第 1 章　计算机体系结构的基本概念

1.1　计算机体系结构的概念

在目前得到大多数人认可的历史中，人们最早使用的电子数字计算机（通常称之为计算机）是 1945 年诞生于美国宾夕法尼亚大学的 ENIAC，用于计算火炮的弹道。1946 年，生于匈牙利的美国数学家冯•诺依曼（John von Neumann，1903 年 12 月 28 日—1957 年2月8日）等人在总结当时计算机研究成果的基础上，明确提出了存储程序计算机（stored-program computer），因此又称存储程序计算机为冯•诺依曼结构计算机，它奠定了现代电子数字计算机的结构基础，成为计算机的标准结构。六十多年来，虽然计算机技术一直处于飞速发展和变革之中，但是存储程序计算机的概念和基本结构一直沿用至今，没有发生根本性的变化，是计算机体系结构研究的基础。

1.1.1　存储程序计算机

存储程序计算机是一种计算机系统设计模型，实现了一种通用图灵机（Universal Turing Machine）。冯•诺依曼描述的计算机由以下 4 个部分组成：

（1）运算器。用于完成数值运算。

（2）存储器。用于存储数据和程序。

（3）输入输出设备。用于完成计算机和外部的信息交换。

（4）控制器。根据程序形成控制（指令、命令）序列，完成对数据的运算。

控制器根据程序指令序列，分解形成对计算机 4 个部分操作的控制信号序列，称为控制流。例如，从存储器某个特定单元取一条指令，把运算器的计算结果送到存储器某个特定单元，启动运算器完成加法运算，等等。计算机在控制流的操作下，计算机 4 个部分之间形成数据和指令的传送序列，称为数据/指令流。存储程序计算机 4 个部分的结构如图 1.1 所示，为了更加直观，图中将输入设备和输出设备分成了两个独立的部件。

存储程序计算机在体系结构上的主要特点如下：

（1）机器以运算器为中心。输入输出设备与存储器之间的数据传送都经过运算器；存储器、输入输出设备的操作以及它们之间的联系都由控制器集中控制。

（2）采用存储程序原理。程序（指令）和数据放在同一存储器中，且没有对两者加以区分。指令和数据一样可以送到运算器进行运算，即由指令组成的程序自身是可以修改的。

（3）存储器是按地址访问的、线性编址的空间。每个单元的位数是相同且固定的，称为存储器编址单位。

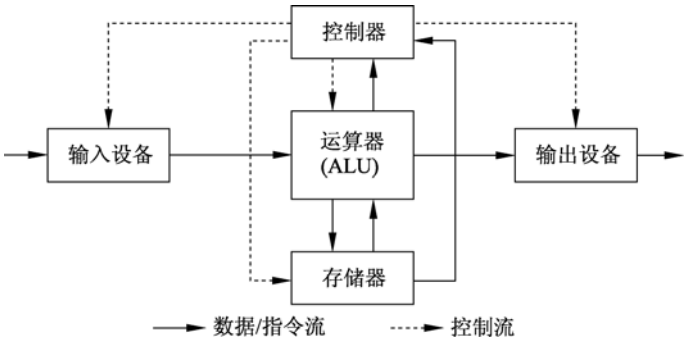

图 1.1　存储程序计算机的结构

（4）控制流由指令流产生。指令在存储器中是按其执行顺序存储，由指令计数器指明每条指令所在单元的地址。每执行完一条指令，程序计数器一般是顺序递增。虽然执行顺序可以根据运算结果改变，但是解题算法仍然是也只能是顺序型的。

（5）指令由操作码和地址码组成。操作码指明本指令的操作类型，地址码指明操作数和操作结果的地址。操作数的类型（定点、浮点或十进制数）由操作码决定，操作数本身不能判定是何种数据类型。

