito siempre han estado comprometidos con las prestaciones de sus programas, porque proporcionar de forma rápida resultados al usuario es fundamental para crear software de éxito. En las décadas de 1960 y 1970, la principal restricción para las prestaciones era el tamaño de la memoria del computador. Así, los programadores frecuentemente seguían una sencilla regla: minimizar el espacio de la memoria para hacer programas rápidos. En la última década, los avances en el diseño de los computadores y en la tecnología de la memoria redujeron drásticamente la importancia del pequeño tamaño de la memoria en muchas aplicaciones, excepto las de los sistemas de computación empuotrada.

Los programadores interesados en las prestaciones necesitan ahora conocer las cuestiones que han reemplazado el modelo de memoria simple de la década de 1960: la naturaleza jerárquica de las memorias y la naturaleza paralela de los procesadores. Los programadores que buscan construir versiones competitivas de los compiladores, los sistemas operativos, las bases de datos, e incluso de aplicaciones tendrán consecuentemente que incrementar su conocimiento de la organización de los computadores.

Nosotros tenemos la oportunidad de explicar qué hay dentro de esta máquina revolucionaria, desentrañando el software bajo su programa y el hardware bajo las cubiertas de su computador. Cuando complete este libro, creemos que será capaz de contestar a las siguientes preguntas:

- ¿Cómo se escriben los programas en un lenguaje de alto nivel, tales como C o Java, cómo se traducen al lenguaje del hardware, y cómo ejecuta el hardware

el programa resultante? Comprender estos conceptos forma la base para entender los aspectos tanto del hardware y como del software que afectan a las prestaciones de los programas.

- ¿Cuál es la interfaz entre el software y el hardware, y cómo el software instruye al hardware para realizar las funciones necesarias? Estos conceptos son vitales para comprender cómo se escriben muchos tipos de software.
- ¿Qué determina las prestaciones de un programa, y cómo un programador puede mejorarlo? Como veremos, esto depende del programa original, el software de traducción de este programa al lenguaje del computador, y de la efectividad del hardware al ejecutar el programa.
- ¿Qué técnicas pueden usar los diseñadores de hardware para mejorar las prestaciones? Este libro introducirá los conceptos básicos del diseño de los computadores actuales. El lector interesado encontrará mucho más material sobre este tema en nuestro libro avanzado, *Arquitectura del Computador: Una aproximación cuantitativa*.
- ¿Cuáles son las razones y consecuencias del reciente paso del procesamiento secuencial al paralelo? En este libro se indican las motivaciones, se describe el hardware disponible actualmente para dar soporte al paralelismo y se revisa la nueva generación de **microprocesadores multinúcleo** (véase capítulo 7).

Microprocesador multinúcleo: procesador que contiene varios procesadores o núcleos en un único circuito integrado.

Sin comprender las respuestas a estas preguntas, mejorar las prestaciones de sus programas en un computador moderno, o evaluar qué características podrían hacer un computador mejor que otro para una aplicación particular, sería un complejo proceso de prueba y error, en lugar de un procedimiento científico conducido por la comprensión y el análisis.

El primer capítulo contiene los fundamentos para el resto del libro. Introduce las ideas básicas y las definiciones, pone en perspectiva los mayores componentes del hardware y del software, muestra como evaluar prestaciones y potencia e introduce los circuitos integrados, la tecnología que alimenta la revolución de los computadores y explica la evolución hacia los multinúcleos.

En este capítulo, y otros posteriores, probablemente usted verá un montón de nuevas palabras, o palabras que puede haber oído, pero que no está seguro de lo que significan. ¡No se preocupe! Sí, hay un montón de terminología especial que se usa en la descripción de los computadores modernos, pero la terminología realmente es una ayuda que nos permite describir con precisión una función o capacidad. Además, a los diseñadores de computadores (incluidos a los autores de este libro) les encanta usar **acrónimos**, ¡que son más fáciles de comprender cuando se sabe lo que significa cada una de sus letras! Para ayudarle a recordar y localizar los términos, hemos incluido una definición resaltada de cada uno de ellos la primera vez que aparece en el texto. Después de un breve periodo trabajando con la terminología, usted estará habituado, y sus amigos se quedarán impresionados de la manera tan correcta como usa palabras como BIOS, CPU, DIMM, DRAM, PCIE, SATA y tantas otras.

Para reforzar la comprensión de cómo los sistemas hardware y software que se usan para ejecutar un programa afectan a las prestaciones, a lo largo del libro usamos la sección especial “Comprender las prestaciones de los programas”. A

Acrónimo: palabra construida tomando las letras iniciales de cada una de las palabras de una frase; por ejemplo, RAM es un acrónimo de *Random Access Memory* (memoria de acceso aleatorio), y CPU es un acrónimo de *Central Processing Unit* (unidad central de proceso).

continuación veremos la primera. Estas secciones resumen detalles importantes de las prestaciones de los programas.

Las prestaciones de un programa dependen de una combinación de efectividad de los algoritmos usados en el programa, de los sistemas de software usados para crear y traducir el programa en instrucciones máquina y de la efectividad del computador al ejecutar esas instrucciones, las cuales pueden incluir operaciones de entrada y salida (E/S). La siguiente tabla resume cómo afectan a las prestaciones tanto el hardware como el software.

Comprender las prestaciones de los programas

Componente Hardware o software	Cómo afecta este componente a las prestaciones	Dónde se cubre este tema
Algoritmo	Determina el número de sentencias de alto nivel y el número de operaciones de E/S que se ejecutarán	¡Otros libros!
Lenguaje de programación, compilador y arquitectura	Determina el número de instrucciones máquina que se ejecutarán por cada sentencia de alto nivel	capítulos 2 y 3
Procesador y sistema de memoria	Determina cuán rápido se pueden ejecutar las instrucciones	capítulos 4, 5 y 7
Sistema de E/S (hardware y sistema operativo)	Determina cuán rápido se pueden ejecutar las operaciones de E/S	capítulo 6

En París sólo me miraban fijamente cuando les hablaba en francés; nunca conseguí hacer entender a esos idiotas su propio idioma.

Mark Twain, *The Innocents Abroad*, 1869

Software de sistemas:

software que proporciona servicios que habitualmente son útiles, entre ellos los sistemas operativos, los compiladores y los ensambladores.

Sistema operativo:

programa de supervisión que gestiona los recursos de un computador para provecho de los programas que se ejecutan en esa máquina.

1.2

Bajo los programas

Una aplicación típica, tal como un procesador de textos o un gran sistema de base de datos, puede consistir de cientos de miles o millones de líneas de código y depender de sofisticadas bibliotecas de software que implementan funciones complejas de soporte a la aplicación. Tal y como veremos, el hardware de un computador sólo puede ejecutar instrucciones extremadamente simples de bajo nivel. Para ir de una aplicación compleja hasta las instrucciones simples se ven involucradas varias capas de software que interpretan o trasladan las operaciones de alto nivel en instrucciones simples del computador.

