hiciera la tercera edición de este libro. Los números binarios y los de punto flotante no han sufrido muchos cambios a últimas fechas, por lo que los apéndices son prácticamente los mismos de la edición anterior.

Por último, algunos problemas se modificaron y se añadieron muchos problemas nuevos respecto a la tercera edición.

Existe un sitio Web para este libro. Es posible obtener electrónicamente archivos PostScript para todas las ilustraciones del libro. Estos archivos pueden imprimirse, por ejemplo, para crear transparencias. Además, el sitio incluye un simulador y otras herramientas de software. El URL para este sitio es

http://www.cs.vu.nl/~ast/sco4/

El simulador y las herramientas de software son producto de Ray Ontko. El autor desea expresar su gratitud a Ray por haber producido tan útiles programas.

Varias personas leyeron (parcialmente) el manuscrito y ofrecieron sugerencias útiles o ayudaron de otras maneras. En particular, quiero agradecer a Henri Bal, Alan Charlesworth, Kourosh Gharachorloo, Marcus Goncalves, Karen Panetta Lentz, Timothy Mattson, Harlan McGhan, Miles Murdocca, Kevin Normoyle, Mike O'Connor, Mitsunori Ogihara, Ray Ontko, Aske Plaat, William Potvin II, Nagarajan Prabhakaran, James H. Pugsley, Ronald N. Schroeder, Ryan Shoemaker, Charles Silio, Jr., y Dale Skrien su ayuda, que ha sido inapreciable. Mis estudiantes, sobre todo Adriaan Bon, Laura de Vries, Dolf Loth y Guido van't Noordende, también ayudaron a depurar el texto. Muchas gracias.

Me gustaría expresar un agradecimiento especial a Jim Goodman por sus muchas contribuciones a este libro, sobre todo a los capítulos 4 y 5. La idea de usar la Máquina Virtual Java fue suya, lo mismo que las microarquitecturas para implementarla. Muchas de las ideas avanzadas se deben a él. Este libro es mucho mejor de lo que habría sido sin su esfuerzo.

Por último, quiero agradecer a Suzanne su paciencia ante mis largas horas absorto en mi Pentium. Desde mi punto de vista, el Pentium es mucho mejor que mi antiguo 386, pero a ella no le parece que haya mucha diferencia. También quiero expresar mi gratitud a Barbara y Marvin por ser unos chicos excelentes, y a Bram por permanecer callado cuando yo trataba de escribir.

Andrew S. Tanenbaum

1

INTRODUCCIÓN

Una computadora digital es una máquina que puede resolver problemas ejecutando las instrucciones que recibe de las personas. Una secuencia de instrucciones que describe cómo realizar cierta tarea se llama **programa**. Los circuitos electrónicos de una computadora pueden reconocer y ejecutar directamente un conjunto limitado de instrucciones sencillas, y todos los programas tienen que convertirse en una serie de esas instrucciones para que la computadora pueda ejecutarlos. Dichas instrucciones básicas casi nunca son más complicadas que

Sumar dos números.

Verificar si un número es cero.

Copiar un dato de una parte de la memoria de la computadora a otra.

Juntas, las instrucciones primitivas de una computadora constituyen un lenguaje que permite a las personas comunicarse con la computadora. Dicho lenguaje se llama **lenguaje de máquina**. Las personas que diseñan una computadora nueva deben decidir qué instrucciones incluirán en su lenguaje de máquina. Por lo regular, se trata de hacer las instrucciones primitivas lo más simples posible, en congruencia con el uso que se piensa dar a la computadora y sus requisitos de desempeño, a fin de reducir la complejidad y el costo de los circuitos requeridos. Casi todos los lenguajes de máquina son tan simples que para las personas resulta difícil y tedioso usarlos.

Con el paso de los años, esta sencilla observación ha dado pie a que las computadoras se estructuren como una serie de abstracciones, donde cada una de éstas se apoya en la que está

abajo de ella. De este modo es posible controlar la complejidad y diseñar sistemas de cómputo de manera sistemática y organizada. Llamamos a este enfoque **organización estructurada de computadoras** y éste es también el nombre que hemos dado al presente libro. En la sección que sigue describiremos el significado de este término. Después veremos un poco de historia, hablaremos de lo último en organización de computadoras, y daremos algunos ejemplos importantes.

1.1 ORGANIZACIÓN ESTRUCTURADA DE COMPUTADORAS

Como ya se mencionó, hay una gran diferencia entre lo que es cómodo para las personas y lo que es cómodo para las computadoras. Las personas quieren hacer *X*, pero las computadoras sólo pueden hacer *Y*. Esto crea un problema. El objetivo de este libro es explicar cómo puede resolverse ese problema.

1.1.1 Lenguajes, niveles y máquinas virtuales

Se puede atacar el problema de dos maneras; ambas implican diseñar un nuevo conjunto de instrucciones que para las personas sea más fácil de usar que el conjunto de instrucciones de máquina original. Juntas, estas nuevas instrucciones también forman un lenguaje, que llamaremos L1, así como las instrucciones de máquina originales forman un lenguaje, que nombraremos L0. Las dos estrategias difieren en la forma en que la computadora ejecuta los programas escritos en L1 ya que, como dijimos, la computadora sólo puede ejecutar programas escritos en su lenguaje de máquina, L0.

Un método de ejecutar un programa escrito en L1 es sustituir primero cada instrucción escrita en L1 por una sucesión equivalente de instrucciones en L0. El programa resultante consiste exclusivamente en instrucciones de L0. Luego, la computadora ejecuta el nuevo programa en L0 en lugar del antiguo programa en L1. Esta técnica se llama **traducción**.

La otra técnica consiste en escribir un programa en L0 que tome programas en L1 como datos de entrada y los ejecute examinando sus instrucciones una por una y ejecutando directamente la sucesión de instrucciones en L0 que equivale a cada una. Con esta técnica, no es necesario generar primero un nuevo programa en L0. La técnica se conoce con el nombre de **interpretación** y el programa que la implementa se denomina **intérprete**.

La traducción y la interpretación son similares. Con ambos métodos las instrucciones en L1 se ejecutan finalmente realizando sucesiones equivalentes de instrucciones en L0. La diferencia es que, con la traducción, todo el programa en L1 se convierte primero en un programa en L0, el programa en L1 se desecha, el nuevo programa en L0 se carga en la memoria de la computadora, y se ejecuta. Durante la ejecución, el programa en L0 recién generado es el que se ejecuta y controla la computadora.

Con la interpretación, después de que cada instrucción en L1 se examina y se decodifica, se ejecuta inmediatamente. No se genera ningún programa traducido. Aquí es el intérprete el que controla la computadora. Para él, el programa en L1 no es más que datos. Ambos métodos se usan ampliamente, y cada vez se usa más una combinación de los dos.

En lugar de pensar en términos de traducción o interpretación, a menudo es más fácil imaginar la existencia de una computadora hipotética o **máquina virtual** cuyo lenguaje de máquina es L1. Llamemos a esta máquina virtual M1 (y sea M0 la máquina virtual que corresponde a L0). Si fuera posible construir tal máquina a un costo razonable, no habría necesidad de tener L1 ni una máquina que ejecutara programas en L1. La gente simplemente escribiría sus programas en L1 y la computadora los ejecutaría directamente. Incluso si es demasiado costoso o complicado construir con circuitos electrónicos la máquina virtual cuyo lenguaje es L1, es posible escribir programas para ella. Tales programas podrían interpretarse o traducirse con un programa escrito en L1 que la computadora existente puede ejecutar directamente. En otras palabras, las personas pueden escribir programas para las máquinas virtuales como si realmente existieran.

Para que la traducción o interpretación sea práctica, los lenguajes L0 y L1 no deben ser "demasiado" diferentes. Esta restricción a menudo implica que L1, aunque mejor que L0, todavía dista mucho de ser ideal para la generalidad de las aplicaciones. Este resultado podría ser decepcionante si pensamos en que el propósito original de crear L1 era evitar que el programador tuviera que expresar algoritmos en un lenguaje más apropiado para las máquinas que para las personas. No obstante, la situación no es desesperada.

La estrategia obvia es inventar un tercer conjunto de instrucciones que esté más orientado hacia las personas y menos orientado hacia la máquina que L1. Esas instrucciones también constituyen un lenguaje, al que llamaremos L2 (con una máquina virtual M2). Las personas pueden escribir programas en L2 como si en realidad existiera una máquina virtual M2 cuyo lenguaje de máquina es L2. Esos programas podrían traducirse a L1 o ser ejecutados por un intérprete escrito en L1.

La invención de una serie de lenguajes, cada uno más cómodo que sus predecesores, puede continuar indefinidamente hasta llegar a uno adecuado. Cada lenguaje se basa en su predecesor, por lo que podemos pensar en una computadora que emplea esta técnica como una serie de **capas** o **niveles**, uno encima del otro, como se muestra en la figura 1-1. El lenguaje o nivel más bajo es el más simple, y el lenguaje o nivel más alto es el más sofisticado.

Existe una relación importante entre un lenguaje y una máquina virtual. Cada máquina tiene cierto lenguaje de máquina, que consiste en todas las instrucciones que la máquina puede ejecutar. Efectivamente, una máquina define un lenguaje. De igual modo, un lenguaje define una máquina, la máquina que puede ejecutar todos los programas escritos en ese lenguaje. Desde luego, la máquina definida por un lenguaje dado podría ser enormemente complicada y demasiado costosa para construirse directamente con circuitos electrónicos, pero eso no quiere decir que no podamos imaginarla. Una máquina cuyo lenguaje de máquina es C++ o COBOL sería en verdad compleja, pero con la tecnología actual sería fácil construirla. No obstante, hay razones de peso para no construir semejante computadora: no sería económica en comparación con otras técnicas.

Una computadora con *n* niveles puede verse como *n* máquinas virtuales distintas, cada una con diferente lenguaje de máquina. Usaremos los términos "nivel" y "máquina virtual" indistintamente. Los circuitos electrónicos sólo pueden ejecutar directamente programas escritos en el lenguaje L0 sin necesidad de traducción ni interpretación. Los programas escritos

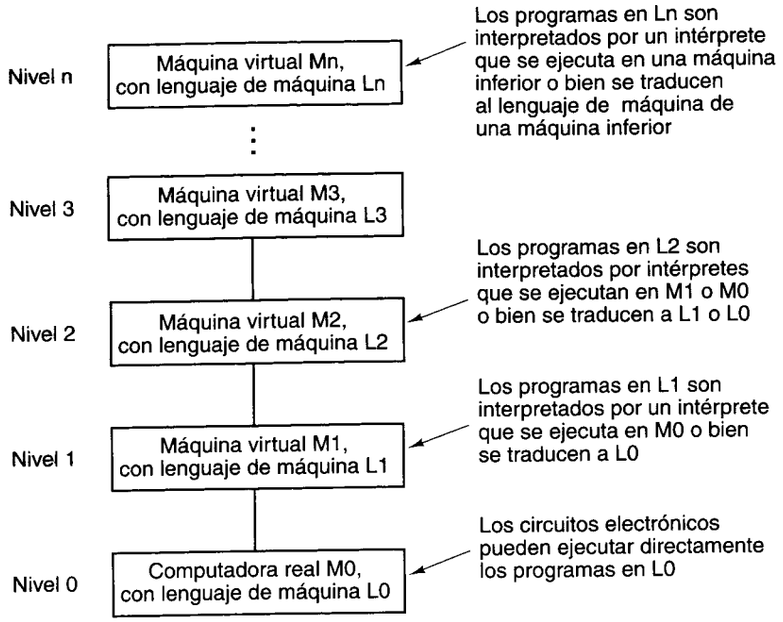

Figura 1-1. Una máquina multinivel.

en L1, L2, . . . , L*n* deberán ser interpretados por un intérprete que se ejecute en un nivel más bajo, o traducirse a otro lenguaje correspondiente a un nivel más bajo.

La persona encargada de escribir programas para la máquina virtual del nivel *n* no tiene que estar consciente de los intérpretes y traductores subyacentes. La estructura de la máquina asegura que de un modo u otro esos programas se ejecutarán. No interesa si un intérprete los ejecuta paso por paso, o si ese intérprete es a su vez ejecutado por otro intérprete o por los circuitos electrónicos. El resultado es el mismo en ambos casos: los programas se ejecutan.

En general, a los programadores que usan una máquina de nivel *n* sólo les interesa el nivel más alto, el que menos se parece al lenguaje de máquina que está hasta abajo. En cambio, las personas interesadas en entender cómo funciona realmente una computadora deben estudiarla en todos los niveles. Quienes se interesen en diseñar nuevas computadoras o nuevos niveles (o sea, nuevas máquinas virtuales) también deberán familiarizarse con niveles distintos del más alto. Los conceptos y técnicas para construir máquinas como una serie de niveles, y los detalles de los niveles mismos, constituyen el tema principal de este libro.

1.1.2 Máquinas multinivel contemporáneas

Casi todas las computadoras modernas constan de dos o más niveles, y pueden llegar a existir máquinas con hasta seis niveles, como se muestra en la figura 1-2. El nivel 0, en la base, es el

verdadero hardware de la máquina. Sus circuitos ejecutan los programas en lenguaje de máquina de nivel 1. Con ánimo totalizador, deberíamos mencionar la existencia de un nivel más bajo de nuestro nivel 0. Este nivel, que no se muestra en la figura 1-2 porque queda dentro del ámbito de la ingeniería eléctrica (y rebasa el alcance de este libro), se llama **nivel de dispositivos**. En este nivel, el diseñador ve transistores individuales, que son las primitivas de más bajo nivel para los diseñadores de computadoras. Si nos preguntamos cómo funcionan internamente los transistores, entramos en la física del estado sólido.