（6）数据以二进制编码表示，采用二进制运算。它主要面向数值计算和数据处理。

存储程序计算机体系结构的这些特点奠定了现代计算机发展的基础。

在存储程序计算机中，程序执行的过程就是对程序指令进行分解，形成控制计算机 4 个部分工作的控制流，对数据进行加工（运算），周而复始地产生数据/指令流，并最终得到数据结果的过程。计算机对每条指令从取指令到得到结果的工作周期，称为一个机器周期。机器周期可以按照设计者的思路，进一步分解为一系列操作，例如分解为两个部分：指令周期和数据周期；还可以按照访问存储器分解为 3 个部分：从存储器取指令、从存储器取操作数和将计算结果写回到存储器。在本教材中，将一条指令的操作分解为 5 个部分：取指令、指令译码、指令执行、访问存储部件和写结果，如图 1.2 所示。

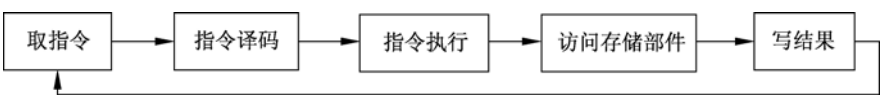

图 1.2　分解为 5 个部分的存储程序计算机的机器周期

在图 1.2 所示的机器周期中，第一，计算机从存储器中取出一条指令；第二，对这条指令进行译码，由硬件分解并确定这条指令所指示的操作，同时确定该指令操作对象（操作数）所在的位置，这个位置可以是寄存器单元、存储器的特定单元或者某个输入设备；第三，根据译码确定的位置去取操作数并送到运算器；第四，运算器按照译码确定的操作进行运算；第五，运算结束后，将结果送到指定的位置，这个位置可以是寄存器单元、存储器的特定单元或者某个输出设备，至此计算机就将一条指令执行完了，并准备执行下一条指令。这就是对一个存储程序计算机的处理器在一个机器周期里面安排的操作序列。

1.1.2 计算机体系结构、组成和实现

虽然计算机体系结构（computer architecture）的概念目前已被广泛使用，但是这个概念所包含的研究内容一直在发生变化。Architecture 本来用在建筑方面，译为“建筑学、建筑术、建筑样式”等。这个词被引入计算机领域后，最初的译法也各有不同，其包含的主要内容有“系统”、“体系”、“方法”、“结构”、“组织”等，以后趋向译为“体系结构”或“系统结构”。

1964 年 4 月，阿姆道尔（C. M. Amdahl）等在 IBM Journal of Research and Development 杂志上，发表了题为 Architecture of the IBM System/360 的文章，首次明确提出了“计算机体系结构”的概念，并且将计算机体系结构定义为：计算机体系结构是程序员所看到的计算机的属性，即概念性结构与功能特性。这就是计算机体系结构概念的经典定义。

程序员所看到的计算机的属性，是指程序（机器语言、汇编语言或者编译程序生成系统）设计者为使其所设计（或生成）的程序能在机器上正确运行，必须掌握和遵循的计算机属性，这些属性包含其概念性结构和功能特性两个方面。目前，对于通用寄存器型机器，这些属性主要是指：

（1）数据表示。硬件能直接辨认和处理的数据类型。

（2）寻址规则。包括最小寻址单元、寻址方式及其表示。

（3）寄存器定义。包括各种寄存器的定义、数量和使用方式。

（4）指令系统。包括机器指令的操作类型和格式、指令间的排序和控制机构等。

（5）中断系统。中断的类型和中断响应硬件的功能等。

（6）机器工作状态的定义和切换。如管态和目态等。

（7）存储系统。主存容量、程序员可用的最大存储容量等。

（8）信息保护。包括信息保护方式和硬件对信息保护的支持。

（9）I/O 结构。包括 I/O 连接方式、处理机/存储器与 I/O 设备间数据传送的方式和格式，以及 I/O 操作的状态等。

这些属性是计算机系统中由硬件或固件完成的功能，程序员在了解这些属性后才能编出在计算机上正确运行的程序。因此，经典计算机体系结构概念的实质是计算机系统中软硬件界面的确定，也就是指令系统的设计。在体系结构界面之上，更多是软件研究领域关注的功能和内容，体系结构界面之下的是硬件和固件等实现方面研究的功能和内容。