Estas capas de software están organizadas principalmente en forma jerárquica, donde las aplicaciones son el anillo más externo y una variedad de **software de sistemas** se coloca entre el hardware y las aplicaciones software, como muestra la figura 1.2.

Hay muchos tipos de software de sistemas, pero actualmente hay dos tipos que son fundamentales para todos los computadores: un sistema operativo y un compilador. Un **sistema operativo** interactúa entre el programa del usuario y el hardware y proporciona una variedad de servicios y funciones de supervisión. Entre sus funciones más importantes están:

- manejo de las operaciones básicas de entrada y salida
- asignación de espacio de almacenamiento y de memoria
- facilitar la compartición del computador entre múltiples aplicaciones simultáneas

Ejemplos de sistemas operativos en uso hoy en día son Windows, Linux y MacOS.



FIGURA 1.2 Vista simplificada del hardware y el software como capas jerárquicas, mostradas como círculos concéntricos con el hardware en el centro y el software de las aplicaciones en el exterior. En aplicaciones complejas frecuentemente se encuentran múltiples capas software. Por ejemplo, un sistema de base de datos puede ejecutarse sobre el software de sistemas que aloja una aplicación, el cual a su vez se ejecuta sobre la base de datos.

Los **compiladores** realizan otra función vital: la traducción de un programa escrito en un lenguaje de alto nivel, como C, C++, Java o Visual Basic a instrucciones que el hardware puede ejecutar. Dada la sofisticación de los modernos lenguajes de programación y las instrucciones simples ejecutadas por el hardware, la traducción desde un programa en un lenguaje de alto nivel a instrucciones hardware es compleja. Daremos una breve visión general del proceso y volveremos a este tema en el capítulo 2 y apéndice B.

Del lenguaje de alto nivel al lenguaje del hardware

Para hablar realmente a una máquina electrónica es necesario enviar señales eléctricas. Las señales eléctricas más fáciles de entender para las máquinas son encendido (*on*) y apagado (*off*), y por lo tanto el alfabeto de la máquina tiene sólo dos letras. Del mismo modo que las 26 letras del alfabeto inglés no limitan cuánto se puede escribir, las dos letras del alfabeto de los computadores no limitan lo que los éstos pueden hacer. Los dos símbolos para estas letras son los números 0 y 1 y habitualmente pensamos en el lenguaje de las máquinas como números en base 2, o números binarios. Nos referimos a cada “letra” como **dígito binario** o **bit**.¹ Los computadores son esclavos de nuestras órdenes. De ahí que el nombre para una orden individual sea **instrucción**. Las instrucciones, que son simples colecciones de bits que el computador puede comprender, se pueden pensar como números. Por ejemplo, los bits

1000110010100000

indican a un computador que sume dos números. En el capítulo 3 explicamos por qué usamos números para instrucciones y datos; no queremos adelantar acontecimientos, pero el uso de números tanto para las instrucciones como para los datos es de importancia capital para la informática.

Los primeros programadores se comunicaban con los computadores mediante números binarios, pero eso era tan laborioso que rápidamente inventaron nuevas notaciones más próximas a la forma de pensar de los humanos. Al principio estas notaciones se traducían a binario a mano, pero ese proceso aún era fatigoso. Usando la propia máquina para ayudar a programar la máquina, los pioneros inventaron programas para traducir de notación simbólica a binario. El primero de estos programas fue llamado **ensamblador**. Este programa traduce la versión simbólica de una instrucción a su versión binaria. Por ejemplo, el programador escribiría

add A, B

y el ensamblador traduciría esta notación a

1000110010100000

Esta instrucción indica al computador que sume los números A y B. El nombre acuñado para este lenguaje simbólico, aún usado hoy en día, es **lenguaje ensamblador**. Por el contrario, el lenguaje binario que entiende el computador se llama **lenguaje máquina**.

Compiladores:

programa que traduce sentencias en un lenguaje de alto nivel a sentencias en lenguaje ensamblador.

Dígito binario: también llamado bit. Uno de los dos números en base 2 (0 ó 1) que son los componentes de la información.

Instrucción: orden que el hardware del computador entiende y obedece.

Ensamblador: programa que traduce una versión simbólica de las instrucciones a su versión binaria.

Lenguaje ensamblador: representación simbólica de las instrucciones de la máquina.

Lenguaje máquina: representación binaria de las instrucciones máquina.

1. Originariamente, contracción inglesa de *binary digit*. (N. del T.)

Aunque sea una enorme mejora, el lenguaje ensamblador todavía queda muy lejos de la notación que le gustaría usar a un científico para simular el flujo de fluidos o la que podría usar un contable para hacer balance de sus cuentas. El lenguaje ensamblador requiere que el programador escriba una línea para cada instrucción que desee que la máquina ejecute; por tanto, este lenguaje fuerza al programador a pensar como la máquina.

El reconocimiento de que se podía escribir un programa para traducir un lenguaje más potente a instrucciones del computador fue uno de los grandes avances en los primeros días de la computación. Los programadores de hoy en día deben su productividad (y su cordura) a la creación de los **lenguajes de programación de alto nivel** y los compiladores que traducen los programas en tales lenguajes en instrucciones. La figura 1.3 muestra la relación entre estos programas y lenguajes.

Lenguaje de programación de alto nivel: lenguaje transportable tal como C, Fortran o Java compuesto por palabras y notación algebraica que un compilador puede traducir en lenguaje ensamblador.

Programa
en lenguaje
de alto nivel
(en C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

Compilador de C

Programa
en lenguaje
ensamblador
(para MIPS)

```
swap:
  muli $2, $5, 4
  add  $2, $4, $2
  lw   $15, 0($2)
  lw   $16, 4($2)
  sw   $16, 0($2)
  sw   $15, 4($2)
  jr   $31
```

Ensamblador

Programa
en lenguaje
máquina
binario
(para MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
0000001111100000000000000001000
```

FIGURA 1.3 Programa en C compilado a lenguaje ensamblador y luego ensamblado a lenguaje máquina binario. Aunque la traducción del lenguaje de alto nivel al lenguaje máquina binario se muestra en dos pasos, algunos compiladores eliminan el paso intermedio y producen lenguaje máquina binario directamente. En el capítulo 2 se examinan con más detalle estos lenguajes y este programa.

Un compilador permiten que un programador escriba esta expresión en lenguaje de alto nivel:

$$A + B$$

El compilador compilaría esto en esta sentencia de lenguaje ensamblador:

```
add A, B
```

El ensamblador traduciría esto en la instrucción binaria que indica al computador que sume los números A y B.

Los lenguajes de programación de alto nivel ofrecen varias ventajas importantes. Primero, permiten al programador pensar en un lenguaje más natural, usando palabras inglesas y notación algebraica, dando lugar a programas con un aspecto mucho más parecido a un texto que a una tabla de símbolos crípticos (véase figura 1.3). Por otro lado, permiten que los lenguajes sean diseñados de acuerdo con su intención de uso. De este modo, el Fortran fue diseñado para computación científica, Cobol para el procesamiento de datos comerciales, Lisp para manipulación simbólica, etc. Hay también lenguajes de dominio específico, incluso para pequeños grupos de usuarios, como por ejemplo los interesados en simulación de fluidos.

La segunda ventaja de los lenguajes de programación es la mejora de la productividad del programador. Una de las pocas cuestiones con amplio consenso en el desarrollo de programas es que toma menos tiempo desarrollar programas cuando se escriben en lenguajes que requieren menos líneas para expresar una idea. La concisión es una ventaja clara de los lenguajes de alto nivel sobre el lenguaje ensamblador.

La ventaja final es que los lenguajes de programación permiten a los programas ser independientes del computador sobre el que se desarrollan, ya que los compiladores y ensambladores pueden traducir programas en lenguaje de alto nivel a las instrucciones binarias de cualquier máquina. Estas tres ventajas son tan decisivas que hoy en día se programa muy poco en lenguaje ensamblador.

1.3 Bajo la cubierta

Ahora que hemos mirado debajo de los programas para descubrir la programación subyacente, abramos la cubierta del computador para aprender sobre la circuitería que hay debajo. El hardware de cualquier computador lleva a cabo las mismas funciones básicas: introducción de datos, extracción de resultados, procesamiento de datos y almacenamiento de datos. El tema principal de este libro es explicar cómo se realizan estas funciones, y los capítulos siguientes tratan las diferentes partes de estas cuatro tareas.

Cuando llegamos a un punto importante en este libro, un punto tan importante que desearíamos que lo recordase para siempre, lo enfatizamos identificándolo con

un ítem de “Idea clave”. Tenemos cerca de una docena de Ideas clave en este libro, siendo la primera los cinco componentes de un computador que realizan las tareas de entrada, salida, proceso y almacenamiento de datos.

IDEA clave

Los cinco componentes clásicos de un computador son entrada, salida, memoria, camino de datos y control, donde las dos últimas a veces están combinadas y se llaman el procesador. La figura 1.4 muestra la organización estándar de un computador. Esta organización es independiente de la tecnología del hardware: se puede colocar cada parte de cada computador, pasado y presente, en una de estas cinco categorías. Para ayudarle a tener todo esto en perspectiva, los cinco componentes de un computador se muestran en la primera página de los capítulos siguientes, con la parte de interés para ese capítulo resaltada.

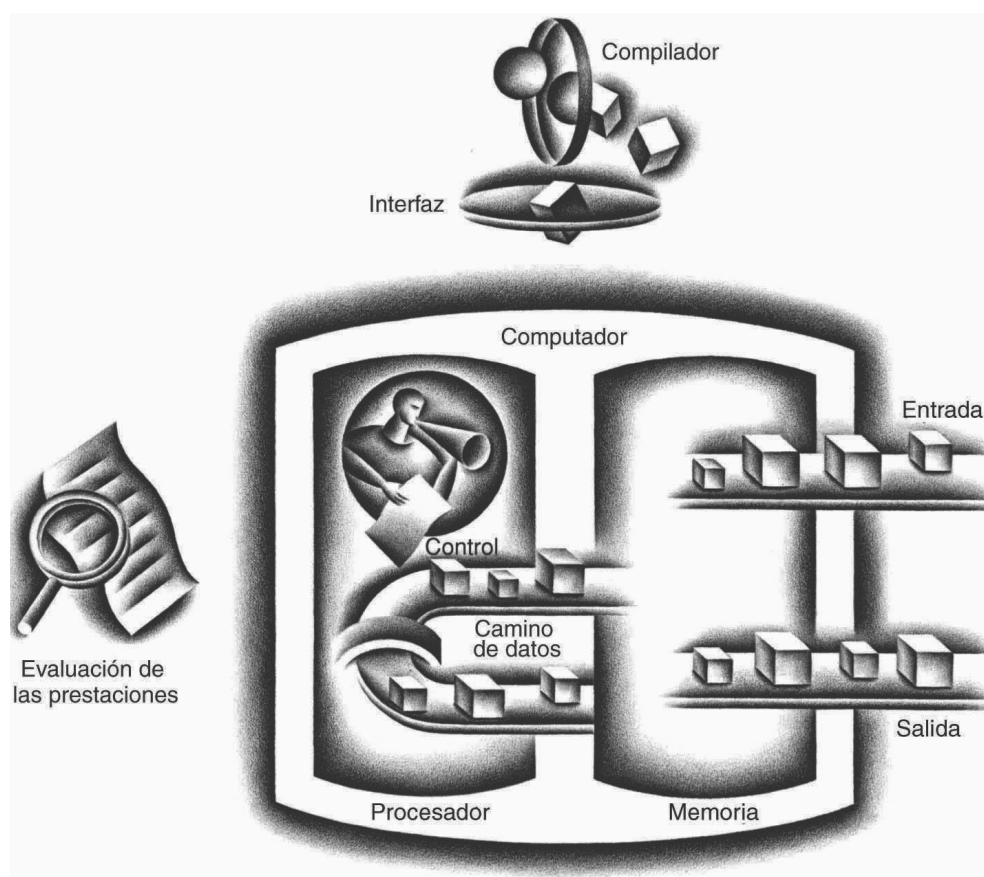


FIGURA 1.4 La organización de un computador, mostrando los cinco componentes clásicos. El procesador toma las instrucciones y datos de la memoria. La entrada escribe datos en la memoria y la salida lee datos de la memoria. El control envía señales que determinan las operaciones del camino de datos, la memoria, la entrada y la salida.

DRAM. La DRAM es la **memoria dinámica de acceso aleatorio** (*dynamic random access memory*). Se usan conjuntamente varias DRAMs para contener las instrucciones y los datos de los programas. En contraste con las memorias de acceso secuencial, como las cintas magnéticas, la parte RAM del término DRAM significa que los accesos a memoria toman el mismo tiempo independientemente de la posición de memoria que se lea.

Memoria de acceso aleatorio dinámica (DRAM): memoria construida como un circuito integrado, que provee acceso aleatorio a cualquier posición.

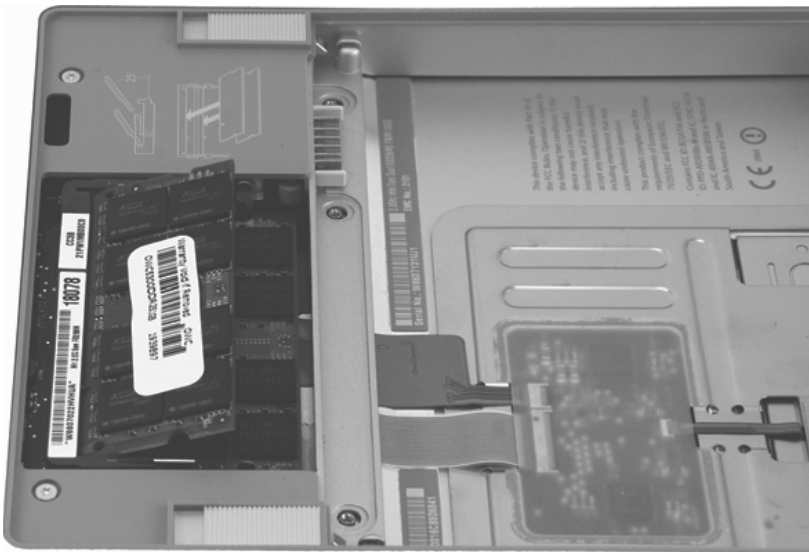


FIGURA 1.8 La memoria se aprecia con una vista cercana de la parte inferior del portátil.

La memoria principal se aloja en una o más pequeñas tarjetas mostradas a la izquierda. El hueco para la batería se puede apreciar a la derecha. Los chips de DRAM se colocan en esas tarjetas (llamadas **DIMM**, *dual inline memory modules* o módulos de memoria en línea duales) y se enchufan a los conectores. Foto por gentileza de OtherWorldComputing.com

El *procesador* es la parte activa de la placa y se encarga de seguir las instrucciones de un programa al pie de la letra. Suma números, comprueba números, indica la activación de dispositivos de E/S, etc. El procesador es el cuadrado grande que se encuentra debajo del ventilador y está cubierto por un disipador térmico, en la parte izquierda de la figura 1.7. A veces, al procesador se le llama CPU, porque el término suena más “oficial”: **unidad central de proceso** (*central processing unit*, CPU).