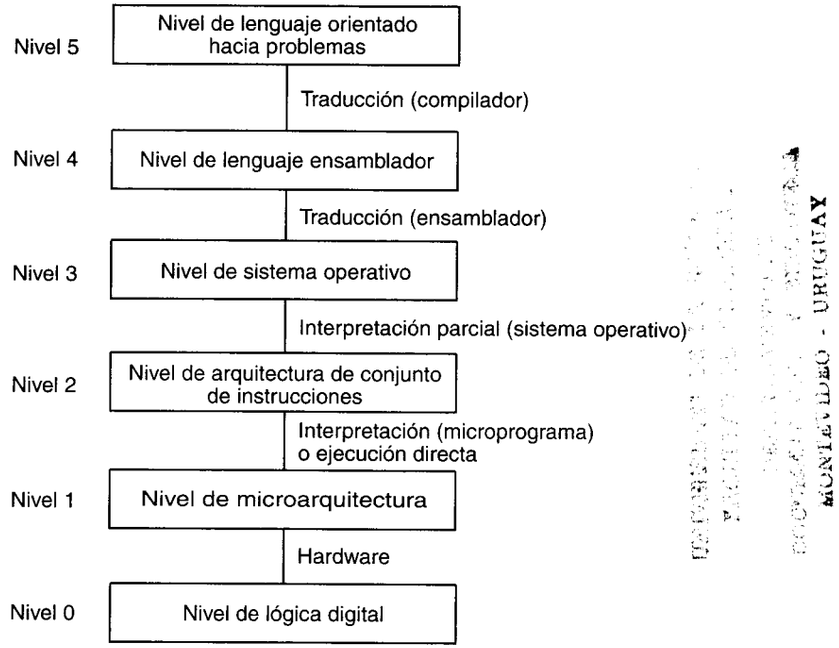

Figura 1-2. Computadora de seis niveles. El método de apoyo para cada nivel se indica inmediatamente abajo de él (junto con el nombre del programa de apoyo).

En el nivel más bajo que estudiaremos, el **nivel de lógica digital**, los objetos integrantes se llaman **compuertas**. Aunque se construyen con componentes analógicos, como los transistores, las compuertas pueden modelarse con exactitud como dispositivos digitales. Cada compuerta tiene una o más entradas digitales (señales que representan 0 o 1) y para generar su salida calcula alguna función sencilla de dichas entradas, como AND u OR. Cada compuerta contiene unos cuantos transistores como máximo. Podemos combinar pocas compuertas para formar una memoria de un bit, capaz de almacenar un 0 o un 1. Las memorias de un bit pueden combinarse en grupos de (por ejemplo) 16, 32 o 64 para formar registros. Cada **registro** puede contener un solo número binario menor que cierto valor límite. Las compuertas también pueden combinarse para formar la máquina calculadora principal misma. Examinaremos las compuertas y el nivel de lógica digital con detalle en el capítulo 3.

El siguiente nivel hacia arriba es el **nivel de microarquitectura**. En este nivel vemos una colección de (típicamente) 8 a 32 registros que forman una memoria local y un circuito llamado **ALU** (**unidad de aritmética lógica**, *Arithmetic Logic Unit*) que puede efectuar operaciones aritméticas sencillas. Los registros se conectan a la ALU para formar una **trayectoria de datos** por donde fluyen los datos. La operación básica de la trayectoria de datos consiste en seleccionar uno o dos registros, hacer que la ALU opere con ellos (sumándolos, por ejemplo), y almacenar después el resultado en algún registro.

En algunas máquinas un programa llamado **microprograma** controla la operación de la trayectoria de datos. En otras máquinas la trayectoria de datos está bajo el control directo del hardware. En ediciones anteriores de este libro llamamos a este nivel el "nivel de microprograma ción" porque en el pasado casi siempre se trataba de un intérprete en software. Puesto que ahora es común que el hardware controle directamente la trayectoria de datos, hemos cambiado el nombre para que refleje este hecho.

En las máquinas con control por software de la trayectoria de datos, el microprograma es un intérprete de las instrucciones en el nivel 2: obtiene, examina y ejecuta las instrucciones una por una, utilizando la trayectoria de datos para hacerlo. Por ejemplo, para una instrucción ADD (sumar), se obtendría la instrucción, se localizarían sus operandos y se colocarían en registros, la ALU calcularía la suma, y por último el resultado se enviaría al lugar correcto. En una máquina con control por hardware de la trayectoria de datos, se llevarían a cabo pasos similares, pero sin un programa almacenado explícito que controle la interpretación de las instrucciones del nivel 2.

En el nivel 2 tenemos un nivel que llamaremos **nivel de arquitectura del conjunto de instrucciones** (*Instruction Set Architecture*) o **nivel ISA**. Cada fabricante de computadoras publica un manual para cada una de las computadoras que vende, intitulado "Manual de referencia del lenguaje de máquina" o "Principios de operación de la computadora Western Wombat Model 100X" o algo parecido. Estos manuales se ocupan realmente del nivel ISA, no de los niveles subyacentes. Cuando describen el conjunto de instrucciones de la máquina, estos manuales están describiendo realmente las instrucciones que el microprograma o los circuitos de ejecución en hardware ejecutan de forma interpretativa. Si un fabricante de computadoras incluye dos intérpretes en una de sus máquinas, para interpretar dos niveles ISA distintos, tendrá que proporcionar dos manuales de referencia del "lenguaje de máquina", uno para cada intérprete.

El siguiente nivel suele ser un nivel híbrido. Casi todas las instrucciones de su lenguaje están también en el nivel ISA. (No hay razón para que una instrucción que aparece en un nivel no pueda estar presente también en otros niveles.) Además, hay un nuevo conjunto de instrucciones, una diferente organización de memoria, la capacidad para ejecutar dos o más programas al mismo tiempo, y varias características más. Entre los distintos diseños de nivel 3 hay más variación que entre los de los niveles 1 o 2.

Las nuevas funciones que se añaden en el nivel 3 son desempeñadas por un intérprete que se ejecuta en el nivel 2, que históricamente se conoce como sistema operativo. El microprograma (o el control por hardware), no el sistema operativo, ejecuta directamente las instrucciones del nivel 3 que son idénticas a las del nivel 2. En otras palabras, algunas de las instrucciones del nivel 3 son interpretadas por el sistema operativo y otras son interpretadas directamente por el microprograma. A esto es a lo que nos referimos con el término "híbrido". Llamamos a este nivel el **nivel de máquina del sistema operativo**.

Existe una discontinuidad fundamental entre los niveles 3 y 4. Los tres niveles más bajos no están diseñados para que sean usados por un programador ordinario. Su propósito primordial es la ejecución de los intérpretes y traductores que se necesitan para apoyar a los niveles superiores. Estos intérpretes y traductores son escritos por personas llamadas **programadores de sistemas** que se especializan en el diseño e implementación de máquinas virtuales nuevas. Los niveles 4 y superiores pertenecen a los programadores de aplicaciones que tienen un problema que resolver.

Otro cambio que ocurre en el nivel 4 tiene que ver con el método de apoyo de los niveles superiores. Los niveles 2 y 3 siempre se interpretan. Los niveles del 4 en adelante por lo regular se traducen, aunque no siempre es así.

Una diferencia más entre los niveles 1, 2 y 3, por un lado, y los niveles del 4 en adelante, por el otro, es la naturaleza del lenguaje empleado. Los lenguajes de máquina de los niveles 1, 2 y 3 son numéricos. Los programas escritos en ellos constan de largas series de números, lo cual es magnífico para las máquinas pero malo para las personas. A partir del nivel 4, los lenguajes contienen palabras y abreviaturas que tienen un significado para las personas.

El nivel 4, el nivel de lenguaje ensamblador, es en realidad una forma simbólica de uno de los lenguajes subyacentes. Este nivel ofrece a las personas un método de escribir programas para los niveles 1, 2 y 3 en una forma no tan incomprensible como los lenguajes de máquina virtuales. Los programas en lenguaje ensamblador primero se traducen a un lenguaje de nivel 1, 2 o 3 y luego se interpretan por la máquina virtual o real apropiada. El programa que realiza la traducción se llama **ensamblador**.

El nivel 5 por lo regular consta de lenguajes diseñados para ser usados por programadores de aplicaciones que quieren resolver problemas. Tales lenguajes suelen llamarse **lenguajes de alto nivel**, y hay literalmente cientos de ellos. Entre los más conocidos están BASIC, C, C++, Java, LISP y Prolog. Los programas escritos en estos lenguajes generalmente se traducen a lenguajes de nivel 3 o 4 con traductores llamados **compiladores**, aunque ocasionalmente se interpretan en vez de traducirse. Los programas en Java, por ejemplo, a menudo se interpretan.

En algunos casos, el nivel 5 consiste en un intérprete para un dominio de aplicación específico, como las matemáticas simbólicas. El nivel proporciona datos y operaciones para resolver problemas de este dominio en términos que la gente que conoce el dominio puede entender fácilmente.

En síntesis, lo más importante que debemos recordar es que las computadoras se diseñan como una serie de niveles, cada uno construido sobre sus predecesores. Cada nivel representa una abstracción distinta, y contiene diferentes objetos y operaciones. Al diseñar y analizar las computadoras de esta manera, podemos suprimir temporalmente los detalles que no son pertinentes y así reducir un tema complejo a algo más fácil de entender.

El conjunto de tipos de datos, operaciones y características de cada nivel es su **arquitectura**. La arquitectura se ocupa de los aspectos que el usuario de ese nivel puede ver. Las características que el programador ve, como la cantidad de memoria disponible, forman parte de la arquitectura. Los aspectos de implementación, como el tipo de tecnología de chips empleado para implementar la memoria, no forman parte de la arquitecutra. El estudio del diseño de las partes de un sistema de cómputo que los programadores pueden ver se llama **arquitectura de computadoras**. En la práctica común, la arquitectura de las computadoras y la organización de las computadoras significan prácticamente lo mismo.

1.1.3 Evolución de las máquinas multinivel

Para entender mejor las máquinas multinivel, examinaremos brevemente su desarrollo histórico, mostrando cómo han evolucionado al paso de los años el número y la naturaleza de los niveles. Los programas escritos en el verdadero lenguaje de máquina de una computadora (nivel 1) pueden ser ejecutados directamente por los circuitos electrónicos de la máquina (nivel 0), sin intervención de intérpretes ni traductores. Estos circuitos electrónicos, junto con la memoria y los dispositivos de entrada/salida, constituyen el **hardware** de la computadora. El hardware consta de objetos tangibles –circuitos integrados, tarjetas de circuitos impresos, cables, fuentes de alimentación, memorias e impresoras– no de ideas abstractas, algoritmos o instrucciones.

El **software**, en cambio, consta de **algoritmos** (instrucciones detalladas que indican cómo hacer algo) y sus representaciones en la computadora: los programas. Los programas se pueden almacenar en un disco duro, disco flexible, CD-ROM u otro medio, pero la esencia del software es la serie de instrucciones que constituye los programas, no los medios físicos en los que se registran.

En las primeras computadoras, la frontera entre hardware y software era muy nítida, pero con el tiempo se ha vuelto mucho más borrosa, principalmente por la adición, eliminación y fusión de niveles a medida que las computadoras evolucionaban. Hoy día, a veces es difícil distinguirlos. De hecho, un tema central de este libro es

El hardware y el software son lógicamente equivalentes.

Cualquier operación realizada por software también puede incorporarse directamente en hardware, de preferencia una vez que se ha entendido muy bien. Como dijo Karen Panetta Lenz: "El hardware no es más que software petrificado." Desde luego, lo contrario también es cierto: cualquier ejecución realizada por el hardware también puede simularse en software. La decisión de colocar ciertas funciones en hardware y otras en software se basa en factores como costo, rapidez, confiabilidad y frecuencia de cambios previstos. Existen pocas reglas rígidas para determinar si X debe incluirse en el hardware y Y debe programarse explícitamente. Tales decisiones cambian con las tendencias en la tecnología y el uso de las computadoras.

La invención de la microprogramación

Las primeras computadoras digitales, en los años cuarenta, sólo tenían dos niveles: el ISA, en el que se efectuaba toda la programación, y el de lógica digital, que ejecutaba esos programas. Los circuitos del nivel de lógica digital eran complicados, difíciles de entender y construir, y poco confiables.

En 1951 Maurice Wilkes, investigador de la Universidad de Cambridge, sugirió la idea de diseñar una computadora de tres niveles a fin de simplificar drásticamente el hardware (Wilkes, 1951). Esta máquina tendría un intérprete inmutable integrado (el microprograma) cuya función sería ejecutar programas del nivel ISA por interpretación. Puesto que el hardware ahora sólo tendría que ejecutar microprogramas, que tienen un conjunto limitado de instrucciones, en lugar de programas del nivel ISA, que tienen un conjunto de instrucciones mucho más grande, se requerirían menos circuitos electrónicos. Dado que en esa época los circuitos se construían con bulbos, tal simplificación prometía reducir el número de bulbos y por tanto mejorar la confiabilidad.

Durante los cincuenta se construyeron unas cuantas máquinas de tres niveles. En los años sesenta se construyeron más, y para 1970 la idea de interpretar el nivel ISA con un microprograma, y no directamente con los circuitos, era preponderante.

La invención del sistema operativo

En estos primeros años, la mayor parte de las computadoras eran de "taller abierto", lo que significa que el programador tenía que operar la máquina personalmente. Junto a cada máquina había una hoja para reservar. Un programador que quería ejecutar un programa reservaba cierto tiempo, digamos el miércoles de las 3 A.M. a las 5 A.M. (a muchos programadores les gustaba trabajar cuando el cuarto de máquinas estaba tranquilo). Al llegar la hora, el programador se dirigía al cuarto de máquinas con un paquete de tarjetas perforadas de 80 columnas (un medio de entrada primitivo) en una mano y un lápiz afilado en la otra. Al llegar al cuarto de la computadora, conducía amablemente al programador anterior hacia la puerta y se hacía cargo de la computadora.

Si el programador quería ejecutar un programa en FORTRAN, era necesario realizar los pasos siguientes:

1. Él† se dirigía al mueble en el que se guardaba la biblioteca de programas, sacaba el gran paquete verde rotulado "compilador FORTRAN", lo colocaba en el lector de tarjetas, y oprimía el botón de inicio.
2. El programador colocaba el programa en FORTRAN en el lector de tarjetas y oprimía el botón de continuar. Se leía el programa.
3. Una vez que la computadora se detenía, él hacía que se leyera otra vez el programa en FORTRAN. Aunque algunos compiladores sólo requerían leer una vez sus entradas, muchos requerían varias pasadas. En cada pasada era necesario leer un paquete grande de tarjetas.
4. Por último, se llegaba a la etapa final de la traducción. El programador a menudo se ponía nervioso a estas alturas porque si el compilador encontraba un error en el programa, el programador tenía que corregirlo e iniciar otra vez el proceso. Si no había errores, el compilador perforaba el programa traducido a lenguaje de máquina en otras tarjetas.
5. Luego, el programador colocaba el programa en lenguaje de máquina en el lector de tarjetas junto con el paquete de la biblioteca de subrutinas para que se leyeran.

†"Él" debe leerse como "él o ella" en todo el libro.

6. El programa comenzaba a ejecutarse. Las más de las veces, no funcionaba y se detenía inesperadamente antes de terminar. En general, el programador movía los interruptores de la consola y miraba las luces de la consola durante un rato. Si tenía suerte, encontraba el problema, corregía el error, y regresaba al mueble que contenía el gran paquete verde del compilador FORTRAN para comenzar otra vez desde el principio. Si no tenía suerte, preparaba un listado del contenido de la memoria, llamado **vaciado de núcleo** (*core dump*) y se lo llevaba a casa para estudiarlo.