随着计算机技术的发展，计算机体系结构所研究的内容不断发生变化和发展。表 1.1 给出了从电子数字计算机产生到现在计算机体系结构研究关注的一些典型内容。

阿姆道尔提出计算机体系结构的概念是在 20 世纪 60 年代，当时指令系统的研究和设计正处在发展时期，是设计计算机系统时必须研究、权衡、取舍的重要内容，影响系统设计目标的实现和成本，所以他们定义的计算机体系结构概念更多地体现出当时的时代特点。

表 1.1　不同年代计算机体系结构研究的一些内容

年　　代	一些重要研究内容	典型计算机（处理器）实例
20 世纪 40 年代	程序控制计算机和存储程序计算机	ENIAC、EDVAC
20 世纪 50—60 年代	指令系统	IBM 360 系列机
20 世纪 60 年代	阵列机和并行处理	ILLIAC IV
20 世纪 70 年代	流水线、向量处理、微处理器	Cray-1、Intel 4004
20 世纪 80 年代	RISC、Cache、流水线	MIPS R1000、Power
20 世纪 90 年代	指令级并行	MIPS R10000、PowerPC 604
2000 年以来	SMP、CMP、SMT、功耗	Intel i7、Power 6、ARM

目前，关于体系结构定义的细节仍有多种不同的说法，其中以斯坦福大学的 J. L. Hennessy 和加州大学伯克利分校的 D.A.Patterson 在他们编写的经典教材《Computer Architecture：A Quantitative Approach》中给出的定义最为普遍认可。他们认为，计算机体系结构包括计算机系统设计的 3 个方面：计算机指令系统、计算机组成和计算机硬件。

（1）计算机指令系统：程序员可见的实际指令系统，是计算机系统硬件和软件之间的一个分界面。例如，MIPS 系列微处理器和 80x86 系列微处理器是两个完全不同的指令系统，这两个系列微处理器分别是 RISC 和 CISC 体系结构。

（2）计算机组成（computer organization）：又称微体系结构（microarchitecture），是计算机系统中各个功能部件及其连接的设计。例如，Intel 的 Nehalem 和 AMD 的 Phenom 拥有相同的 x86 指令系统，但是处理器的组成不同，包括流水线、多核互连、Cache 结构等。

（3）计算机硬件：是指计算机具体的实现（implementation）技术，包括逻辑设计、集成电路工艺、封装等。同一个系列的计算机一般具有相同的指令系统和组成，但是硬件上存在差别。例如，Intel 的 Nehalem 系列中 i7、i5 和 i3 几乎一样，但是工艺、封装、Cache 容量、主频等不相同。

在维基百科（www.wikipedia.org）英文版中，对计算机体系结构的描述有 3 个部分。第一部分，体系结构是计算机系统的概念设计和基础结构设计，是计算机中各个部件的总体框架描述、需求功能描述和设计实现，体系结构特别关注于 CPU 运行和存储器访问。第二部分，体系结构还是一种对硬件部件进行选择和连接的学问，其研究目标是构造一个满足功能、性能和价格目标的计算机。第三部分，体系结构最少包括以下 3 个方面的内容：

（1）指令系统。这是机器语言程序员看见的计算机系统的一个抽象，包括指令、字宽、寻址方式、寄存器、地址和数据类型等。

（2）微体系结构。又称计算机组成，是对计算机系统更加具体和细化的描述，包括系统中各部件的连接、如何协作来完成指令系统的功能。

（3）系统设计。包括计算机系统中的其他硬件部件设计问题，如总线和开关等系统互连方式，存储控制器和存储层次，包括 DMA 在内的 CPU 和各种部件之间的协同机制，

多处理技术等。

维基百科对计算机硬件实现技术进一步分解为以下 3 个不同层次，并认为这些层次之间的界限也不是非常清晰：

（1）逻辑实现。这是一种框图级的设计，通常在寄存器传输级（register-transfer level，RTL）和门级来描述计算机的微体系结构，例如门级的逻辑图。