Adentrándonos aún más en el hardware, la figura 1.9 revela detalles del procesador. El procesador comprende dos componentes principales: el camino de datos (*datapath*) y el control, la fuerza y el cerebro del procesador, respectivamente. El **camino de datos** realiza las operaciones aritméticas. El **control** indica al camino de datos, a la memoria y a los dispositivos de E/S lo que deben hacer, de acuerdo con la voluntad de las instrucciones del programa. El capítulo 4 explica el camino de datos y el control para obtener un diseño con mejores prestaciones.

Introducirse en las profundidades de cualquier componente del hardware supone comprender lo que hay en la máquina. Dentro del procesador hay otro tipo de memoria: la memoria *cache*. La **memoria cache** es una memoria pequeña

DIMM (módulo de memoria de dos líneas):

pequeña tarjeta que contiene chips DRAM en ambas caras. Los SIMMs tienen DRAMs en una sola cara.

Unidad central de proceso (CPU): también llamada procesador. Es la parte activa de un computador; contiene los caminos de datos y el control que suma y comprueba números, indica a los dispositivos de E/S que se activen, etc.

Camino de datos: componente del procesador que ejecuta las operaciones aritméticas.

Control: componente del procesador que gobierna el camino de datos, la memoria y los dispositivos de E/S según a las instrucciones del programa.

Memoria cache: memoria rápida y pequeña que actúa como un *búfer* para otra memoria mayor y más lenta.

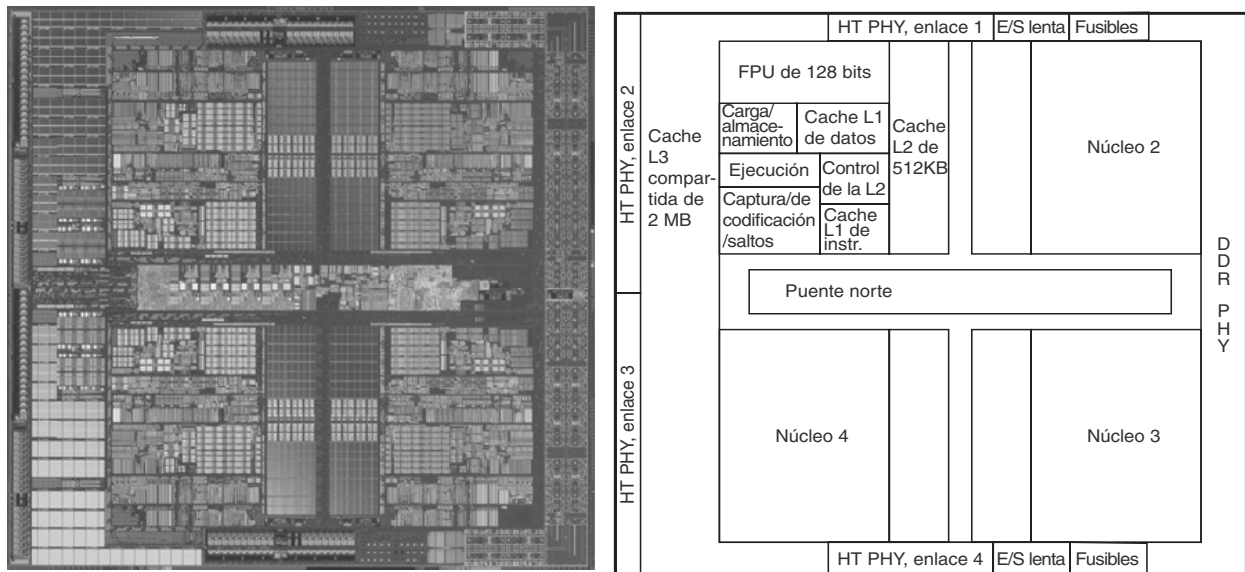


FIGURA 1.9 Interior del chip del microprocesador AMD Barcelona. La parte izquierda es una microfotografía de un chip procesador AMD Barcelona y la parte derecha muestra los principales bloques del procesador. Este chip tiene 4 procesadores o núcleos. El microprocesador del portátil de la figura 1.7 tiene dos núcleos, Intel Core 2 Duo.

Memoria de acceso aleatorio estática (SRAM):

también es una memoria construida como un circuito integrado, pero es más rápida y menos densa que la DRAM.

Abstracción: modelo que oculta temporalmente los detalles de menor nivel de los sistemas de computadores para facilitar el diseño de sistemas sofisticados.

Arquitectura del repertorio de instrucciones (también llamada arquitectura):

en una máquina, es una interfaz abstracta entre el hardware y el nivel más bajo del software que incluye toda la información necesaria para escribir un programa en lenguaje máquina que se ejecutará correctamente, e incluye

continúa...

y rápida que actúa como un búfer para la memoria DRAM. (Esta definición no técnica de *cache* es una buena forma de ocultar detalles que se verán posteriormente.) La memoria *cache* se construye usando una tecnología de memoria diferente, memoria de acceso aleatorio estático (*static random access memory, SRAM*). La SRAM es más rápida pero menos densa, y por lo tanto, más cara que la DRAM (véase capítulo 5).

Se observa fácilmente un aspecto común en las descripciones tanto del software como del hardware: penetrar en las profundidades revela más información, o a la inversa, los detalles de bajo nivel se ocultan para ofrecer un modelo más simple a los niveles más altos. El uso de estos niveles o **abstracciones** es una técnica fundamental para diseñar computadores muy complejos.

Una de las abstracciones más importantes es la interfaz entre el hardware y el nivel más bajo del software. Por su importancia recibe un nombre especial: **arquitectura del repertorio de instrucciones** (*instruction set architecture*), o simplemente arquitectura, de una máquina. La arquitectura del repertorio de instrucciones incluye todo lo que los programadores puedan necesitar para construir un programa correcto en lenguaje máquina binario, incluidas las instrucciones, los dispositivos de E/S, etc. Típicamente, el sistema operativo se encarga de realizar los detalles de las operaciones de E/S, de asignación de memoria y de otras funciones de bajo nivel del sistema, de manera que los programadores de aplicaciones no necesitan preocuparse de

esos detalles. La combinación del repertorio de instrucciones básico y la interfaz del sistema operativo proporcionados para los programadores de aplicaciones se llama la **interfaz binaria de las aplicaciones** (*application binary interface, ABI*).

Una *arquitectura del repertorio de instrucciones* permite a los diseñadores de computadores hablar de las funciones independientemente del hardware que las lleva a cabo. Por ejemplo, se puede hablar de las funciones de un reloj digital (mantener y mostrar la hora, establecer alarmas) independientemente de la circuitería del reloj (cristal de cuarzo, pantalla *LED*, botones de plástico). Los diseñadores de computadores hacen una distinción entre arquitectura e **implementación** de una arquitectura: una implementación es el hardware que obedece a una abstracción arquitectónica. Estas ideas nos llevan a otra Idea clave.

...continuación

las instrucciones, los registros, el acceso a memoria, la E/S, etc.

Interfaz binaria de aplicación (ABI): es la porción del repertorio de instrucciones correspondiente al usuario más la interfaz del sistema operativo que usan los programadores de aplicaciones. Define un estándar para la portabilidad binaria entre computadores.