Este procedimiento, con variaciones menores, fue normal en muchos centros de cómputo durante años; obligaba a los programadores a aprender a operar la máquina y a saber qué hacer cuando tenía un desperfecto, que era muy seguido. La máquina pasaba mucho tiempo ociosa mientras la gente llevaba tarjetas de un lado a otro del recinto o se rascaba la cabeza tratando de averiguar por qué sus programas no estaban funcionando correctamente.

Alrededor de 1960 la gente trató de reducir el desperdicio de tiempo automatizando la tarea del operador. Un programa llamado **sistema operativo** estaba cargado en la computadora todo el tiempo. El programador incluía ciertas tarjetas de control junto con el programa, y el sistema operativo las leía y ejecutaba sus instrucciones. En la figura 1-3 se muestra un ejemplo de paquete de tarjetas para uno de los primeros sistemas operativos de mayor uso, FMS (FORTRAN Monitor System), en la IBM 709.

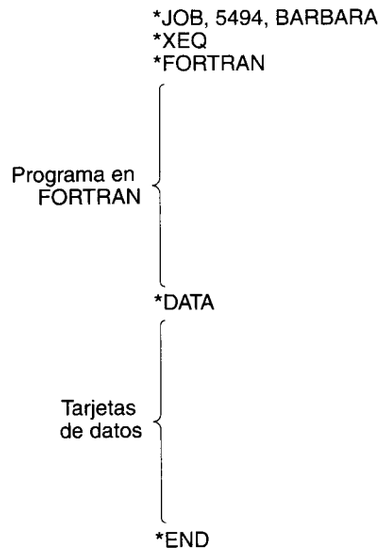

Figura 1-3. Ejemplo de trabajo para el sistema operativo FMS.

El sistema operativo leía la tarjeta *JOB y utilizaba la información que contenía para fines de contabilidad. (El asterisco identificaba las tarjetas de control, para que la computadora no las confundiera con las tarjetas del programa y de datos.) Después, el sistema operativo leía la tarjeta *FORTRAN, que era una instrucción para cargar el compilador FORTRAN de una cinta magnética. Luego, el compilador leía y compilaba el programa en FORTRAN. Cuando el compilador terminaba, devolvía el control al sistema operativo, que entonces leía la tarjeta *DATA. Ésta era una instrucción para ejecutar el programa traducido, utilizando las tarjetas que seguían a la tarjeta *DATA como datos.

Aunque el sistema operativo se diseñó para automatizar la tarea del operador (de ahí el nombre), también fue el primer paso en la creación de una nueva máquina virtual. La tarjeta *FORTRAN podía verse como una instrucción "compilar programa" virtual. Así mismo, la tarjeta *DATA podía considerarse como una instrucción "ejecutar programa" virtual. Un nivel con sólo dos instrucciones no era mucho, pero era un paso en esa dirección.

En años subsecuentes, los sistemas operativos se volvieron más y más sofisticados. Se añadieron nuevas instrucciones, recursos y características al nivel ISA hasta que comenzó a parecer un nuevo nivel. Algunas de las instrucciones de este nuevo nivel eran idénticas a las del nivel ISA, pero otras, sobre todo las de entrada/salida, eran totalmente distintas. Las nuevas instrucciones se conocían como **macros del sistema operativo** o **llamadas al supervisor**. El término usual ahora es **llamada al sistema**.

Los sistemas operativos evolucionaron también en otros sentidos. Los primeros leían paquetes de tarjetas e imprimían sus salidas en la impresora en línea. Esta organización se llamaba **sistema por lotes**. Por lo regular, había una espera de varias horas entre el momento en que se presentaba un programa y el momento en que los resultados estaban listos. Desarrollar software era difícil en esas circunstancias.

A principios de los años sesenta investigadores del Dartmouth College, MIT, y otros sitios crearon sistemas operativos que permitían a (varios) programadores comunicarse directamente con la computadora. En esos sistemas, terminales remotas se conectaban a la computadora central a través de líneas telefónicas. Muchos usuarios compartían el mismo CPU. Un programador podía introducir un programa con el teclado y obtener los resultados impresos casi de inmediato en su oficina, en la cochera de su casa o en donde fuera que se instalara la terminal. Esos sistemas se llamaban, y se siguen llamando, **sistemas de tiempo compartido**.

Nuestro interés en los sistemas operativos se centra en las partes que interpretan las instrucciones y en las características que están presentes en el nivel 3 pero no en el nivel ISA, más que en los aspectos de tiempo compartido. Aunque no haremos hincapié en ello, el lector debe tener presente que los sistemas operativos hacen más que sólo interpretar funciones añadidas al nivel ISA.

Migración de la funcionalidad al microcódigo

Una vez que se hizo común la microprogramación (hacia 1970), los diseñadores se dieron cuenta de que podían añadir nuevas instrucciones con sólo extender el microprograma. En otras palabras, podían agregar "hardware" (nuevas instrucciones de máquina) programando. Esta revelación dio pie a una verdadera explosión en los conjuntos de instrucciones de máquina, a medida que los diseñadores trataban de producir conjuntos de instrucciones más grandes y

mejores que los de otros diseñadores. Muchas de esas instrucciones no eran indispensables en el sentido de que su efecto podía lograrse fácilmente con las instrucciones existentes, pero a menudo eran un poco más rápidas que una sucesión de instrucciones existentes. Por ejemplo, muchas instrucciones tenían una instrucción INC (INCrementar) que sumaba 1 a un número. Puesto que esas máquinas también tenían una instrucción general ADD para sumar, tener una instrucción especial para sumar 1 (o para sumar 270, para el caso) no era necesario. Sin embargo, INC era por lo regular un poco más rápida que ADD, así que se incluyó.

Se añadieron muchas otras instrucciones al microprograma siguiendo el mismo razonamiento. Entre ellas estaban

1. Instrucciones para multiplicación y división de enteros.
2. Instrucciones para aritmética de punto flotante.
3. Instrucciones para invocar procedimientos y regresar de ellos.
4. Instrucciones para acelerar los ciclos.
5. Instrucciones para manejar cadenas de caracteres.

Además, una vez que los diseñadores de máquinas vieron lo fácil que era añadir nuevas instrucciones, comenzaron a buscar otras funciones que añadir a sus microprogramas. Ejemplos de tales adiciones son

1. Funciones para acelerar cálculos con matrices (direccionamiento indexado e indirecto).
2. Funciones para trasladar programas de un lugar a otro de la memoria después de que han iniciado su ejecución (recursos de reubicación).
3. Sistemas de interrupciones que avisan a la computadora cuando se termina una operación de entrada o salida.
4. La capacidad para suspender un programa e iniciar otro en un número reducido de instrucciones (conmutación de procesos).

Con los años, se han añadido muchas otras funciones y recursos, casi siempre para acelerar alguna actividad específica.

La eliminación de la microprogramación

Los microprogramas crecieron durante los años dorados de la microprogramación (años sesenta y setenta). El único problema fue que también tendían a volverse más y más lentos a medida que se hacían más voluminosos. Por fin algunos investigadores se dieron cuenta de que si eliminaban el microprograma, reducían considerablemente el conjunto de instrucciones y hacían que las instrucciones restantes se ejecutaran en forma directa (es decir, control por hardware de la trayectoria de datos), las máquinas podían acelerarse. En cierto sentido, el

diseño de computadoras había descrito un círculo completo, volviendo a la forma que tenía antes de que Wilkes inventara la microprogramación.

El punto importante de esta explicación es mostrar que la frontera entre hardware y software es arbitraria y cambia constantemente. El software de hoy podría ser el hardware de mañana, y viceversa. Además, las fronteras entre los diversos niveles también son variables. Desde el punto de vista del programador, la forma en que se implementa una instrucción dada no es importante (excepto tal vez por su velocidad). Una persona que programa en el nivel ISA puede utilizar su instrucción de multiplicar como si fuera una instrucción de hardware sin preocuparse por ello, o sin siquiera saber si en efecto es una instrucción de hardware. El hardware de una persona es el software de otra. Regresaremos a estos temas más adelante.

1.2 ACONTECIMIENTOS IMPORTANTES EN ARQUITECTURA DE COMPUTADORAS

Se han diseñado y construido cientos de tipos distintos de computadoras durante la evolución de la computadora digital moderna. Casi todos se han perdido en el olvido, pero unos cuantos han tenido un impacto importante sobre las ideas modernas. En esta sección bosquejaremos brevemente algunos de los sucesos históricos clave para entender mejor cómo llegamos a donde estamos ahora. No es necesario decir que esta sección sólo trata los puntos principales y deja muchas cosas sin mencionar. En la figura 1-4 se lee una lista de algunas de las máquinas que trataremos en esta sección. Slater (1987) presenta material histórico adicional acerca de las personas que inauguraron la era de las computadoras. En la obra de Morgan (1997), se encuentran biografías breves, ilustradas con bellas fotografías a color de esos fundadores, tomadas por Louis Fabian Bachrach. Otro libro de biografías es (Slater, 1987).

1.2.1 La generación cero – computadoras mecánicas (1642-1945)

La primera persona que construyó una máquina calculadora funcional fue el científico francés Blaise Pascal (1623-1662), en cuyo honor se nombró el lenguaje de programación Pascal. Este dispositivo, construido en 1642, cuando Pascal tenía apenas 19 años, fue diseñado para ayudar a su padre, un cobrador de impuestos del gobierno francés. La calculadora era totalmente mecánica, con engranes, y se impulsaba con una manivela operada a mano.

La máquina de Pascal sólo podía sumar y restar, pero 30 años después el gran matemático alemán Barón Gottfried Wilhelm von Leibniz (1646-1716) construyó otra máquina mecánica que también podía multiplicar y dividir. Efectivamente, Leibniz había construido el equivalente de una calculadora de bolsillo de cuatro funciones, hace tres siglos.

No sucedió algo relevante durante 150 años hasta que un profesor de matemáticas de la University of Cambridge, Charles Babbage (1792-1871), inventor del velocímetro, diseñó y construyó su **máquina de diferencias**. Este aparato mecánico, que al igual que el de Pascal sólo podía sumar y restar, fue diseñado para calcular tablas de números útiles para la navegación marítima. La máquina entera estaba diseñada para ejecutar un solo algoritmo, el método

Año	Nombre	Hecho por	Comentario
1834	Máquina analítica	Babbage	Primer intento de construir una computadora digital
1936	Z1	Zuse	Primera máquina calculadora de relevadores funcional
1943	COLOSSUS	Gobierno inglés	Primera computadora electrónica
1944	Mark I	Aiken	Primera computadora estadounidense de propósito deneral
1946	ENIAC I	Eckert/Mauchley	Inicia la hitoria moderna de la computación
1949	EDSAC	Wilkes	Primera computadora de programa almacenado
1951	Whielwind I	M.I.T.	Primera computadora de tiempo lineal
1952	IAS	Von Neumann	Casi todas las máqinas actuales emplean este diseño
1960	PDP-1	DEC	Primera minicomputadora (50 vendidas)
1961	1401	IBM	Máquina pequeña para ngocios de enorme popularidad
1962	7094	IBM	Dominó la computación científica a principios de los añossesenta
1963	B5000	Burroughs	Primera máquina diseñada para un lenguaje de alto nivel
1964	360	IBM	Primera línea de productos diseñada como familia
1964	6600	CDC	Primera supercomputadora cietífica
1965	PDP-8	DEC	Primera minicomputadora con mercado masivo (50,000 vedidas)
1970	PDP-11	DEC	Dominó las minicomputadoras en los años setenta
1974	8080	Intel	Primera computadora de propósito general de 8 bits en un chip
1974	CRAY-1	Cray	Primera supercomputadora vectorial
1978	VAX	DEC	Primera superminicomputadora de 32 bits
1981	IBM PC	IBM	Inició la era de la computadora personal moderna
1985	MIPS	MIPS	Primera máquina RISC comercial
1987	SPARC	Sun	Primera estación de trabajo RISC basada en SPARC
1990	RSC6000	IBM	Primera máquina superescalar

Figur a 1-4. Algunos acontecimientos importantes en la evolución de la computadora digital moderna.

de diferencias finitas empleando polinomios. La característica más interesante de la máquina de diferencias era su método de salida: perforaba sus resultados en una placa de cobre para grabados con un troquel de acero, un precursor de posteriores medios de escritura única como las tarjetas perforadas y los CD-ROM.

Aunque la máquina de diferencias funcionaba razonablemente bien, Babbage pronto se aburrió de una máquina que sólo podía ejecutar un algoritmo. Él comenzó a dedicar cantidades cada vez más grandes de su tiempo y de la fortuna de su familia (sin mencionar 17,000 libras esterlinas de dinero del gobierno) en el diseño y construcción de una sucesora llamada **máquina analítica**. La máquina analítica tenía cuatro componentes: el almacén (memoria), el molino (unidad de cómputo), la sección de entrada (lector de tarjetas perforadas) y la

sección de salida (salidas perforadas e impresas). El almacén consistía en 1000 palabras de 50 dígitos decimales, cada una de las cuales servía para almacenar variables y resultados. El molino podía aceptar operandos del almacén, y luego sumarlos, restarlos, multiplicarlos o dividirlos, y por último devolver el resultado al almacén. Al igual que la máquina de diferencias, la máquina analítica era totalmente mecánica.

El gran avance de la máquina analítica fue su utilidad general: leía instrucciones de tarjetas perforadas y las ejecutaba. Algunas instrucciones ordenaban a la máquina obtener dos números del almacén, llevarlos al molino, efectuar operaciones sobre ellos (por ejemplo, sumarlos) y enviar el resultado de vuelta al almacén. Otras instrucciones podían probar un número y efectuar una ramificación condicional dependiendo de si era positivo o negativo. Al perforar un programa distinto en las tarjetas de entrada, era posible lograr que la máquina analítica realizara diferentes cálculos, algo que no podía hacer la máquina de diferencias.

Puesto que la máquina analítica era programable en un sencillo lenguaje ensamblador, necesitaba software. Para producir este software, Babbage contrató a una joven mujer llamada Ada Augusta Lovelace, hija del afamado poeta británico Lord Byron. Así, Ada Lovelace fue la primera programadora de computadoras del mundo. El moderno lenguaje de programación Ada® honra su memoria.