（2）电路实现。主要集中在基本部件的晶体管级设计，包括门、多路选择器、门闩寄存器等，有些较大的功能模块，例如 ALU、Cache 等，为了性能优化会进行跨级设计，从逻辑实现级延伸到电路实现级甚至物理实现级。

（3）物理实现。完成物理电路的实现，包括所有电路器件在芯片版图或者印刷电路板（printed circuit board，PCB）上的布局和布线。

可以看出，计算机体系结构概念的核心内容具有一定的稳定性，包括指令系统设计、CPU 设计、计算机系统设计和硬件实现等，但是它的研究重点和范围随着基础科学和技术的发展而变化，具有不确定性和与时俱进的特点。

在本教材中，体系结构的概念用于描述计算机系统设计的技术、方法和理论，主要包括计算机指令系统、计算机组成和计算机硬件实现 3 个方面，涵盖处理器和多处理器、存储器、输入输出系统、互连与通信等计算机系统设计的主要内容，同时还涉及性能评价、编译和操作系统的关键技术，并通过定量分析的途径，学习掌握现代计算机体系结构研究的基本方法。

1.1.3 计算机系统中的层次概念

现代计算机系统是由软件和硬/固件组成的十分复杂的系统，图 1.3 从计算机语言的角度，把计算机系统按功能划分成多级层次结构。

图 1.3 计算机系统的多级层次结构

计算机语言可分成一系列的层次（level）或级，最低层语言的功能最简单，最高层语言的功能最强。对于用某一层语言编写程序的程序员来说，无论程序在机器中是如何执行的，只要程序正确，最终能得到预期的结果，似乎有了一种新的机器，这层语言就是这种机器的机器语言，该机器能执行相应层语言写的全部程序。因此，计算机系统就可以按语言的功能划分成多级层次结构。每一层以一种不同的语言为特征。这样，可以把现代计算机系统画成如图 1.3 所示的层次结构。图中第 1 级和第 2 级由硬件或固件实现，第 4 级以上基本由软件实现。由软件实现的机器为虚拟机器（virtual machine），以

区别于由硬件或固件实现的实际机器。

每个计算机语言的层次结构（每种虚拟机）包括 3 个部分：语言、执行机制和程序。图 1.4 定义了一个计算机（虚拟机）的体系结构。

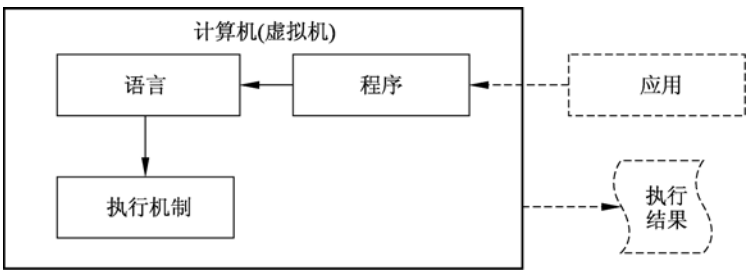

图 1.4　从计算机语言角度看计算机（虚拟机）的体系结构

（1）语言：命令的集合，计算机（虚拟机）的指令系统。

（2）执行机制：按照语言的定义执行程序，计算机（虚拟机）的具体实现。

（3）程序：描述应用的语言序列。

按照计算机语言看计算机（虚拟机）的角度，对图 1.3 中计算机系统的多级层次结构的每一个层次都可以进行分解。

第 1 级是微程序机器级。这级的机器语言是微指令集，其程序员实际上是计算机系统的设计人员，他们用微程序描述的实际上是计算机指令集中每一条指令的功能。用微指令编写的微程序一般是直接由硬件解释实现的。

第 2 级是传统机器级。这级的机器语言是该机的指令集，程序员用机器指令集编写的程序可以由微程序进行解释。这个解释程序运行在第 1 级上。由微程序解释指令集又称做仿真（emulation）。实际上，在第 1 级可以有一个或数个能够在它上面运行的解释程序，每一个解释程序都定义了一种指令集。因此，可以通过仿真在一台机器上实现多种指令集。