Implementación:

Hardware que cumple la abstracción de una arquitectura.

Tanto el hardware como el software están estructurados en niveles jerárquicos, cada uno de los cuales oculta detalles al nivel superior. Mediante este principio de *abstracción* los diseñadores de hardware y de software pueden enfrentarse a la complejidad de los computadores. Una interfaz clave entre los niveles de abstracción es la *arquitectura del repertorio de instrucciones*: la interfaz entre el hardware y el software de bajo nivel. Esta interfaz abstracta permite muchas *implementaciones* de costes y prestaciones diferentes para ejecutar programas idénticos.

Transistor: interruptor de encendido/apagado controlado por una señal eléctrica.

Circuito integrado a muy gran escala (VLSI): dispositivo que contiene cientos de miles a millones de transistores.

Un **transistor** es simplemente un interruptor de encendido/apagado, controlado eléctricamente. El *circuito integrado* combina desde docenas hasta cientos de ellos en un solo chip. Para describir el enorme incremento de integración desde cientos a millones de transistores, se utiliza el adjetivo *a muy gran escala*, que da lugar a la expresión **circuito a muy gran escala de integración**, o **circuito VLSI** (*very large scale of integration*).

Esta velocidad de aumento de la integración ha sido notablemente estable. La figura 1.12 muestra el incremento de la capacidad de las DRAM desde 1977. La industria, prácticamente, ha cuadruplicado la capacidad cada 3 años, ¡un incremento total que excede las 16 000 veces en unos 20 años! Este incremento en el número de transistores en un circuito integrado se conoce popularmente como la ley de Moore, que establece que la capacidad en transistores se dobla cada 18 ó 24 meses. La ley de Moore es el resultado de una predicción del crecimiento en la capacidad de los circuitos integrados hecha por Gordon Moore, uno de los fundadores de Intel, durante los años sesenta del siglo pasado.

El mantenimiento de esta tasa de progreso durante al menos 40 años ha requerido increíbles innovaciones en las técnicas de manufacturación. En la sección 1.7 abordamos cómo se fabrican los circuitos integrados.

La ecuación clásica de las prestaciones de la CPU

Ahora se puede escribir la ecuación básica de las prestaciones en términos del **número de instrucciones** (número de instrucciones ejecutadas por el programa), del CPI y del tiempo de ciclo:

$$\text{Tiempo de ejecución} = \text{Número de instrucciones} \times \text{CPI} \times \text{Tiempo de ciclo}$$

o bien, dado que la frecuencia es el inverso del tiempo de ciclo:

$$\text{Tiempo de ejecución} = \frac{\text{Número de instrucciones} \times \text{CPI}}{\text{Frecuencia de reloj}}$$

Estas fórmulas son especialmente útiles porque distinguen los tres factores claves que influyen en las prestaciones. Estas fórmulas se pueden utilizar para comparar dos realizaciones diferentes o para evaluar un diseño alternativo si se conoce el impacto en estos tres parámetros.

Número de instrucciones: número de instrucciones ejecutadas por el programa.

Componentes de las prestaciones	Unidades de medida
Tiempo de ejecución de CPU de un programa	Segundos por programa
Número de instrucciones	Número de instrucciones ejecutadas por el programa
Ciclos por instrucción (CPI)	Número medio de ciclos por instrucción
Tiempo de ciclo del reloj	Segundos por ciclo de reloj

FIGURA 1.14 Componentes básicos de las prestaciones y cómo se mide cada uno de ellos.

Comprender las prestaciones de los programas

Las prestaciones de un programa depende del algoritmo, del lenguaje, del compilador, de la arquitectura y del hardware real. La siguiente tabla resume de qué manera estos componentes afectan a los distintos factores de la ecuación de las prestaciones.

Componente hardware o software	¿A qué afecta?	¿Cómo?
Algoritmo	Número de instrucciones, posiblemente CPI	El algoritmo determina el número de instrucciones del programa fuente ejecutadas y por lo tanto el número de instrucciones del procesador ejecutadas. El algoritmo puede también afectar al CPI, favoreciendo instrucciones más lentas o más rápidas. Por ejemplo, si el algoritmo utiliza más operaciones en punto flotante, tenderá a tener un mayor CPI.
Lenguaje de programación	Número de instrucciones, CPI	El lenguaje de programación afecta al número de instrucciones, ya que las sentencias del lenguaje son traducidas a instrucciones del procesador, lo cual determina el número de instrucciones. Las características del lenguaje también pueden afectar al CPI; por ejemplo, un lenguaje con soporte para datos abstractos (p. ej. Java) requerirá llamadas indirectas, las cuales utilizarán instrucciones con un CPI mayor.
Compilador	Número de instrucciones, CPI	La eficiencia del compilador afecta tanto al número de instrucciones como al promedio de los ciclos por instrucción, ya que el compilador determina la traducción de las instrucciones del lenguaje fuente a instrucciones del computador. El papel del compilador puede ser muy complejo y afecta al CPI de formas complejas.