Desafortunadamente, al igual que muchos diseñadores modernos, Babbage nunca logró eliminar todos los problemas del hardware. La dificultad es que necesitaba miles y miles de levas y engranes y ruedas fabricadas con un grado de precisión que la tecnología del siglo XIX no podía alcanzar. No obstante, sus ideas se habían adelantado mucho a su época, e incluso hoy la mayor parte de las computadoras modernas tienen una estructura muy similar a la de la máquina analítica, por lo que es justo decir que Babbage fue el padre (o el abuelo) de la computadora digital moderna.

El siguiente avance importante ocurrió a fines de la década de los treinta, cuando un estudiante de ingeniería alemán llamado Konrad Zuse construyó una serie de máquinas calculadoras automáticas empleando relevadores electromagnéticos. Zuse no logró obtener financiamiento del gobierno después de iniciada la guerra porque los burócratas esperaban ganar la guerra con tal rapidez que la nueva máquina no estaría lista antes de terminar las acciones. Zuse no conocía el trabajo de Babbage, y sus máquinas fueron destruidas por el bombardeo aliado de Berlín en 1944, por lo que su trabajo no influyó en las máquinas subsecuentes. No obstante, Zuse fue uno de los pioneros en este campo.

Poco tiempo después, en Estados Unidos, dos personas diseñaron también calculadoras. John Atanasoff del Iowa State College y George Stibbitz de Bell Labs. La máquina de Atanasoff era asombrosamente avanzada para su época; utilizaba aritmética binaria y tenía una memoria de condensadores, los cuales se renovaban periódicamente para evitar que la carga desapareciera, en un proceso que él llamaba "refrescar la memoria". Los modernos chips de memoria dinámica (RAM) funcionan exactamente de la misma manera. Por desgracia, la máquina nunca logró entrar en operación. En cierto modo, Atanasoff era como Babbage: un visionario que finalmente fue derrotado por la insuficiente tecnología de hardware de su época.

La computadora de Stibbitz, aunque más primitiva que la de Atanasoff, sí funcionaba. Stibbiz hizo una demostración pública en una conferencia en Dartmouth College en 1940. Entre el públi-

co estaba John Mauchley, un profesor de física desconocido de la Universidad de Pennsylvania. El mundo de las computadoras oiría más acerca del profesor Mauchley posteriormente.

Mientras Zuse, Stibbitz y Atanasoff diseñaban calculadoras automáticas, un joven llamado Howard Aiken realizaba tediosos cálculos numéricos a mano como parte de su tesis de doctorado en Harvard. Después de graduarse, Aiken reconoció la importancia de poder efectuar cálculos con una máquina. Él acudió a la biblioteca, descubrió los trabajos de Babbage, y decidió construir con relevadores la computadora de propósito general que Babbage no había logrado construir con ruedas dentadas.

La primera máquina de Aiken, el Mark I, se completó en Harvard en 1944. Tenía 72 palabras de 23 dígitos decimales cada una, y un tiempo de instrucción de 6 segundos. Las entradas y salidas se efectuaban mediante una cinta de papel perforada. Para cuando Aiken terminó su sucesor, el Mark II, las computadoras de relevadores eran obsoletas. Se había iniciado la era de la electrónica.

1.2.2 La primera generación – bulbos (1945-1955)

El estímulo para la computadora electrónica fue la Segunda Guerra Mundial. Durante la primera parte de la guerra, submarinos alemanes hacían estragos entre la armada británica. Los almirantes alemanes enviaban órdenes por radio hasta los submarinos, pero eran interceptadas por los ingleses. El problema es que los mensajes se codificaban mediante un aparato llamado **ENIGMA**, cuyo precursor había sido diseñado por un inventor aficionado y expresidente de Estados Unidos, Thomas Jefferson.

A principios de la guerra, los espías británicos lograron adquirir una máquina ENIGMA de los espías polacos, quienes la habían robado a los alemanes. Sin embargo, se requería un número abrumador de cálculos para decodificar un mensaje, y esto tenía que hacerse inmediatamente después de interceptarse el mensaje para que sirviera de algo. Para decodificar estos mensajes, el gobierno británico estableció un laboratorio supersecreto que construyó una computadora electrónica llamada COLOSSUS. El famoso matemático británico Alan Turing ayudó a diseñar esta máquina. COLOSSUS entró en operación en 1943, pero como el gobierno británico mantuvo prácticamente todos los aspectos del proyecto clasificados como secreto militar durante 30 años, la línea de COLOSSUS fue básicamente un callejón sin salida. Sólo vale la pena mencionarlo porque fue la primera computadora electrónica digital del mundo.

Además de destruir las máquinas de Zuse y estimular la construcción de COLOSSUS, la guerra también afectó la computación en Estados Unidos. El ejército necesitaba tablas de alcance para apuntar su artillería pesada, y producía esas tablas contratando a cientos de mujeres que las elaboraban utilizando calculadoras manuales (se pensaba que las mujeres eran más precisas que los hombres). No obstante, el proceso era muy tardado y los errores eran frecuentes.

John Mauchley, quien conocía el trabajo de Atanasoff y de Stibbitz, se dio cuenta de que el ejército estaba interesado en calculadoras mecánicas. Al igual que muchos científicos de la computación que le siguieron, Mauchley elaboró una propuesta de subvención en la que pedía al ejército fondos para construir una computadora electrónica. La propuesta fue aceptada en 1943, y Mauchley y su estudiante de posgrado, J. Presper Eckert, procedieron a construir una computadora electrónica a la que llamaron **ENIAC** (***Electronic Numerical Integrator And Computer***). ENIAC consistía en 18,000 bulbos y 1500 relevadores, pesaba 30 toneladas y consumía 140 kilowatts de potencia. En términos de arquitectura, la máquina tenía 20 registros, cada uno capaz de almacenar un número decimal de 10 dígitos. (Un registro decimal es una memoria muy pequeña que puede almacenar un número menor que cierto número máximo de dígitos decimales, parecido al odómetro que registra la distancia recorrida por un automóvil.) ENIAC se programaba ajustando 6000 interruptores de multiposición y conectando numerosas bases con una verdadera maraña de cables interconectores.

La máquina no quedó terminada sino hasta 1946, demasiado tarde para realizar su tarea original. Sin embargo, como la guerra había terminado, se les permitió a Mauchley y Eckert organizar un curso de verano para describir el trabajo a sus colegas científicos. Ese curso de verano fue el principio del auge del interés en construir computadoras digitales grandes.

Después de ese histórico curso de verano, muchos otros investigadores se propusieron construir computadoras electrónicas. La primera que entró en operación fue EDSAC (1949), construida por Maurice Wilkes en la Universidad de Cambridge. Otras fueron JOHNIAC, de la Rand Corporation, ILLIAC de la Universidad de Illinois, MANIAC de Los Alamos Laboratory y WEIZAC del Instituto Weizmann de Israel.

Eckert y Mauchley pronto comenzaron a trabajar en un sucesor, **EDVAC** (***Electronic Discrete Variable Automatic Computer***). Sin embargo, ese proyecto quedó condenado al fracaso cuando ellos dejaron la Universidad de Pennsylvania para formar una compañía nueva, la Eckert-Mauchley Computer Corporation, en Filadelfia (todavía no se había creado Silicon Valley). Después de una serie de fusiones, esta compañía se convirtió en la moderna Unisys Corporation.

Como dato curioso, Eckert y Mauchley solicitaron una patente asegurando haber inventado la computadora digital. En retrospectiva, habría sido bueno tener una patente así. Después de años de pleitos, las cortes decidieron que la patente de Eckert y Mauchley no era válida y que John Atanasoff era el inventor de la computadora digital, aunque nunca la haya patentado.

Mientras Eckert y Mauchley estaban trabajando en la EDVAC, una de las personas que participaron en el proyecto ENIAC, John von Neumann, acudió al Institute of Advanced Studies de Princeton para construir su propia versión de EDVAC, la **máquina IAS**. Von Neuman fue un genio del calibre de Leonardo da Vinci; hablaba muchos idiomas, era experto en las ciencias físicas y en matemáticas, y recordaba perfectamente todo lo que había escuchado, visto o leído. Podía citar de memoria el texto exacto de libros que había leído hacía años. Cuando se interesó por las computadoras, ya era el matemático más eminente del mundo.

Una cosa que pronto fue obvia para él era que programar computadoras con un gran número de interruptores era lento, tedioso e inflexible. Von Neumann se dio cuenta de que el programa podía representarse en forma digital en la memoria de la computadora, junto con los datos. Él percibió también que la torpe aritmética decimal en serie utilizada por ENIAC, en la que cada dígito estaba representado por 10 bulbos (uno encendido y nueve apagados) podía ser sustituida por una aritmética binaria.

El diseño básico, que él describió por primera vez, ahora se conoce como **máquina de von Neumann**. Se usó en EDSAC, la primera computadora de programa almacenado, y sigue siendo la base de casi todas las computadoras digitales aun ahora, más de medio siglo después. Este diseño, y la máquina IAS, construida en colaboración con Herman Goldstine, ha tenido una influencia de tal magnitud que vale la pena describirla brevemente. En la figura 1-5 se presenta un bosquejo de su arquitectura.

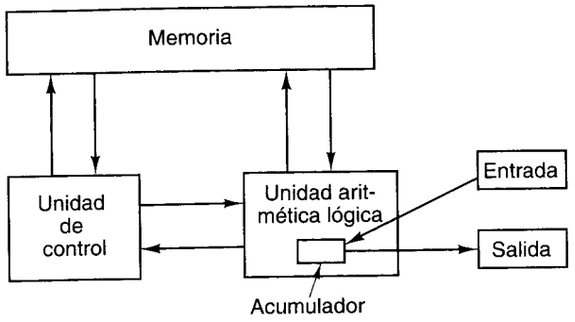

Figura 1-5. La máquina de von Neumann original.

La máquina de von Neumann tenía cinco partes básicas: la memoria, la unidad aritmética lógica, la unidad de control y el equipo de entrada y salida. La memoria constaba de 4096 palabras, cada una de las cuales contenía 40 bits que podían ser 0 o 1. Cada palabra contenía dos instrucciones de 20 bits o bien un entero con signo de 40 bits. Ocho bits de cada instrucción estaban dedicados a indicar el tipo de instrucción, y 12 bits se dedicaban a especificar una de las 4096 palabras de memoria.

Dentro de la unidad de aritmética lógica había un registro interno especial de 40 bits llamado **acumulador**. Una instrucción típica sumaba una palabra de la memoria al acumulador o almacenaba el contenido del acumulador en la memoria. La máquina no tenía aritmética de punto flotante porque von Neumann pensaba que cualquier matemático competente debería poder seguir mentalmente la pista al punto decimal (en realidad el punto binario).

Más o menos en las mismas fechas en que von Neumann estaba construyendo la máquina IAS, investigadores de MIT también estaban construyendo una computadora. A diferencia de IAS, ENIAC y otras máquinas de su clase, que empleaban palabras largas y estaban diseñadas para manipulación pesada de números, la máquina del MIT, Whirlwind I, tenía palabras de 16 bits y estaba diseñada para el control en tiempo real. Este proyecto dio pie a la invención de la memoria de núcleos magnéticos por Jay Forrester, y finalmente a la primera minicomputadora comercial.

Mientras todo esto estaba sucediendo, IBM era una compañía pequeña dedicada a fabricar perforadoras de tarjetas y máquinas clasificadoras de tarjetas. Aunque IBM había aportado parte del financiamiento de Aiken, no estaba muy interesada en las computadoras hasta que produjo la 701 en 1953, mucho después de que la compañía de Eckert y Mauchley se había convertido en la número uno del mercado comercial con su computadora UNIVAC.

La 701 tenía 2048 palabras de 36 bits, con dos instrucciones por palabra. Era la primera de una serie de máquinas científicas que llegaron a dominar la industria en el plazo de una década. Tres años después apareció la 704, que tenía 4K de memoria de núcleos, instrucciones de 36 bits y hardware de punto flotante. En 1958 IBM inició la producción de su última máquina de bulbos, la 709, que básicamente era una 704 más potente.

1.2.3 La segunda generación – transistores (1955-1965)

El transistor fue inventado en los Bell Labs en 1948 por John Bardeen, Walter Brattain y William Schockley, quienes fueron galardonados con el premio Nobel de física de 1956 por su trabajo. En un plazo de 10 años el transistor revolucionó a las computadoras, y para fines de la década de los cincuenta las computadoras de bulbos eran obsoletas. La primera computadora transistorizada se construyó en el Lincoln Laboratory del MIT, una máquina de 16 bits con un diseño parecido al de la Whirlwind I. Se le llamó **TX-0** (computadora Transistorizada eXperimental 0) y su único propósito era probar la TX-2, mucho más elegante.

La TX-2 no logró un progreso decisivo, pero uno de los ingenieros que trabajaban en el laboratorio, Kenneth Olsen, formó una compañía, Digital Equipment Corporation (DEC) en 1957 para fabricar una máquina comercial parecida a la TX-0. Tuvieron que pasar cuatro años para que apareciera esta máquina, la PDP-1, principalmente porque los inversionistas que financiaban a DEC creían firmemente que no había mercado para las computadoras. En vez de ello, DEC se dedicó primordialmente a vender pequeñas tarjetas de circuitos.

Cuando por fin la PDP-1 apareció en 1961, tenía 4K de palabras de 18 bits y un tiempo de ciclo de 5 μs.† Este desempeño era la mitad del de la IBM 7090, la sucesora transistorizada de la 709 y la computadora más rápida del mundo en esas fechas. La PDP-1 costaba $120,000 dólares; la 7090 costaba millones. DEC vendió docenas de PDP-1, y con esto la industria de las minicomputadoras había nacido.

Una de las primeras PDP-1 se entregó a MIT, donde rápidamente atrajo la atención de algunos de los jóvenes genios en formación tan comunes en esa institución. Una de las muchas innovaciones de la PDP-1 era una presentación visual y la capacidad de graficar puntos en cualquier lugar de su pantalla de 512 por 512. En poco tiempo, los estudiantes habían programado la PDP-1 para jugar guerras espaciales, y el mundo tuvo su primer juego de video.

Unos cuantos años después DEC introdujo la PDP-8, que era una máquina de 12 bits, pero mucho más económica que la PDP-1 ($16,000). La PDP-8 tenía una importante innovación: un bus único, u ómnibus, que se muestra en la figura 1-6. Un **bus** es un conjunto de alambres en paralelo que sirve para conectar los componentes de una computadora. Esta arquitectura marcó una divergencia importante respecto a la máquina IAS centrada en la memoria, y ha sido adoptada por casi todas las computadoras pequeñas posteriores. DEC llegó a vender 50,000 PDP-8, lo que la estableció como líder en el negocio de las minicomputadoras

†Prefijos métricos: mili = 10^{-3}, micro (μ) = 10^{-6}, nano = 10^{-9}; kilo = 10^{3}, mega = 10^{6}, giga = 10^{9}

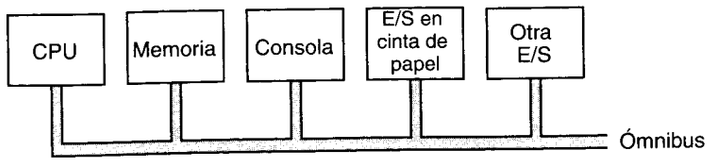

Figura 1-6. El ómnibus de la PDP-8.