计算机系统中也可以没有微程序机器级。在这些计算机系统中是用硬件直接实现传统机器的指令集，而不必由任何解释程序进行干预。目前广泛使用的 RISC 技术就是采用这样的设计思想，处理器的指令集用硬件直接实现，目的是加快指令执行的速度。

第 3 级是操作系统虚拟机。从操作系统的基本功能来看，一方面它要直接管理传统机器中的软硬件资源，另一方面它又是传统机器的引申。它提供了传统机器所没有的某些基本操作和数据结构，如文件结构与文件管理的基本操作、存储体系和多道程序以及多重处理所用的某些操作、设备管理等，这些基本操作和数据结构往往和指令集一起，以整体形式提供给更高层次的虚拟机使用，称之为系统功能调用和系统参数。

第 4 级是汇编语言虚拟机。这级的机器语言是汇编语言，用汇编语言编写的程序，首先翻译成第 3 级和第 2 级语言，然后再由相应的机器执行。完成汇编语言翻译的程序就叫做汇编程序。

在第 4 级上出现了一个重要变化。通常的第 1、2 和 3 级是用解释（interpretation）

方法实现的，而第 4 级或更高级则经常使用翻译（translation）方法实现。

翻译和解释是语言实现的两种基本技术。它们都是以执行一串 *N* 级指令来实现 *N*+1 级指令，但二者仍存在着差别：翻译技术是先把 *N*+1 级程序全部变换成 *N* 级程序后，再去执行新产生的 *N* 级程序，在执行过程中，*N*+1 级程序不再被访问。而解释技术是每当一条 *N*+1 级指令被译码后，就直接去执行一串等效的 *N* 级指令，然后再去取下一条 *N*+1 级的指令，依此重复进行。在这个过程中不产生翻译出来的程序，因此解释过程是边变换边执行的过程。在实现新的虚拟机器时，这两种技术都被广泛使用。一般来说，解释执行比翻译花的时间多，但存储空间占用较少。

第 5 级是高级语言虚拟机。这级的机器语言就是各种高级语言，目前高级语言已达数百种。高级语言的翻译过程就是编译（compile），完成翻译的程序就是编译器（compiler）或编译程序。用高级语言，如 C/C++、Pascal、FORTRAN 等所编写的程序一般是由编译器翻译到第 4 级或第 3 级上。个别的高级语言也用解释的方法实现，如绝大多数 BASIC 语言系统。

第 6 级是应用语言虚拟机。这一级是为使计算机满足某种用途而专门设计的，因此这一级语言就是各种面向问题的应用语言。可以设计专门用于人工智能、教育、行政管理、计算机设计、网络应用开发等方面的虚拟机，如 Lisp、SQL、Perl、Python，这些虚拟机也是当代计算机应用领域的重要研究课题。应用语言编写的程序一般是由应用程序包翻译到第 5 级上，也有越级翻译到更低一些级其他机器上的。

按照计算机系统的多级层次结构，不同级程序员所看到的计算机具有不同的属性。例如，机器（汇编）语言程序员所看到计算机的主要属性是该机指令集的功能特性；而高级语言虚拟机程序员所看到计算机的主要属性是该机所配置的高级语言所具有的功能特性。显然，不同的计算机系统，从机器语言程序员或汇编语言程序员看，是具有不同的属性的。但是，从高级语言（如 FORTRAN、C/C++、Java、Python）程序员看，它们就几乎没有什么差别，具有相同的属性。或者说，这些传统机器级所存在的差别是高级语言程序员所“看不见”的，或者说是不需要他们知道的。在计算机技术中，对这种本来是存在的事物或属性，但从某种角度看又好像不存在的概念称为透明性（transparency）。通常，在一个计算机系统中，低层机器的属性往往对高层机器的程序员是透明的，如传统机器级的概念性结构和功能特性，对高级语言程序员来说是透明的，也就是说，FORTRAN 程序员是难以知道计算机中是否使用了 Cache，使用了几级 Cache，Cache 的容量是多大，从而 Cache 对他们（FORTRAN 程序员）是透明的。由此看出，在层次结构的各个级上都有它的体系结构。阿姆道尔提出的经典体系结构是指机器语言级程序员所看见的计算机属性。