Arquitectura del repertorio de instrucciones	Número de instrucciones, frecuencia de reloj, CPI	La arquitectura del repertorio de instrucciones afecta a los tres aspectos de las prestaciones de la CPU, ya que afecta a las instrucciones necesarias para realizar una función, al coste en ciclos de cada instrucción, y a la frecuencia del reloj del procesador.

Extensión: Aunque se podría esperar que el valor mínimo para el CPI es 1, como veremos en el capítulo 4, algunos procesadores buscan y ejecutan varias instrucciones en cada ciclo de reloj; para reflejar este hecho, algunos diseñadores invierten el CPI para obtener el IPC, *instrucciones por ciclo*. Si un procesador ejecuta una media de 2 instrucciones por ciclo, el IPC es 2 y, por lo tanto, el CPI es 0.5.

1.5 El muro de la potencia

La figura 1.15 ilustra el incremento en la frecuencia de reloj y el consumo de potencia de ocho generaciones de microprocesadores de Intel en los últimos 25 años. Tanto la frecuencia de reloj como el consumo de potencia crecieron de forma rápida durante décadas, y de forma más moderada recientemente. La razón de este crecimiento conjunto es que están correlacionados, y la razón de un crecimiento moderado más recientemente es que se han alcanzado los límites prácticos de disipación de potencia en los microprocesadores corrientes.

La tecnología dominante en la fabricación de circuitos integrados es la tecnología CMOS (*complementary metal oxide semiconductor*). En CMOS, la fuente principal de disipación de potencia es la llamada potencia dinámica; es decir, potencia consumida en las transiciones. La disipación de potencia dinámica depende de la carga capacitiva de cada transistor, del voltaje aplicado y de la frecuencia de conmutación del transistor:

$$\text{Potencia} = \text{carga capacitiva} \times \text{voltaje}^2 \times \text{frecuencia de conmutación}$$

La frecuencia de conmutación depende de la frecuencia de la señal de reloj. La carga capacitiva por transistor es una función del número de transistores conectados a una salida (llamado *fanout*) y de la tecnología, que determina la capacitancia de las conexiones y los transistores.

¿Cómo es posible que la frecuencia de la señal de reloj crezca un factor 1000 mientras la potencia crece sólo un factor 30? La potencia puede reducirse disminuyendo el voltaje, algo que ha ocurrido con cada nueva generación de la tecnología, ya que depende de forma cuadrática del voltaje.

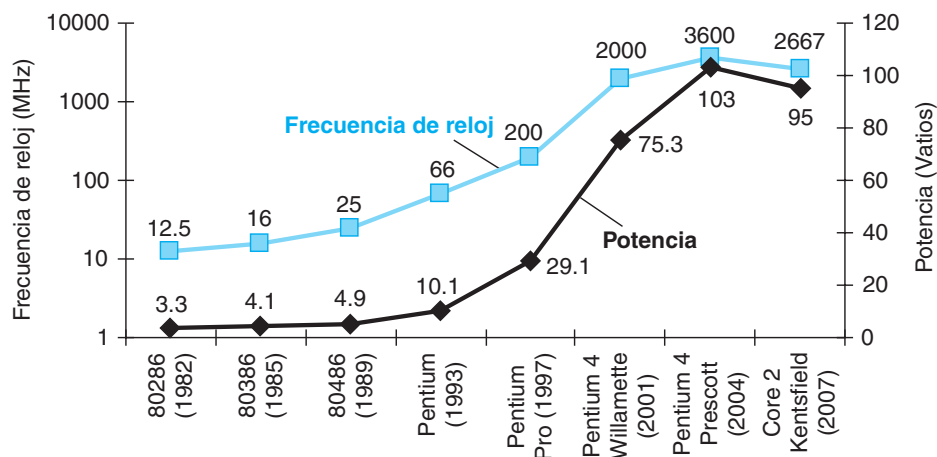


FIGURA 1.15 Variación de la frecuencia de reloj y la potencia en las últimas 8 generaciones en 25 años de microprocesadores x86. La arquitectura Pentium 4 dio un salto muy importante en la frecuencia de reloj y la potencia, pero menos importante en las prestaciones. La línea Pentium 4 se abandonó debido a los problemas térmicos del Prescott. La línea Core 2 vuelve a una segmentación más simple con frecuencias de reloj más bajas y varios procesadores por chip.

1.6

El gran cambio: el paso de monoprocesadores a multiprocesadores

El muro de potencia ha forzado un cambio dramático en el diseño de microprocesadores. La figura 1.16 muestra la mejora en el tiempo de respuesta de programas para microprocesadores de computadores de sobremesa en los últimos años. Desde 2002, el factor de aumento ha disminuido desde un factor 1.5 por año hasta un factor menor que 1.2 por año.

En lugar de continuar con la reducción del tiempo de respuesta de un único programa ejecutándose en un único procesador, desde 2006 todas las compañías de computadores personales y servidores están incorporando microprocesadores con varios procesadores por chip, con los cuales el beneficio es, a menudo, más patente en la productividad que en el tiempo de respuesta. Para eliminar la confusión entre las palabras procesador y microprocesador, los fabricantes utilizan el término “núcleo (*core*)” para referirse a un procesador, y a estos microprocesadores se les denomina habitualmente microprocesadores multinúcleo (*multicore*). Así, un microprocesador *quadcore*, es un chip que contiene cuatro procesadores o cuatro núcleos.

La figura 1.17 muestra el número de procesadores (núcleos), potencia y frecuencias de reloj de varios microprocesadores recientes. La previsión de muchos fabricantes es doblar el número de núcleos por microprocesador en cada generación de tecnología de semiconductores, es decir, cada dos años (véase capítulo 7).

En el pasado, los programadores contaban con las innovaciones en el hardware, la arquitectura y los compiladores para doblar las prestaciones de sus programas cada 18 meses sin tener que cambiar ninguna línea de código. En la actualidad, para obtener mejoras significativas en tiempos de respuesta, los programadores deben reescribir sus programas para extraer las ventajas de disponer de múltiples procesadores. Además, para alcanzar los beneficios históricos de ejecuciones más rápidas en nuevos microprocesadores, los programadores tendrán que continuar mejorando las prestaciones de sus códigos a medida que el número de núcleos se duplica.

Para enfatizar la manera en que hardware y software cooperan, a lo largo del libro usamos la sección especial *interfaz hardware/software*, la primera de las cuales se incluye a continuación. Estos elementos resumen visiones importantes de esta interfaz crítica.