Mientras tanto, la reacción de IBM ante el transistor fue construir una versión transistorizada de la 709, la 7090, como ya dijimos, y después la 7094. Ésta tenía un tiempo de ciclo de 2 μs y una memoria de núcleos de 32 K con palabras de 36 bits. La 7090 y la 7094 marcaron el fin de las máquinas tipo ENIAC, y dominaron la computación científica gran parte de la década de los sesenta.

Al mismo tiempo que IBM se convertía en un protagonista importante en la computación científica con la 7094, estaba ganando enormes cantidades de dinero vendiendo una pequeña máquina orientada hacia los negocios: la 1401. Esta máquina podía leer y grabar cintas magnéticas, leer y perforar tarjetas, e imprimir salidas casi tan rápidamente como la 7094, a una fracción del precio. Era muy mala para la computación científica, pero era perfecta para la contabilidad comercial.

La 1401 era poco común en cuanto a que no tenía registros, y ni siquiera una longitud de palabra fija. Su memoria consistía en 4K bytes de 8 bits (4 KB†). Cada byte contenía un carácter de 6 bits, un bit administrativo y un bit que servía para indicar el fin de una palabra. Una instrucción MOVE (mover), por ejemplo, tenía una dirección de origen y una de destino, y comenzaba a transferir bytes del origen al destino hasta encontrar uno cuyo bit de fin de palabra valía 1.

En 1964 una compañía nueva, Control Data Corporation (CDC), introdujo la 6600, una máquina que era casi un orden de magnitud más rápida que la poderosa 7094. Para los "trituradores de números" fue amor a primera vista, y CDC estaba camino al éxito. El secreto de su velocidad, la razón por la que era mucho más rápida que la 7094, era que dentro de la CPU había una máquina con alto grado de paralelismo: contaba con varias unidades funcionales para sumar, otras para multiplicar y otras para dividir, y todas podían operar en paralelo. Aunque para aprovecharla al máximo se requería una programación cuidadosa, con un poco de esfuerzo era posible lograr la ejecución simultánea de 10 instrucciones.

Como si esto no fuera suficiente, la 6600 tenía en su interior varias computadoras pequeñas que la ayudaban, algo así como Blanca Nieves y las siete personas con disfunción vertical. Esto permitía a la CPU dedicar todo su tiempo a la trituración de números, dejando todos los detalles de control de trabajos y entrada/salida a las computadoras pequeñas. En retrospectiva, la 6600 se había adelantado décadas a su época. Muchas de las ideas fundamentales en que se basan las computadoras modernas se remontan directamente a la 6600.

El diseñador de la 6600, Seymour Cray, fue una figura legendaria, comparable con von Neumann. Dedicó su vida a construir máquinas cada vez más rápidas, ahora llamadas

†$1K = 2^{10} = 1024$, así que 1 KB = 1024 bytes; también $1M = 2^{20} = 1{,}048{,}576$ bytes

supercomputadoras, entre las que figuraron la 6600, la 7600 y la Cray-1. También inventó un algoritmo para comprar automóviles que se ha hecho famoso: acudir a la concesionaria más cercana al hogar, apuntar al automóvil más cercano a la puerta, y decir: "Me llevo ése." Este algoritmo desperdicia el mínimo de tiempo en hacer cosas sin importancia (como comprar automóviles) y deja el máximo de tiempo libre para hacer cosas importantes (como diseñar supercomputadoras).

Hubo muchas otras computadoras en esta era, pero una sobresale por una razón muy distinta y vale la pena mencionarla: la Burroughs B5000. Los diseñadores de máquinas como la PDP-1, la 7094 y la 6600 se concentraban totalmente en el hardware, ya sea en bajar su costo (DEC) o en hacerlo más rápido (IBM y CDC). El software casi no venía al caso. Los diseñadores de la B5000 adoptaron una estrategia distinta: construyeron una máquina con la intención específica de programarla en Algol 60, un precursor del Pascal, e incluyeron en el hardware muchas funciones que facilitaban la tarea del compilador. Había nacido la idea de que el software también cuenta. Lo malo es que la idea cayó en el olvido casi de inmediato.

1.2.4 La tercera generación – circuitos integrados (1965-1980)

La invención del circuito integrado de silicio por Robert Noyce en 1958 hizo posible colocar docenas de transistores en un solo chip. Este empaquetamiento permitió construir computadoras más pequeñas, más rápidas y menos costosas que sus predecesores transistorizados. A continuación se describen algunas de las computadoras más importantes de esta generación.

Para 1964 IBM era el principal fabricante de computadoras y tenía un problema grave con sus dos máquinas de mayor éxito, la 7094 y la 1401: eran absolutamente incompatibles. Una era un triturador de números de alta velocidad que empleaba aritmética binaria en paralelo con registros de 36 bits, y la otra era un procesador de entrada/salida que parecía hacer más de lo que en verdad hacía y que utilizaba aritmética decimal en serie con palabras de longitud variable en la memoria. Muchos de sus clientes corporativos tenían ambas máquinas y no les gustaba la idea de mantener dos departamentos de programación distintos que nada tenían en común.

Cuando llegó la hora de sustituir estas dos series, IBM dio un paso radical: introdujo una sola línea de productos, la System/360, basada en circuitos integrados, diseñada para computación tanto científica como comercial. La System/360 contenía muchas innovaciones, siendo la más importante que era una familia de cerca de media docena de máquinas con el mismo lenguaje ensamblador y con tamaño y potencia crecientes. Una compañía podía reemplazar su 1401 por una 360 Model 30 y su 7094 por una 360 Model 75. La Model 75 era más grande y rápida (y más costosa), pero el software escrito para ella podía, en principio, ejecutarse en la Model 30. En la práctica, el software escrito para un modelo inferior se ejecutaba en un modelo superior sin problemas, pero al pasar de una máquina superior a una inferior existía la posibilidad de que el programa no cupiera en la memoria. Aun así, esto representó un avance importante respecto a la situación que prevalecía con la 7094 y la 1401. La idea de las familias de máquinas de inmediato fue bien recibida, y en unos cuantos años casi todos los fabricantes de computadoras tenían una familia de máquinas similares que cubrían una

amplia gama de precio y desempeño. En la figura 1-7 se muestran algunas características de la familia 360 inicial. Posteriormente se introdujeron otros modelos.

Propiedad	Modelo 30	Modelo 40	Modelo 50	Modelo 65
Desempeño relativo	1	3.5	10	21
Tiempo de ciclo (ns)	1000	625	500	250
Memoria máxima (KB)	64	256	256	512
Bytes obtenidos por ciclo	1	2	4	16
Número máximo de canales de datos	3	3	4	6

Figura 1-7. La oferta inicial de la línea de productos IBM 360.

Otra importante innovación de la 360 fue la **multiprogramación**: tener varios programas en la memoria a la vez de modo que mientras uno estaba esperando el término de una operación de entrada/salida otro podía realizar cálculos.

La 360 también fue la primera máquina que podía emular (simular) otras computadoras. Los modelos más pequeños podían emular la 1401, y los más grandes podían emular la 7094, para que los clientes pudieran seguir usando sus antiguos programas binarios sin modificación en tanto efectuaban la conversión a la 360. Algunos modelos ejecutaban los programas de la 1401 con tal rapidez (mucho mayor que en la 1401 misma) que muchos clientes nunca convirtieron sus programas.

La emulación era fácil en la 360 porque todos los modelos iniciales y casi todos los modelos posteriores eran microprogramados. Lo único que tenía que hacer IBM era escribir tres microprogramas: para el conjunto de instrucciones nativo de la 360, para el conjunto de instrucciones de la 1401 y para el conjunto de instrucciones de la 7094. Esta flexibilidad fue una de las razones principales por las que se introdujo la microprogramación.

La 360 resolvió el dilema de binaria paralela *versus* decimal en serie con un término medio: la máquina tenía 16 registros de 32 bits para la aritmética binaria, pero su memoria estaba organizada en bytes, como la de la 1401. Además, contaba con instrucciones seriales tipo 1401 para transferir registros de tamaño variable en la memoria.

Otra característica importante de la 360 fue un espacio de direcciones enorme (para esa época) de 2^{24} bytes (16 megabytes). Puesto que la memoria costaba varios dólares por byte en esos días, 16 megabytes parecía casi infinito. Desafortunadamente, la serie 360 fue seguida más adelante por la serie 370, la 4300, la 3080 y la 3090, todas las cuales empleaban la misma arquitectura. Para mediados de los ochenta, el límite de 16 megabytes se había convertido en un verdadero problema, e IBM tuvo que abandonar parcialmente la compatibilidad al cambiar a las direcciones de 32 bits que se necesitaban para direccionar la nueva memoria de 2^{32} bytes.

En retrospectiva, puede argumentarse que como esas máquinas tenían palabras y registros de 32 bits, probablemente debían haber tenido también direcciones de 32 bits, pero en esa época nadie podía imaginarse una máquina de 16 megabytes. Criticar a IBM por esta falta de visión es como criticar a un moderno fabricante de computadoras personales por tener direcciones de sólo 32 bits. En unos cuantos años las computadoras personales tal vez necesiten más de 4 GB de memoria, y entonces las direcciones de 32 bits serán intolerablemente pequeñas.

El mundo de las minicomputadoras también dio un gran paso en la tercera generación cuando DEC introdujo la serie PDP-11, un sucesor de la PDP-8 de 16 bits. En muchos sentidos, la serie PDP-11 era como el hermano menor de la serie 360, igual que la PDP-1 había sido como un hermanito de la 7094. Tanto la 360 como la PDP-11 tenían registros orientados hacia palabras y memoria orientada hacia bytes, y ambas abarcaban una gama muy amplia de precio y desempeño. La PDP-11 tuvo un éxito enorme, sobre todo en las universidades, y mantuvo la ventaja de DEC sobre los otros fabricantes de minicomputadoras.

1.2.5 La cuarta generación – integración a muy grande escala (1980-?)

Para los años ochenta la **VLSI** (**integración a muy grande escala**, *Very Large Scale Integration*) había hecho posible colocar primero decenas de miles, luego centenares de miles y por último millones de transistores en un solo chip. Este avance dio pie a computadoras más pequeñas y más rápidas. Antes de la PDP-1, las computadoras eran tan grandes y costosas que las compañías y universidades necesitaban departamentos especiales llamados **centros de cómputo** para operarlas. Con la llegada de la minicomputadora, un departamento podía comprar su propia computadora. Para 1980 los precios habían bajado tanto que una persona podía tener su propia computadora. Se había iniciado la era de la computadora personal.

Las computadoras personales se usaban de forma muy diferente que las computadoras grandes; se utilizaban para procesamiento de texto, hojas de cálculo y muchas otras aplicaciones altamente interactivas que las computadoras grandes no podían manejar bien.

Las primeras computadoras personales solían venderse como "kits". Cada kit contenía una tarjeta de circuitos impresos, un puñado de chips, que típicamente incluían un Intel 8080, varios cables, una fuente de potencia y tal vez una unidad de disco flexible de 8 pulgadas. Al comprador correspondía armar las piezas para crear una computadora. No se proporcionaba software. Si usted quería software, lo escribía. Más adelante se popularizó el sistema operativo CP/M para el 8080, escrito por Gary Kildall. Era un verdadero sistema operativo para disco (flexible), con un sistema de archivos y comandos de usuario que se introducían con el teclado.

Otra de las primeras computadoras personales fue la Apple y posteriormente la Apple II, diseñadas por Steve Jobs y Steve Wozniak en la proverbial cochera. Esta máquina se popularizó mucho entre los usuarios caseros y en las escuelas y convirtió a Apple de la noche a la mañana en protagonista.

Después de muchas deliberaciones y de observar lo que otras compañías estaban haciendo, IBM, que entonces era la fuerza dominante en la industria de la computación, por fin decidió que quería ingresar en el negocio de las computadoras personales. En lugar de diseñar una máquina desde cero, empleando únicamente componentes de IBM, lo cual habría tardado mucho tiempo, IBM hizo algo totalmente fuera de carácter: entregó a uno de sus ejecutivos, Philip Estridge, una gran bolsa de dinero y le ordenó irse a algún lugar lejos de los entrometidos burócratas de las oficinas corporativas en Armonk, N.Y., y que no volviera

antes de tener una computadora personal funcional. Estridge se estableció lejos de la oficina central, en Boca Raton, Florida, escogió el Intel 8088 como su CPU, y construyó la IBM Personal Computer a partir de componentes comerciales. La máquina se introdujo en 1981 y se convirtió de inmediato en la computadora más vendida de la historia.

IBM hizo otra cosa fuera de carácter de la que más tarde se arrepentiría. En lugar de mantener totalmente en secreto el diseño de la máquina (o al menos protegido por una barrera de patentes), como normalmente hacía, publicó los planos completos, incluidos todos los diagramas de circuitos, en un libro que vendió a 49 dólares. Lo que se buscaba es que otras compañías fabricaran tarjetas que pudieran insertarse en la IBM PC (plug-ins) a fin de aumentar su flexibilidad y popularidad. Por desgracia para IBM, dado que el diseño ya era totalmente público y todas las piezas se podían conseguir fácilmente de proveedores comerciales con facilidad, muchas otras compañías comenzaron a fabricar **clones** de la PC, casi siempre a un costo mucho menor que lo que IBM cobraba. Fue así como nació toda una industria.

Aunque otras compañías fabricaron computadoras personales utilizando unas CPU que no eran de Intel, como las Commodore, Apple, Amiga y Atari, el ímpetu de la industria de la IBM PC era tan grande que las demás fueron arrolladas. Sólo unas cuantas sobrevivieron, aunque estuvieron restringidas a ciertos nichos del mercado, como estaciones de trabajo de ingeniería o supercomputadoras.