1.1.4 系列机和兼容

系列机（family machine）是具有相同体系结构，但组成和实现不同的一系列不同型号的计算机系统。IBM 公司在推出 IBM S360 时首次提出系列机的概念，被认为是计算机发展史上的一个重要里程碑。至今，各计算机厂家仍按系列机研发产品。现代计算机

不但系统系列化，其构成部件和软件也系列化，如微处理器（CPU）、硬盘、操作系统、高级语言等。目前计算机领域中产量最大的系列计算机莫过于 IBM PC 及其兼容系列个人计算机（personal computer，PC）和 Intel 的 x86 系列微处理器。

PC 系列计算机则从 1981 年开始至今，历史悠久、厂家众多、产量庞大、发展的技术路线各异、品种复杂，其中最基本的分类是根据所采用处理器的类型来进行的。表 1.2 列出了处理器、处理器字宽、I/O 总线和主要操作系统的比较。

表 1.2　PC 系列机典型特性比较

计算机	时间	处理器	字宽 位	主要 I/O 总线	主要操作系统
PC 和 PC XT	1981	8088	16	PC 总线	DOS
PC AT	1982	80286	16	AT（ISA）	DOS、XENIX
80386 PC	1985	80386	32	ISA/EISA	DOS、Windows 3.0
80486 PC	1989	80486	32	ISA+VL	DOS、Windows 3.1
Pentium PC	1993	Pentium	32	ISA+PCI	DOS、Windows 3.1
Pentium Ⅱ PC	1997	Pentium Ⅱ	32	ISA+PCI+AGP	Windows 95
Pentium Ⅲ PC	1999	Pentium Ⅲ	32	PCI+AGP +USB	Windows 98、2000
Pentium 4 PC	2000	Pentium 4	32	PCI-X+AGP +USB	Windows Me、XP
Core PC	2006	Core Solo	32/64	PCI-E+AGP +USB	Windows XP、Vista
Core PC	2010	Core i7/i5/i3	32/64	PCI-E+AGP +USB	Windows XP、7

从上述系列机的例子可见：一种体系结构可以有多种组织、多种物理实现。系列机具有相同的体系结构，软件可以在系列计算机的各档机器上运行，人们称这种情况下的各档机器是软件兼容的（software compatibility），即同一个软件可以不加修改地运行于体系结构相同的各档机器上，而且它们所获得的结果一样，差别只在于有不同的运行时间。系列机从程序设计者看都具有相同的机器属性（体系结构），因此按这个属性实现的编译器和编制的程序能通用于各档机器，这种软件兼容的概念是比较宽泛的，还可以进一步分为二进制级兼容、汇编级兼容、高级语言兼容和数据级兼容等，这里不深入探讨。

长期以来，程序员希望有一个稳定的软件环境，使编制出来的程序能够在更加广泛的计算机类型中得到长期的应用，而机器设计人员则希望根据硬件技术和器件技术的进展不断地推出新的机器，系列机的出现较好地解决了软件要求环境稳定和硬件、器件技术迅速发展之间的矛盾，对计算机技术的发展起到了重要的推动作用。计算机厂家为了减少市场风险而大量研制兼容产品。不同厂家生产的具有相同体系结构的计算机称为兼容机（compatible machine）。早在 20 世纪 60 年代，就出现了以 Amdahl 公司为代表的接插兼容机（plug-compatible mainframe，PCM）厂家，专门生产在功能上和电气性能上与 IBM 公司相同的主机、配件和设备，它不但可以运行 IBM 公司的软件，而且又可以作为 IBM 产品的替换件插入 IBM 系统。由于采用了新的硬件和器件技术，也改善了性能价格比，因而成为 IBM 公司的强有力的竞争对手，这种竞争有力地推动了计算机技术和应用