Hasta ahora, la mayor parte del software ha sido como la música escrita para un solista; con la actual generación de microprocesadores, estamos teniendo una pequeña experiencia con duetos y cuartetos y con otros grupos de pocos intérpretes; pero conseguir un trabajo para una gran orquesta y un coro es un reto diferente.

Brian Hayes, *Computing in a Parallel Universe*, 2007.

El paralelismo siempre ha sido un elemento crítico para las prestaciones en computación, pero a menudo ha estado oculto. El capítulo 4 aborda la segmentación, una técnica elegante que permite ejecutar más rápidamente los programas mediante el solapamiento de la ejecución de las instrucciones. Este es un ejemplo del *paralelismo a nivel de instrucciones*, donde la naturaleza paralela del hardware está tan oculta que el programador y el compilador creen que el hardware ejecuta las instrucciones secuencialmente.

Conseguir que los programadores estén pendientes del hardware paralelo y reescriban sus programas explícitamente para ser paralelos había sido la “tercera vía” de la arquitectura de computadores, para aquellas compañías que el pasado dependieron de este cambio de conducta fallido (véase sección 7.14 en el CD). Desde esta perspectiva histórica, resulta asombroso que la industria de las Tecnologías de la

**Interfaz
hardware
software**

Información (TI) al completo haya apostado porque los programadores finalmente se habituarán a una programación explícitamente paralela.

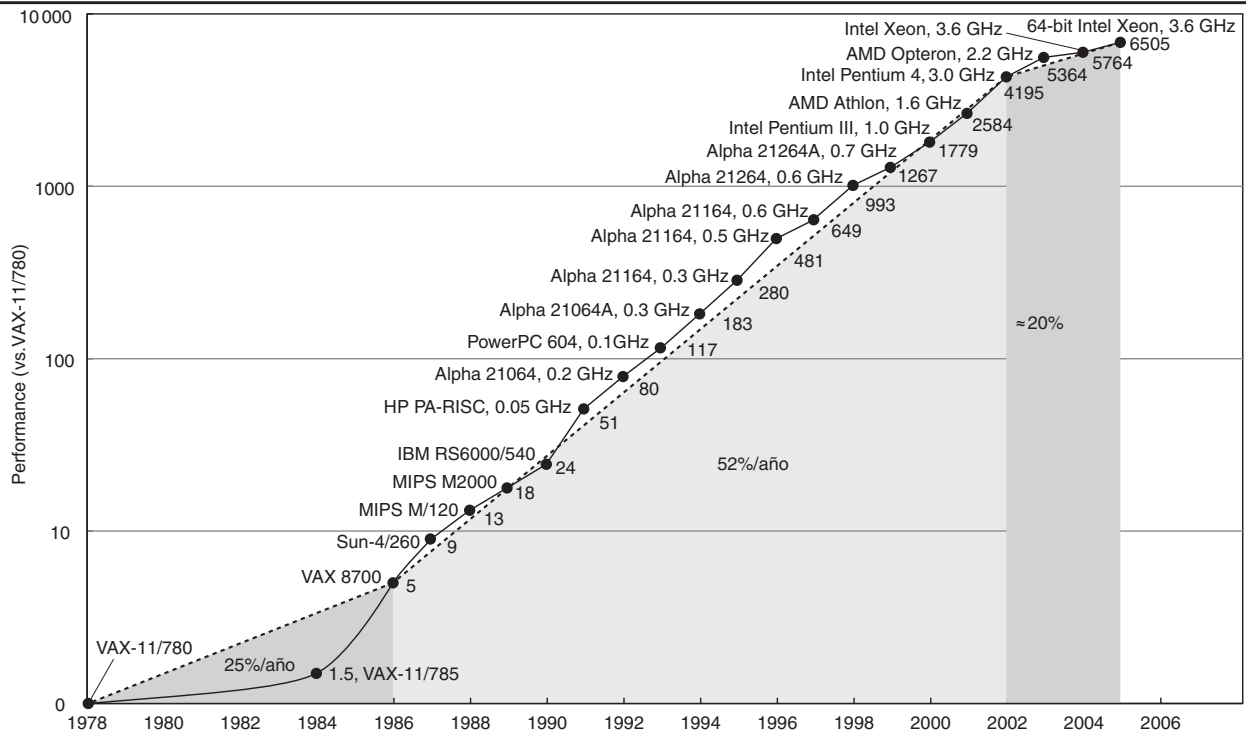


FIGURA 1.16 Aumento de las prestaciones de los procesadores desde la década de 1980. Esta gráfica muestra las prestaciones relativas al VAX 11/780 obtenidas con los programas de prueba SPECint (véase sección 1.8). Antes de mediados de la década de 1980 el aumento de las prestaciones se debía principalmente a la tecnología y de media suponía un 25% cada año. A partir de entonces hay un incremento del 52% durante varios años, atribuible a avances y nuevas ideas en la arquitectura y la organización. En 2002, este aumento había supuesto multiplicar por 7 las prestaciones. Las prestaciones de aplicaciones y cálculos orientados a punto flotante era todavía mayor. Desde 2002, el aumento de las prestaciones de un monoprocesador se redujo a un 20% anual debido a los límites en la disipación de potencia, el paralelismo a nivel de instrucciones disponible y la latencia de la memoria.

Producto	AMD Opteron X4 (Barcelona)	Intel Nehalem	IBM Power 6	Sun Ultra SPARC T2 (Niagara 2)
Núcleos por chip	4	4	2	8
Frecuencia de reloj	2.5 GHz	~ 2.5 GHz ?	4.7 GHz	1.4 GHz
Potencia microprocesador	120 W	~ 100 W ?	~ 100 W ?	94 W

FIGURA 1.17 Número de núcleos por chip, frecuencia de reloj y potencia en varios microprocesadores multinúcleo de 2008.

¿Por qué ha sido tan difícil conseguir que los programadores escribiesen programas explícitamente paralelos? La primera razón es que la programación paralela es por definición programación de prestaciones, que presenta una mayor dificultad. No solamente obliga a que el programa sea correcto, resuelva un problema importante, y proporcione una interfaz útil a los usuarios o a otros progra-

madores que lo utilicen, el programa debe ser también rápido. Por el contrario, si no se necesitan prestaciones, es más sencillo escribir un programa secuencial.

La segunda razón es que rápido para un hardware paralelo significa que el programador debe dividir un aplicación de forma que todos los procesadores tengan que hacer, a grandes rasgos, la misma cantidad de trabajo en el mismo tiempo, y que la sobrecarga debido a la planificación y la coordinación no dilapide los beneficios potenciales del paralelismo.

Podemos utilizar la siguiente analogía. Supongamos la tarea de escribir un artículo para un periódico. Ocho periodistas trabajando en el mismo artículo podrían, potencialmente, escribir el artículo ocho veces más rápido. Para alcanzar este aumento de velocidad, sería necesario dividir la tarea para que cada periodista tuviese algo que hacer en todo momento. Por lo tanto, deberíamos planificar las subtareas. Si algo estuviese mal planificado y un periodista necesitase más tiempo que los otros siete, podrían perderse los beneficios de disponer de ocho escritores. Así, para obtener la aceleración deseada se debe *equilibrar la carga* equitativamente. Otro peligro sería que los periodistas tuviesen que perder demasiado tiempo hablando entre sí para escribir sus secciones. Por otra parte, se podría fracasar en alcanzar ese objetivo si una parte del artículo, por ejemplo las conclusiones, no pudiesen escribirse hasta que todas las demás partes estuviesen terminadas. Así, se debe poner especial atención en reducir la *sobrecarga de las comunicaciones y la sincronización*. Tanto para esta analogía como para la programación paralela, los retos son la planificación, el equilibrio de la carga, el tiempo de sincronización y la sobrecarga de las comunicaciones entre colaboradores. Como se puede adivinar, el reto es todavía mayor con más periodistas para un artículo y más procesadores para programación paralela.