La versión inicial de la IBM PC venía equipada con el sistema operativo MS-DOS provisto por la entonces pequeña Microsoft Corporation. Cuando Intel logró producir CPU cada vez más potentes, IBM y Microsoft pudieron producir un sucesor del MS-DOS llamado OS/2, que contaba con una interfaz gráfica con el usuario similar a la de la Apple Macintosh. Mientras tanto, Microsoft creó también su propio sistema operativo Windows que se ejecutaba encima del MS-DOS, por si acaso el OS/2 no tenía aceptación. Sin abundar en la historia, el OS/2 no experimentó una buena acogida, IBM y Microsoft tuvieron una gran pelea extremadamente pública, y Microsoft se dedicó a hacer de Windows un enorme éxito. La forma en que la diminuta Intel y la todavía más diminuta Microsoft lograron destronar a IBM, una de las corporaciones más grandes, ricas y poderosas en la historia del mundo, es una parábola que sin duda se relata con lujo de detalle en escuelas de negocios de todo el mundo.

Para mediados de los ochenta una nueva idea llamada RISC comenzó a dominar, reemplazando arquitecturas complejas (CISC) por otras mucho más sencillas pero más rápidas. En los noventa comenzaron a aparecer las CPU superescalares. Estas máquinas podían ejecutar varias instrucciones al mismo tiempo, a menudo en un orden distinto del que tenían en el programa. Presentaremos los conceptos de CISC, RISC y superescalar en el capítulo 2 y los trataremos con detalle en todo el libro.

1.3 EL ZOOLÓGICO DE LAS COMPUTADORAS

En la sección anterior presentamos una muy breve historia de los sistemas de cómputo. En ésta examinaremos el presente y miraremos hacia el futuro. Aunque las computadoras personales son las computadoras más conocidas, hay otras clases de máquinas en uso actualmente, y vale la pena dar un vistazo a esas otras computadoras.

1.3.1 Fuerzas tecnológicas y económicas

La industria de la computación está avanzando como ninguna otra. La fue
mordial es la capacidad de los fabricantes de chips para empacar cada vez n
un chip. Un mayor número de transistores, que son diminutos interrupto
implica memorias más grandes y procesadores más potentes.

La rapidez del progreso tecnológico puede modelarse de acuerdo con una observación llamada **ley de Moore**, descubierta por Gordon Moore, cofundador y director de Intel, en 1965. Mientras preparaba un discurso para un grupo de la industria, Moore se dio cuenta de que cada nueva generación de chips de memoria se estaba introduciendo tres años después de la anterior. Puesto que cada nueva generación tenía cuatro veces más memoria que su predecesora, Moore se percató de que el número de transistores en un chip estaba aumentando de forma constante y predijo que este crecimiento continuaría durante varias décadas. Hoy día, la ley de Moore suele expresarse como que el número de transistores se duplica cada 18 meses. Cabe señalar que esto equivale a un incremento de cerca del 60% en el número de transistores cada año. Los tamaños de los chips de memoria y sus fechas de introducción, que se muestran en la figura 1-8, confirman que la ley de Moore se sigue cumpliendo.

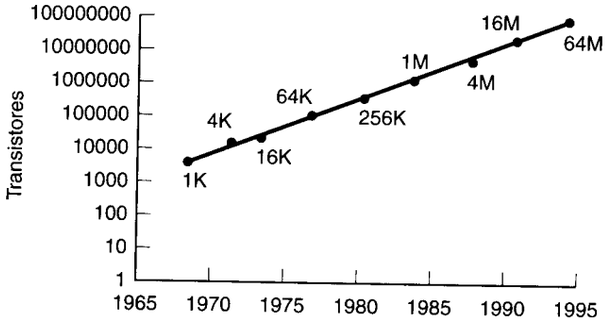

Figura 1-8. La ley de Moore predice un incremento anual de 60% en el número de transistores que se pueden colocar en un chip. Los puntos de datos dados en esta figura son tamaños de memoria, en bits.

Desde luego, la ley de Moore no es realmente una ley, sino sólo una observación empírica acerca de la rapidez con que los físicos de estado sólido y los ingenieros de proceso están empujando hacia adelante las fronteras tecnológicas, y una predicción de que seguirán haciéndolo al mismo ritmo en el futuro. Muchos observadores de la industria esperan que la ley de Moore se seguirá cumpliendo ya entrado el siglo XXI, posiblemente hasta 2020. En ese punto los transistores consistirán en un número demasiado reducido de átomos como para ser confiables, aunque los adelantos en la computación cuántica podrían hacer posible para entonces el almacenamiento de un bit utilizando el espín de un solo electrón.

La ley de Moore ha creado lo que algunos economistas llaman un **círculo virtuoso**. Los adelantos en la tecnología (transistores por chip) dan pie a mejores productos y precios más bajos. Los precios bajos dan pie a nuevas aplicaciones (nadie estaba creando juegos de video

antes de tener una computadora personal funcional. Estridge se estableció lejos de la oficina central, en Boca Raton, Florida, escogió el Intel 8088 como su CPU, y construyó la IBM Personal Computer a partir de componentes comerciales. La máquina se introdujo en 1981 y se convirtió de inmediato en la computadora más vendida de la historia.

IBM hizo otra cosa fuera de carácter de la que más tarde se arrepentiría. En lugar de mantener totalmente en secreto el diseño de la máquina (o al menos protegido por una barrera de patentes), como normalmente hacía, publicó los planos completos, incluidos todos los diagramas de circuitos, en un libro que vendió a 49 dólares. Lo que se buscaba es que otras compañías fabricaran tarjetas que pudieran insertarse en la IBM PC (plug-ins) a fin de aumentar su flexibilidad y popularidad. Por desgracia para IBM, dado que el diseño ya era totalmente público y todas las piezas se podían conseguir fácilmente de proveedores comerciales con facilidad, muchas otras compañías comenzaron a fabricar **clones** de la PC, casi siempre a un costo mucho menor que lo que IBM cobraba. Fue así como nació toda una industria.

Aunque otras compañías fabricaron computadoras personales utilizando unas CPU que no eran de Intel, como las Commodore, Apple, Amiga y Atari, el ímpetu de la industria de la IBM PC era tan grande que las demás fueron arrolladas. Sólo unas cuantas sobrevivieron, aunque estuvieron restringidas a ciertos nichos del mercado, como estaciones de trabajo de ingeniería o supercomputadoras.

La versión inicial de la IBM PC venía equipada con el sistema operativo MS-DOS provisto por la entonces pequeña Microsoft Corporation. Cuando Intel logró producir CPU cada vez más potentes, IBM y Microsoft pudieron producir un sucesor del MS-DOS llamado OS/2, que contaba con una interfaz gráfica con el usuario similar a la de la Apple Macintosh. Mientras tanto, Microsoft creó también su propio sistema operativo Windows que se ejecutaba encima del MS-DOS, por si acaso el OS/2 no tenía aceptación. Sin abundar en la historia, el OS/2 no experimentó una buena acogida, IBM y Microsoft tuvieron una gran pelea extremadamente pública, y Microsoft se dedicó a hacer de Windows un enorme éxito. La forma en que la diminuta Intel y la todavía más diminuta Microsoft lograron destronar a IBM, una de las corporaciones más grandes, ricas y poderosas en la historia del mundo, es una parábola que sin duda se relata con lujo de detalle en escuelas de negocios de todo el mundo.

Para mediados de los ochenta una nueva idea llamada RISC comenzó a dominar, reemplazando arquitecturas complejas (CISC) por otras mucho más sencillas pero más rápidas. En los noventa comenzaron a aparecer las CPU superescalares. Estas máquinas podían ejecutar varias instrucciones al mismo tiempo, a menudo en un orden distinto del que tenían en el programa. Presentaremos los conceptos de CISC, RISC y superescalar en el capítulo 2 y los trataremos con detalle en todo el libro.

1.3 EL ZOOLÓGICO DE LAS COMPUTADORAS

En la sección anterior presentamos una muy breve historia de los sistemas de cómputo. En ésta examinaremos el presente y miraremos hacia el futuro. Aunque las computadoras personales son las computadoras más conocidas, hay otras clases de máquinas en uso actualmente, y vale la pena dar un vistazo a esas otras computadoras.

1.3.1 Fuerzas tecnológicas y económicas

La industria de la computación está avanzando como ninguna otra. La fuerza impulsora primordial es la capacidad de los fabricantes de chips para empacar cada vez más transistores en un chip. Un mayor número de transistores, que son diminutos interruptores electrónicos, implica memorias más grandes y procesadores más potentes.

La rapidez del progreso tecnológico puede modelarse de acuerdo con una observación llamada **ley de Moore**, descubierta por Gordon Moore, cofundador y director de Intel, en 1965. Mientras preparaba un discurso para un grupo de la industria, Moore se dio cuenta de que cada nueva generación de chips de memoria se estaba introduciendo tres años después de la anterior. Puesto que cada nueva generación tenía cuatro veces más memoria que su predecesora, Moore se percató de que el número de transistores en un chip estaba aumentando de forma constante y predijo que este crecimiento continuaría durante varias décadas. Hoy día, la ley de Moore suele expresarse como que el número de transistores se duplica cada 18 meses. Cabe señalar que esto equivale a un incremento de cerca del 60% en el número de transistores cada año. Los tamaños de los chips de memoria y sus fechas de introducción, que se muestran en la figura 1-8, confirman que la ley de Moore se sigue cumpliendo.

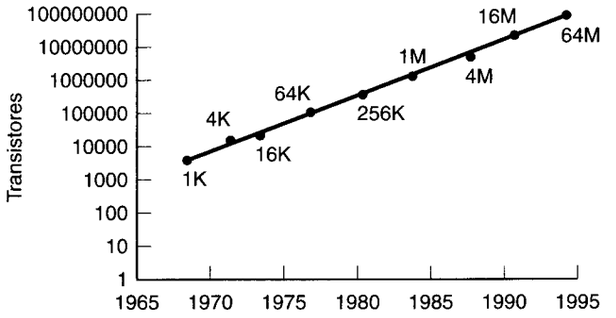

Figura 1-8. La ley de Moore predice un incremento anual de 60% en el número de transistores que se pueden colocar en un chip. Los puntos de datos dados en esta figura son tamaños de memoria, en bits.

Desde luego, la ley de Moore no es realmente una ley, sino sólo una observación empírica acerca de la rapidez con que los físicos de estado sólido y los ingenieros de proceso están empujando hacia adelante las fronteras tecnológicas, y una predicción de que seguirán haciéndolo al mismo ritmo en el futuro. Muchos observadores de la industria esperan que la ley de Moore se seguirá cumpliendo ya entrado el siglo XXI, posiblemente hasta 2020. En ese punto los transistores consistirán en un número demasiado reducido de átomos como para ser confiables, aunque los adelantos en la computación cuántica podrían hacer posible para entonces el almacenamiento de un bit utilizando el espín de un solo electrón.

La ley de Moore ha creado lo que algunos economistas llaman un **círculo virtuoso**. Los adelantos en la tecnología (transistores por chip) dan pie a mejores productos y precios más bajos. Los precios bajos dan pie a nuevas aplicaciones (nadie estaba creando juegos de video

para computadoras cuando éstas costaban 10 millones de dólares cada una). Las nuevas aplicaciones dan pie a nuevos mercados y nuevas compañías que surgen para aprovecharlos. La existencia de todas estas compañías da pie a una competencia que, a su vez, crea una demanda económica de mejores tecnologías con las cuales superar a los demás. Esto completa el círculo.

Otro factor que impulsa el mejoramiento tecnológico es la primera ley del software de Nathan (debida a Nathan Myhrvold, un alto ejecutivo de Microsoft) que dice: "El software es un gas; se expande hasta llenar el recipiente que lo contiene." En los años ochenta el procesamiento de textos se efectuaba con programas como Troff (que se utilizó para este libro). Troff ocupa unas cuantas decenas de kilobytes de memoria. Los modernos procesadores de textos ocupan decenas de megabytes. Los programas futuros sin duda requerirán decenas de gigabytes. El software que continuamente añade funciones (como los cascos de los barcos que sin cesar adquieren bálanos) crea una demanda constante de procesadores más rápidos, memorias más grandes y mayor capacidad de entrada/salida.

Si bien el aumento en el número de transistores por chip con el paso de los años ha sido impresionante, los avances en otras tecnologías de cómputo no se han quedado muy atrás. Por ejemplo, en 1982 se introdujo la IBM PC/XT con un disco duro de 10 MB. Por regla general, los sistemas de escritorio actuales tienen discos un tanto mayores. Medir el mejoramiento de los discos no es tan sencillo, porque implica varios parámetros (capacidad, tasa de datos, precio, etc.), pero casi cualquier métrica indicará que las capacidades han aumentado desde entonces a razón de 50% anual, por lo menos.

Otras áreas que han sido testigo de avances espectaculares son las telecomunicaciones y las redes. En menos de dos décadas hemos pasado de módems de 300 bits/s a módems analógicos de 56 kbps, líneas telefónicas ISDN de 2×64 kbps y redes de fibra óptica de más de un gigabit/s. Los cables telefónicos trasatlánticos de fibras ópticas, como TAT-12/13, cuestan unos 700 millones de dólares, duran 10 años y pueden transportar 300,000 llamadas simultáneas, lo que da un costo de menos de un centavo de dólar por una llamada intercontinental de 10 minutos. En el laboratorio se ha demostrado la factibilidad de sistemas de comunicación ópticos que operan a 1 terabit/s (10^{12} bits/s) a distancias de más de 100 km sin amplificadores. El crecimiento exponencial de Internet es tan evidente que no hace falta mencionarlo aquí.

1.3.2 La gama de las computadoras

Richard Hamming, antiguo investigador de Bell Labs, cierta vez observó que un cambio de un orden de magnitud en la cantidad causa un cambio en la calidad. Así, un automóvil que puede correr a 1000 km/h en el desierto de Nevada tiene un tipo de máquina fundamentalmente distinto de la de un automóvil normal que viaja a 100 km/h en una autopista. Así mismo, un rascacielos de 100 pisos no es sólo un edificio de apartamentos de 10 pisos a mayor escala. Y con las computadoras no estamos hablando de factores de 10, sino de factores de un millón en el curso de tres décadas.

Los aumentos predichos por la ley de Moore se pueden utilizar de varias maneras. Una de ellas es construir computadoras cada vez más potentes que no cuesten más. Otro enfoque es construir la misma computadora con menos y menos dinero cada año. La industria de la computación ha hecho ambas cosas y más, y el resultado es la amplia variedad de computadoras

que se venden actualmente. En la figura 1-9 se presenta una clasificación muy burda de las computadoras actuales.

Tipo	Precio	Ejemplo de aplicación
Computadora desechable	1	Tarjetas de felicitación
Computadora incorporada	10	Relojes, automóviles, aparatos
Computadora de juegos	100	Juegos de video caseros
Computadora personal	1 K	Computadora de escritorio o portátil
Servidor	10 K	Servidor de red
Colección de estaciones de trabajo	100 K	Minisupercomputadora
Mainframe (macrocomputadora)	1 M	Procesamiento de datos por lotes en un banco
Supercomputadora	10 M	Predicción del tiempo a largo plazo

Figura 1-9. Gama de computadoras disponibles en la actualidad. Los precios no deben tomarse muy en serio.

En el extremo inferior encontramos chips individuales pegados al interior de tarjetas de felicitación para tocar "Feliz cumpleaños", la "Marcha nupcial" o alguna otra tonadilla igualmente abominable. El autor todavía no ha encontrado una tarjeta de condolencias que toque una marcha fúnebre, pero después de haber sacado esta idea al dominio público, se cree que no tardará en aparecer. Para cualquier persona que haya crecido con mainframes de millones de dólares, la idea de una computadora desechable es tan lógica como la de un avión desechable. No obstante, es indudable que las computadoras desechables llegaron para quedarse. (¿Qué tal bolsas para la basura parlantes que nos recuerden reciclar las latas de aluminio?)

En el siguiente peldaño tenemos las computadoras que están incorporadas en teléfonos, televisores, hornos de microondas, reproductores de CD, juguetes, muñecas y miles de aparatos más. En unos cuantos años, todo artículo eléctrico incluirá una computadora. El número de computadoras incorporadas se medirá en miles de millones, rebasando por órdenes de magnitud a todas las demás computadoras combinadas. Estas computadoras contienen un procesador, menos de un megabyte de memoria, y cierta capacidad de E/S, todo en un solo chip que cuesta unos cuantos dólares.

En el siguiente peldaño están las máquinas de videojuegos. Se trata de computadoras normales, con capacidades gráficas especiales, pero software limitado y casi ninguna extensibilidad. En este intervalo de precios también están los organizadores personales, los asistentes digitales portátiles y otros dispositivos de computación de bolsillo similares, así como las computadoras de red y las terminales de Web. Lo que estas computadoras tienen en común es que contienen un procesador, unos cuantos megabytes de memoria, algún tipo de pantalla (posiblemente un televisor) y poco más. Eso es lo que los hace tan baratos.

A continuación vienen las computadoras personales en las que la mayoría de la gente piensa cuando oye la palabra "computadora". Éstas incluyen modelos de escritorio y portátiles. Por lo regular incluyen varios megabytes de memoria, un disco duro con unos cuantos gigabytes

de datos, una unidad de CD-ROM, un módem, una tarjeta de sonido y otros periféricos. Estas máquinas cuentan con sistemas operativos complejos, muchas opciones de expansión y un enorme surtido de software comercial. Las que tienen una CPU Intel suelen llamarse "computadoras personales", mientras que las que tienen otro tipo de CPU a menudo se llaman "estaciones de trabajo", aunque en lo conceptual casi no hay diferencias entre los dos tipos.

Ciertas computadoras personales o estaciones de trabajo mejoradas suelen utilizarse como servidores de red, tanto para redes de área local (casi siempre dentro de una sola compañía) como para Internet. Éstas tienen configuraciones de un solo procesador o de múltiples procesadores, tienen unos cuantos gigabytes de memoria, muchos gigabytes de espacio en disco duro, y capacidad de trabajo en red de alta velocidad. Algunas de ellas pueden manejar docenas o cientos de llamadas a la vez.

Más allá de los servidores multiprocesador pequeños encontramos sistemas conocidos como sistemas **NOW (redes de estaciones de trabajo**, *Networks of Workstations*) o **COW (cúmulos de estaciones de trabajo**, *Clusters of Workstations*). Éstos son computadores personales o estaciones de trabajo estándar conectadas por redes de gigabits/s y que ejecutan programas especiales que permiten a todas las máquinas colaborar en la solución de un solo problema, que a menudo es científico o de ingeniería. Es fácil cambiar la escala de estos sistemas, desde un puñado de máquinas hasta miles de ellas. Gracias al bajo precio de sus componentes, los departamentos independientes pueden poseer tales máquinas, que de hecho son minisupercomputadoras.

Ahora llegamos a las mainframes, computadoras que llenan una habitación y que se remontan a los sesenta. En muchos casos, estas máquinas son descendientes directos de mainframes IBM 360 adquiridas décadas atrás. En su mayor parte, estas computadoras no son mucho más rápidas que servidores potentes, pero suelen tener mayor capacidad de E/S y están equipadas con grandes "granjas" de discos que llegan a contener un terabyte de datos o más (1 TB = 10^{12} bytes). Aunque son extremadamente costosas, estas máquinas a menudo se mantienen funcionando en vista de la enorme inversión en software, datos, procedimientos operativos y personal que representan. Muchas compañías consideran que es más económico pagar unos cuantos millones de dólares de vez en cuando por una nueva, que siquiera contemplar el esfuerzo necesario para reprogramar todas sus aplicaciones para máquinas más pequeñas.

Es esta clase de computadoras la que dio pie al problema del Año 2000 de triste fama, el cual es culpa de los programadores en COBOL de los sesenta y setenta que representaban el año como dos dígitos decimales (para ahorrar memoria). Ellos nunca imaginaron que su software duraría tres o cuatro décadas. Muchas compañías han repetido el mismo error al limitarse a agregar dos dígitos más al año. El autor aprovecha la ocasión para predecir el fin de la civilización que conocemos a la media noche del 31 de diciembre de 9999, cuando 8000 años de viejos programas en COBOL dejarán de funcionar simultáneamente.

Después de las mainframes vienen las verdaderas supercomputadoras, cuyas CPU son inconcebiblemente rápidas, tienen muchos gigabytes de memoria principal, y sus discos y sus redes muy rápidos. En años recientes, muchas de las supercomputadoras se han convertido en máquinas altamente paralelas, no muy diferentes de los diseños COW, pero con componentes más rápidos y numerosos. Las supercomputadoras se emplean para resolver problemas que requieren muchos cálculos en ciencia e ingeniería, como simular el choque de galaxias, sintetizar nuevas medicinas o modelar el flujo del aire alrededor de un ala de avión.

1.4 EJEMPLOS DE FAMILIAS DE COMPUTADORAS

En esta sección presentaremos una breve introducción a las tres computadoras que usaremos como ejemplos en el resto del libro: la Pentium II, la UltraSPARC II y la picoJava II [sic].

1.4.1 Introducción al Pentium II

En 1968 Robert Noyce, inventor del circuito integrado de silicio; Gordon Moore, cuya ley ya es famosa, y Arthur Rock, un inversionista de San Francisco, formaron la Intel Corporation para fabricar chips de memoria. En su primer año de operación, Intel sólo vendió 3000 dólares de chips, pero el negocio ha prosperado desde entonces.

A fines de los años sesenta, las calculadoras eran grandes máquinas electromecánicas del tamaño de una moderna impresora láser y con un peso de unos 20 kg. En septiembre de 1969 una compañía japonesa, Busicom, se acercó a Intel para pedirle que fabricara 12 chips a la medida para una calculadora electrónica propuesta. El ingeniero de Intel asignado a este proyecto, Ted Hoff, examinó el plan y se percató de que podía poner una CPU de 4 bits de propósito general en un solo chip, y que esto efectuaría el mismo trabajo y sería además más simple y económico. Fue así como en 1970 nació la primera CPU en un solo chip, la 4004 de 2300 transistores (Faggin *et al.*, 1996).

Vale la pena señalar que ni Intel ni Busicom tenían la menor idea de lo que acababan de hacer. Cuando Intel decidió que valdría la pena hacer el intento de usar la 4004 en otros proyectos, ofreció comprar a Busicom los derechos del nuevo chip devolviéndole los 60,000 dólares que ésta había pagado a Intel por desarrollarlo. Busicom aceptó rápidamente el ofrecimiento de Intel, y éste comenzó a trabajar en una versión de ocho bits del chip, el 8008, introducido en 1972.

Intel no esperaba mucha demanda por el 8008, así que estableció una línea de producción de bajo volumen. Para asombro de todos, el chip despertó un interés enorme, por lo que Intel se dedicó a diseñar un nuevo chip de CPU que superara la limitación de 16K de memoria del 8008 (impuesta por el número de agujas en el chip). Este diseño dio como resultado el 8080, una CPU pequeña, de propósito general, introducido en 1974. Como había hecho la PDP-8, este producto tomó a la industria por asalto y de la noche a la mañana se convirtió en un artículo de mercado masivo. Sólo que en lugar de vender millares, como había hecho DEC, Intel vendió millones.

En 1978 llegó el 8086, una verdadera CPU de 16 bits en un solo chip. El 8086 tenía un diseño similar al del 8080, pero no era totalmente compatible con éste. Al 8086 siguió el 8088, que tenía la misma arquitectura que el 8086 y ejecutaba los mismos programas, pero tenía un bus de 8 bits en lugar de uno de 16 bits, lo que lo hacía más lento pero más económico que el 8086. Cuando IBM escogió el 8088 como CPU para la IBM PC original, este chip rápidamente se convirtió en el estándar para la industria de las computadoras personales.

Ni el 8088 ni el 8086 podían direccionar más de un megabyte de memoria. Para principios de los ochenta esto se convirtió en un problema cada vez más grave, por lo que Intel diseñó el 80286, una versión del 8086 compatible hacia arriba. El conjunto de instruc-

ciones básico era casi el mismo que el del 8086 y el 8088, pero la organización de la memoria era muy diferente y un tanto torpe, debido al requisito de compatibilidad con los chips anteriores. El 80286 se usó en la IBM PC/AT y en los modelos PS/2 de mediano precio. Al igual que el 8088, este chip tuvo un éxito enorme, principalmente porque la gente lo veía como un 8088 más rápido.

El siguiente paso lógico era una verdadera CPU de 32 bits en un chip, el 80386, que salió al mercado en 1985. Al igual que el 80286, éste era más o menos compatible con todos los chips anteriores, hasta el 8080. Ser compatible con lo existente era una bendición para la gente que necesitaba ejecutar software viejo, pero un fastidio para quienes habrían preferido una arquitectura sencilla, limpia y moderna que no cargara con los errores y la tecnología del pasado.

Cuatro años después salió el 80486. Éste era básicamente una versión más rápida del 80386 que además tenía una unidad de punto flotante y 8K de memoria caché en el chip. La **memoria caché** sirve para retener las palabras de memoria más comúnmente usadas dentro o cerca de la CPU, para evitar los (lentos) accesos a la memoria principal. El 80386 también contaba con apoyo de multiprocesador integrado, que permitía a los fabricantes construir sistemas con múltiples CPU.

En este punto Intel se enteró por las malas (al perder una demanda legal de violación de marca comercial) que los números (como 80486) no pueden registrarse como marcas, así que la nueva generación recibió un nombre: Pentium (del vocablo griego penta, cinco). A diferencia del 80486, que tenía una fila de procesamiento, el Pentium tenía dos, y esto lo ayudaba a ser dos veces más rápido (trataremos las filas de procesamiento con detalle en el capítulo 2).

Cuando apareció la siguiente generación, la gente que esperaba el Sexium (*sex* es seis en latín) sufrió una decepción. El nombre Pentium ya era tan conocido que la gente de comercialización quería conservarlo, así que el nuevo chip recibió el nombre de Pentium Pro. A pesar de lo trivial del cambio en el nombre respecto al de su predecesor, este procesador representó una importante ruptura con el pasado. En lugar de tener dos o más filas de procesamiento, el Pentium Pro tenía una organización interna muy distinta y podía ejecutar hasta cinco instrucciones a la vez.

Otra innovación del Pentium Pro fue una memoria caché de dos niveles. El chip procesador mismo tenía 8 KB de memoria para retener las instrucciones de uso común, y 8 KB de memoria para retener datos de uso común. En la misma cavidad dentro del paquete Pentium Pro (pero no en el chip mismo) había una segunda memoria caché de 256 KB.

El siguiente procesador Intel fue el Pentium II, que era básicamente un Pentium Pro con la adición de extensiones multimedia especiales (llamadas **MMX**). Estas instrucciones estaban diseñadas para acelerar los cálculos requeridos para procesar audio y video, lo que hacía innecesaria la adición de coprocesadores de multimedia especiales. Estas instrucciones también estuvieron disponibles en procesadores Pentium posteriores, pero no en el Pentium Pro, de modo que el Pentium II combinaba las ventajas del Pentium Pro con multimedios.

A principios de 1998 Intel introdujo una nueva línea de productos llamada **Celeron**, que es básicamente una versión de bajo costo y bajo desempeño del Pentium II proyectada para PC del extremo inferior. Puesto que el Celeron tiene la misma arquitectura que el Pentium II, no hablaremos más de él en este libro. En junio de 1998 Intel introdujo una versión especial del Pentium II para el extremo superior del mercado. Este procesador, llamado **Xeon**, tiene

un caché más grande, un bus más rápido y mejor apoyo multiprocesador, pero por lo demás es un Pentium II normal, así que tampoco lo trataremos individualmente. La familia Intel se muestra en la figura 1-10.

Chip	Fecha	MHz	Transistores	Memoria	Notas
4004	4/1971	0.108	2,300	640	Primer microprocesador en un chip
8008	4/1972	0.108	3,500	16 KB	Primer microprocesador de 8 bits
8080	4/1974	2	6,000	64 KB	Primera CPU de propósito general en un bit
8086	6/1978	5-10	29,000	1 MB	Primera CPU de 16 bits en un chip
8088	6/1979	5-8	29,000	1 MB	Se usó en la IBM PC
80286	2/1982	8-12	134,000	16 BM	Cunta con protección de memoria
80386	10/1985	16-33	275,000	4 GB	Primer CPU de 32 bits
80486	4/1989	25-100	1.2M	4 GB	Memoria caché de 8K integrada
Pentium	3/1993	60-233	3.1M	4 GB	Dos conductos; modelos posteriores tenían MMX
Pentium Pro	3/1995	150-200	5.5M	4 GB	Dos niveles de caché integrados
Pentium II	5/1997	233-400	7.5M	4 GB	Pentium Pro más MMX

Figura 1-10. La familia de CPU Intel. Las velocidades de reloj se miden en MHz (megahertz), donde 1 MHz es 1 millón de ciclos/s.

Todos los chips Intel son compatibles con sus predecesores hasta el 8086. En otras palabras, un Pentium II puede ejecutar programas del 8086 sin modificación. Esta compatibilidad siempre ha sido un requisito de diseño para Intel, a fin de que los usuarios puedan mantener su inversión en software existente. Desde luego, el Pentium II es 250 veces más complejo que el 8086, así que puede hacer una buena cantidad de cosas que el 8086 nunca pudo hacer. Estas extensiones incrementales han dado pie a una arquitectura menos elegante de lo que podría haber sido si alguien hubiera entregado a los arquitectos del Pentium II 7.5 millones de transistores e instrucciones para comenzar otra vez desde el principio.