Para analizar la influencia de este gran cambio en la industria, cada uno de los siguientes cinco capítulos de esta edición del libro tienen una sección sobre las implicaciones de la revolución paralela sobre los conceptos de ese capítulo.

- *Capítulo 2, sección 2.11: Paralelismo e instrucciones: Sincronización.* Habitualmente, es necesario coordinar las tareas paralelas independientes en un cierto instante de tiempo, por ejemplo para indicar cuando han terminado su trabajo. En este capítulo se explican las instrucciones de sincronización de tareas utilizadas en los procesadores multinúcleo.
- *Capítulo 3, sección 3.6: Paralelismo y aritmética del computador: Asociatividad.* A menudo los programadores paralelos toman como punto de partida un programa secuencial ya operativo. La pregunta natural para determinar si versión paralela funciona correctamente es “¿se obtiene el mismo resultado?”. Si la respuesta es no, la conclusión lógica es que la versión nueva tiene algún error. Esta lógica está asumiendo que la aritmética del computador es asociativa: el resultado de la suma de un millón de números es el mismo independientemente del orden en que se sumen. En este capítulo se explica que este argumento es válido para números enteros, pero no para número punto flotante.
- *Capítulo 4, sección 4.10: Paralelismo y paralelismo a nivel de instrucciones avanzado.* Dada la dificultad de la programación paralela explícita, desde comienzos de la década de 1990 se ha hecho un gran esfuerzo en el desarrollo de hardware y compiladores que aprovechen el paralelismo implícito. En este capítulo se describen

Pensé (que el computador) sería una idea aplicable universalmente, tal y como lo es un libro. Pero no pensé que se desarrollaría tan rápido como lo ha hecho, porque no imaginé que fuésemos capaces de poner tantos componentes en un chip tal y como finalmente se ha conseguido. El transistor llegó de manera inesperada. Todo sucedió mucho más rápido de lo que esperábamos.

J. Persper Eckert,
coinventor del ENIAC,
hablando en 1991.

1.7

Casos reales: fabricación y evaluación del AMD Opteron x4

Cada capítulo tiene una sección llamada “Casos reales” que enlaza los conceptos del libro con el computador que uno puede usar a diario. Estas secciones informan sobre la tecnología usada en los computadores modernos. En este primer “Caso real”, estudiamos cómo se fabrican los circuitos integrados y cómo se miden las prestaciones y la potencia, tomando el AMD Opteron X4 como ejemplo.

Empecemos por el principio. La fabricación de un chip empieza con el **silicio**, sustancia que se encuentra en la arena. Debido a que el silicio no es un buen conductor, se le llama **semiconductor**. Con un proceso químico especial, es posible añadir al silicio materiales que permiten transformar áreas muy pequeñas en uno de los tres dispositivos siguientes:

- Conductores excelentes de la electricidad (utilizando cables microscópicos de cobre o aluminio).
- Excelentes aislantes de la electricidad (como el plástico o vidrio).
- Áreas que pueden conducir o ser aislantes en determinadas condiciones (como un interruptor).

Los transistores están en la última categoría. Así, un circuito VLSI no es más que miles de millones de conductores, aislantes e interruptores fabricados en un único bloque muy pequeño.

El proceso de fabricación de circuitos integrados es crucial para el coste de los chips y, por lo tanto, importante para los diseñadores de computadores. La figura 1.18 muestra este proceso, que comienza con un **lingote de cristal de silicio**, cuyo aspecto es el de una salchicha gigante. Hoy en día, estas barras miden de unos 15 a 30 cm de diámetro y entre 30 y 60 cm de longitud. Las barras se rebanan muy finamente en **obleas** de menos de 0.25 cm de grosor. A estas obleas se les aplica un proceso de varias etapas en las cuales sobre cada una de ellas se estampan los patrones de productos quí-

Silicio: elemento natural que es semiconductor.

Semiconductor: sustancia que no conduce bien la electricidad.

Lingote de cristal de silicio: barra compuesta de cristal de silicio que tiene entre 15 y 30 cm de diámetro y entre 30 y 60 cm de longitud.

Oblea: rebanada de un lingote de silicio, de grosor no mayor que 0.25 cm, que se usa para fabricar chips.

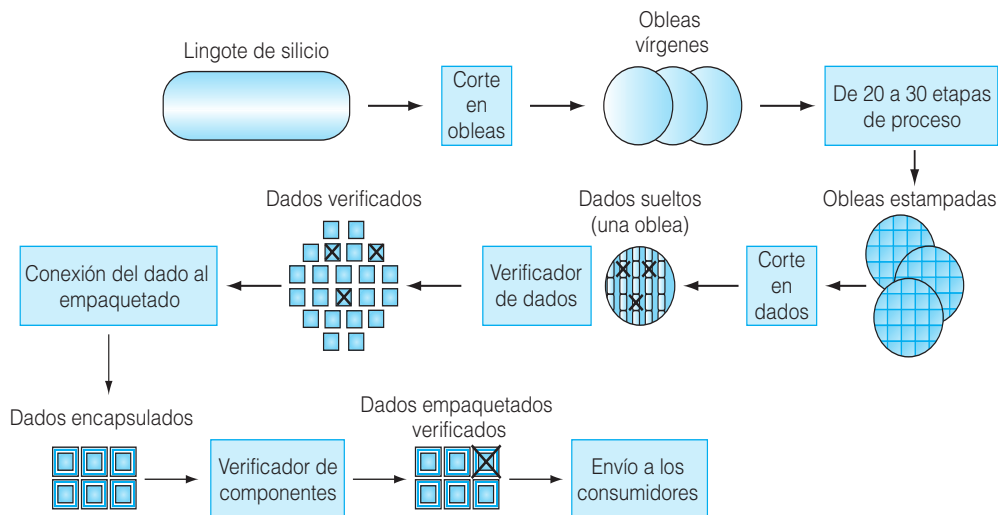


FIGURA 1.18 Proceso de fabricación de chips. Después de cortarse del lingote de silicio, las obleas vírgenes pasan por un proceso que tiene de 20 a 40 etapas para crear obleas estampadas (véase figura 1.9). Estas obleas estampadas se comprueban con un comprobador de obleas y se hace un mapa de las partes correctas. A continuación, las obleas se cortan en dados (véase figura 1.9). En esta figura, una oblea produce 20 dados, de los cuales 17 pasaron con éxito la comprobación. (X significa que el dado es defectuoso.) El factor de producción de dados correctos en este caso es de 17/20, o un 85%. Estos dados correctos se unen a los empaquetados y se comprueban una vez más antes de enviar los chips empaquetados a los clientes. En esta comprobación final se encontró un empaquetado defectuoso.