Resulta interesante que si bien la ley de Moore ha estado asociada desde hace mucho al número de bits de una memoria, aplica igualmente bien a los chips de CPU. Si graficamos los números de transistores dados en la figura 1-10 contra la fecha de introducción de cada chip en una escala semilogarítmica, vemos que la ley de Moore también se cumple. La gráfica correspondiente se presenta en la figura 1-11.

1.4.2 Introducción al UltraSPARC II

En los setenta UNIX era popular en las universidades, pero ninguna computadora personal lo ejecutaba, así que los amantes de UNIX tenían que usar minicomputadoras de tiempo compar-

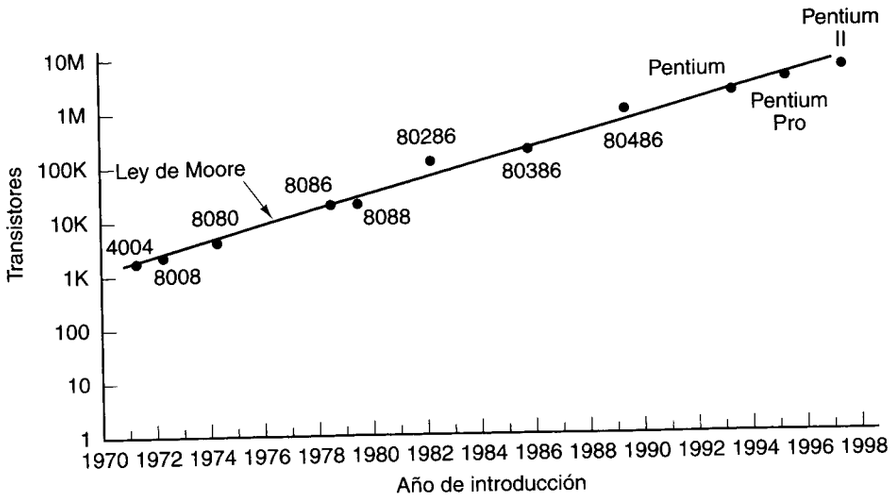

Figura 1-11. Ley de Moore para chips de CPU.

tido (a menudo sobrecargadas) como la PDP-11 y VAX. En 1981 un estudiante de posgrado alemán en Stanford, Andy Bechtolsheim, quien se sentía frustrado por tener que acudir al centro de cómputo para usar UNIX, decidió resolver el problema construyendo para su uso personal una estación de trabajo UNIX utilizando componentes comerciales, y la llamó SUN-1 (Stanford University Network).

Bechtolsheim pronto atrajo la atención de Vinod Khosla, un hindú de 27 años que tenía el deseo ferviente de retirarse como millonario antes de cumplir los 30 años. Khosla convenció a Bechtolsheim de formar una compañía para construir y vender estaciones de trabajo Sun. Luego, Khosla contrató a Scott McNealy, otro estudiante de posgrado de Stanford, para encargarse del área de producción. Para escribir el software contrataron a Bill Joy, el principal arquitecto del Berkeley UNIX. Los cuatro fundaron Sun Microsystems en 1982.

El primer producto de Sun, la Sun-1, basada en la CPU Motorola 68020, tuvo un éxito inmediato, lo mismo que las máquinas subsecuentes Sun-2 y Sun-3, que también usaban CPU Motorola. A diferencia de otras computadoras personales de la época, estas máquinas eran mucho más potentes (de ahí la designación "estación de trabajo") y estaban diseñadas desde un principio para operar en una red. Cada estación de trabajo Sun venía equipada con una conexión Ethernet y con software TCP/IP para conectarse con ARPANET, la precursora de Internet.

Para 1987 Sun, que ya estaba vendiendo 500 millones de dólares al año en sistemas, decidió diseñar su propia CPU basándola en un nuevo y revolucionario diseño de la University of California at Berkeley (el RISC II). Esta CPU, llamada **SPARC** (**arquitectura de procesador escalable**, *Scalable Processor ARChitecture*), fue la base para la estación de trabajo Sun-4. En poco tiempo, todos los productos de Sun utilizaban la CPU SPARC.

A diferencia de muchas otras compañías de computadoras, Sun decidió no fabricar el chip de CPU SPARC ella misma. En vez de ello, otorgó licencias a varios fabricantes de semiconductores para producirlo, esperando que la competencia entre ellos hiciera subir el

desempeño y bajar los precios. Los proveedores fabricaron varios chips distintos, basados en diferentes tecnologías, que operaban a diferente velocidad de reloj, y con distintos precios. Estos chips incluyeron MicroSPARC, HyperSPARC, SuperSPARC y TurboSPARC. Aunque estas CPU diferían en aspectos menores, todas eran compatibles en lo binario y ejecutaban los mismos programas de usuario sin modificación.

Sun siempre quiso que SPARC fuera una arquitectura abierta, con muchos proveedores de componentes y sistemas, a fin de forjar una industria capaz de competir en un mundo de computadoras personales ya dominado por las CPU basadas en Intel. Para ganarse la confianza de compañías que estaban interesadas en SPARC pero no querían invertir en un producto controlado por un competidor, Sun creó un consorcio de la industria, SPARC International, para controlar el desarrollo de versiones futuras de la arquitectura SPARC. Por ello es importante distinguir entre la arquitectura SPARC, que es una especificación del conjunto de instrucciones y otras características visibles para el programador, y una implementación específica de ella. En este libro estudiaremos la arquitectura SPARC genérica y, al hablar de los chips de CPU en los capítulos 3 y 4, de un chip SPARC específico empleado en estaciones de trabajo Sun.

El SPARC inicial era una máquina completa de 32 bits que operaba a 36 MHz. La CPU, llamada **IU** (**unidad de enteros**, *Integer Unit*), era magra y feroz, con sólo tres formatos de instrucción principales y sólo 55 instrucciones en total. Además, una unidad de punto flotante aportaba otras 14 instrucciones. Esta historia puede contrastarse con la línea Intel, que comenzó con chips de 8 y 16 bits (8088, 8086, 80286) y por fin se convirtió en un chip de 32 bits con el 80386.

La primera ruptura de SPARC con el pasado ocurrió en 1995, con el desarrollo de la versión 9 de la arquitectura SPARC, una arquitectura cabal de 64 bits, con direcciones de 64 bits y registros de 64 bits. La primera estación de trabajo Sun que implementó la arquitectura V9 (versión 9) fue la **UltraSPARC I**, introducida en 1995 (Tremblay y O'Connor, 1996). A pesar de ser una máquina de 64 bits, era totalmente compatible en lo binario con los SPARC de 32 bits existentes.

El propósito de UltraSPARC era explorar nuevos terrenos. Mientras que máquinas anteriores estaban diseñadas para manejar datos alfanuméricos, ejecutar programas como procesadores de textos y hojas de cálculo, la UltraSPARC se diseñó desde el principio para manejar imágenes, audio, video y multimedia en general. Entre otras innovaciones además de la arquitectura de 64 bits había 23 instrucciones nuevas, incluidas algunas para empacar y desempacar pixeles de palabras de 64 bits, cambiar la escala de imágenes y girarlas, transferir bloques y realizar compresión y descompresión de video en tiempo real. Estas instrucciones, llamadas **VIS** (**conjunto de instrucciones visuales**, *Visual Instruction Set*) tenían por objeto ofrecer una capacidad multimedia general, análoga a las instrucciones MMX de Intel.

El UltraSPARC estaba dirigido a aplicaciones del extremo superior, como servidores de Web multiprocesador grandes, con docenas de CPU y memorias físicas de hasta 2 TB [1 TB (terabyte) = 10^{12} bytes]. Sin embargo, versiones más pequeñas pueden usarse también en computadoras portátiles.

Los sucesores del UltraSPARC I fueron UltraSPARC II y UltraSPARC III. Estos modelos difieren primordialmente en cuanto a velocidad de reloj, pero también se añadieron algu-

nas funciones nuevas en cada iteración. En este libro, cuando tratemos la arquitectura SPARC, utilizaremos el UltraSPARC II V9 de 64 bits como ejemplo.

1.4.3 Introducción al picoJava II

El lenguaje de programación C fue inventado por Dennis Ritchie de Bell Labs para usarlo con el sistema operativo UNIX. Gracias a su diseño económico y la popularidad de UNIX, C pronto se convirtió en el lenguaje de programación dominante en el mundo para programación de sistemas. Algunos años después, Bjarne Stroustrup, también de Bell Labs, añadió a C ideas del mundo de la programación orientada a objetos para crear C++, que también se volvió muy popular.

A mediados de los noventa, investigadores de Sun Microsystems estaban explorando formas de lograr que los usuarios obtuvieran programas binarios por Internet y los ejecutaran como parte de páginas de la World Wide Web. A ellos les gustaba C++, excepto que no era seguro. En otras palabras, un programa binario en C++ recién bajado de Internet fácilmente podía espiar la máquina que lo había adquirido e interferirla de otras maneras. Su solución a este problema fue inventar un nuevo lenguaje de programación, Java, inspirado en C++ pero sin los problemas de seguridad de éste. Java es un lenguaje orientado a objetos seguro, que cada vez se usa más para muchas aplicaciones. Por ser un lenguaje popular y elegante, lo usaremos en los ejemplos de programación de este libro.

Puesto que Java no es más que un lenguaje de programación, es posible escribir compiladores de él para la arquitectura Pentium, SPARC o cualquier otra. Existen tales compiladores, pero el principal objetivo de Sun al introducir Java era poder intercambiar programas ejecutables Java entre computadoras de Internet y ejecutarlos sin modificación. Si un programa Java compilado en una máquina SPARC se enviara por Internet a una Pentium, no se ejecutaría, y no se alcanzaría la meta de poder enviar un programa binario a cualquier sitio y ejecutarlo ahí.

Para poder transportar programas binarios entre máquinas distintas, Sun definió una arquitectura de máquina virtual llamada **JVM (Java Virtual Machine)**. Esta máquina tiene una memoria formada por palabras de 32 bits, y puede ejecutar 226 instrucciones. Casi todas estas instrucciones son sencillas, pero unas cuantas son muy complejas y requieren varios ciclos de memoria.

A fin de hacer portátil a Java, Sun escribió un compilador de Java para la JVM, y también escribió un intérprete JVM para ejecutar programas binarios Java. Éste intérprete se escribió en C y por tanto puede compilarse y ejecutarse en cualquier máquina que posea un compilador de C, lo que en la práctica incluye a casi todas las máquinas del mundo. Por consiguiente, lo único que el dueño de una máquina necesita hacer para que ésta pueda ejecutar programas binarios Java es conseguir el programa binario ejecutable del intérprete de Java para esa plataforma (digamos, Pentium II y Windows 98, SPARC y UNIX, etc.) junto con ciertos programas y bibliotecas de apoyo asociados. Además, casi todos los navegadores de Internet contienen un intérprete de JVM para poder ejecutar fácilmente **applets**, que son pequeños programas binarios Java asociados a páginas de la World Wide Web. Muchos de estos applets proporcionan animación y sonido.

La interpretación de programas para JVM (o de cualesquier otros programas, para el caso) es lenta. Una estrategia alternativa a ejecutar un applet u otro programa JVM recién recibido es compilarlo primero para la máquina en cuestión y luego ejecutar el programa compilado. Esta estrategia requiere tener un compilador de JVM a lenguaje de máquina dentro del navegador y poder activarlo automáticamente cuando se necesite. Tales compiladores, llamados compiladores **JIT (justo a tiempo**, *Just In Time*), existen y son comunes, pero son grandes e introducen un retraso entre la llegada del programa JVM y su ejecución mientras el programa se compila a lenguaje de máquina.

Además de las implementaciones en software de la máquina virtual Java (intérpretes de JVM y compiladores JIT), Sun y otras compañías han diseñado chips de JVM. Éstos son diseños de CPU que ejecutan directamente programas binarios JVM, sin necesidad de una capa de interpretación por software o compilación JIT. Las arquitecturas iniciales, picoJava I (O'Connor y Tremblay, 1997) y picoJava II (McGhan y O'Connor, 1998), estaban pensadas para el mercado de los sistemas incorporados. Este mercado requiere chips potentes, flexibles y sobre todo de bajo costo (menos de 50 dólares, y a veces mucho menos) que se incorporan en tarjetas inteligentes, televisores, teléfonos y otros aparatos, sobre todo los que necesitan comunicarse con el mundo exterior. Los licenciatarios de Sun pueden fabricar sus propios chips utilizando el diseño picoJava, adaptándolos hasta cierto punto incluyendo o quitando la unidad de punto flotante, ajustando el tamaño de los cachés, etcétera.

El valor de un chip Java para el mercado de los sistemas incorporados es que un dispositivo puede alterar su funcionalidad mientras está operando. Por ejemplo, considere un ejecutivo de negocios que tiene un teléfono celular basado en Java y que nunca vislumbró la necesidad de leer faxes en la diminuta pantalla del teléfono, pero que repentinamente necesita hacerlo. Si llama a su proveedor celular, el ejecutivo puede bajar a su teléfono un applet para visualización de faxes y añadir esta funcionalidad al dispositivo. Las necesidades de velocidad excluyen la posibilidad de interpretar los applets Java, y la falta de memoria en el teléfono impide la compilación JIT. Ésta es una situación en la que un chip JVM es útil.

Aunque el picoJava II no es un chip concreto (no puede ir a una tienda y comprar uno), es la base de varios chips, como la CPU microJava 701 de Sun y muchos más de otros licenciatarios de Sun. Utilizaremos el diseño picoJava II como uno de nuestros ejemplos constantes en todo el libro porque es muy diferente de las CPU Pentium II y UltraSPARC, y está dirigido a un área de aplicación muy diferente. Esta CPU resulta especialmente interesante para nuestros propósitos porque en el capítulo 4 presentaremos un diseño para implementar un subconjunto de JVM empleando microprogramación. Luego podremos contrastar nuestro diseño microprogramado con un verdadero diseño en hardware.

El picoJava II tiene dos unidades opcionales: un caché y una unidad de punto flotante, que cada fabricante de chips puede incluir o eliminar, a voluntad. Por sencillez, nos referiremos al picoJava II como si fuera un chip en lugar de un diseño de chip. Cuando sea importante (en contadas ocasiones), seremos específicos y hablaremos del chip microJava 701 que implementa el diseño picoJava II. Aun cuando no mencionemos el microJava 701 específicamente, el material aplicará a él y a todos los demás chips Java de otros fabricantes basados en este diseño.