的发展。兼容机的出现，极大地推动了计算机各种部件标准的规范化，例如 ISA/EISA、VISA、AGP、MCA、PCI、PCI-E 等系统级总线标准，FPRAM、EDO RAM、RDRAM、WRAM、VRAM、SDRAM（包括 SDR、DDR、DDR2 和 DDR3）等存储器接口标准，RS232、1394、USB、e-ATA 等设备接口标准，这些标准的产生和发展，极大地推进了计算机产品标准化的进程，降低了生产和制造成本，对计算机的普及发挥了根本作用。目前这种技术已经成为计算机系统发展的重要技术之一，有一大批厂家专门生产符合各种标准的设备，这些设备可以在拥有标准接口的各种计算机上使用，大厂商也从这些厂家订购配件用于自己的计算机系统，甚至订购整机贴上自己的商标销售。这些厂商一般称为独立设备生产厂商或者第三方生产厂商。目前市场上比较流行的组装机器也叫兼容机，就是使用各种标准配件，组装成 PC。这种情况又称为硬件兼容（hardware compatibility），即在某些 CPU、总线、主板和操作系统中，计算机硬件部件具有兼容性，各种软件可以在这些兼容的硬件部件下正确运行。制定规范标准的同时也产生了各种标准之间的竞争，因为拥有规范或标准的公司在产品竞争中会带来显著的经济优势。

系列机为了保证软件的兼容，要求体系结构不能改变，这种约束无疑又妨碍了计算机体系结构的发展。实际上，系列机的软件兼容还有向上兼容、向下兼容、向前兼容和向后兼容之分。向上（下）兼容指的是按某档机器编制的程序，不加修改就能运行于比它高（低）档的机器上。向前（后）兼容指的是按某个时期投入市场的某种型号机器编制的程序，不加修改就能运行于在它之前（后）投入市场的机器。图 1.5 说明了这些概念。

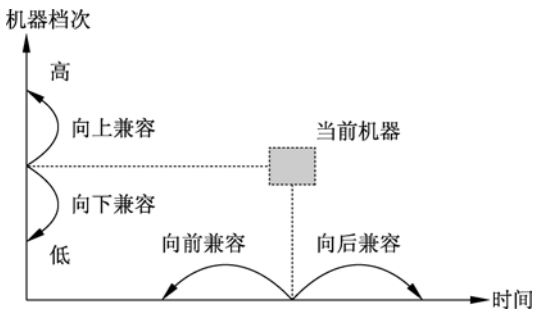

图 1.5　兼容性示意图

为了适应系列机中性能不断提高和应用领域不断扩大的要求，后续各档机器的体系结构也是可以改变的。如增加浮点运算指令以提高速度，或者增加事务处理指令以满足事务处理方面的需要，增加多媒体指令来强化信号处理能力等。但是，这种改变一定是原有体系结构的扩充，而不是任意的更改或缩小。这样，对系列机的软件向下和向前兼容可以不作要求，向上兼容在某种情况下也可能做不到（如在低档机器上增加了面向事务处理的指令），但向后兼容却是肯定要做到的，可以说向后兼容才是软件兼容的根本特征，也是系列机的根本特征。一个系列机的体系结构设计的好坏，是否有生命力，就看是否能在保证向后兼容的前提下，不断地改进其组成和实现。Intel 公司的 x86 系列微处理器在向后兼容方面是非常具有代表性的，从 1979 年的 8086 到 2009 年的 Nehalem（包括 i7、i5 和 i3 共 3 个系列微处理器），由 16 位系统发展到 64 位系统，增加了保护方式指令、多媒体指令（MMX 扩展）和 64 位指令扩展等，但它保持了极好的二进制代码级的向后兼容性。向后兼容虽然削弱了系列机对体系结构发展的约束，但仍然是计算机体系结构发展的一个沉重包袱，例如，Intel 的新一代 64 位 Iantium 体系结构，由于没有解决好对 x86 体系结构的兼容，执行 32 位软件的效率较低，一直不能很好地被市场接受。在 20 世纪 80 年代，

具有 RISC 体系结构的微处理器没有历史的包袱，可以大量应用新结构、新技术，使得 RISC 微处理器在技术和市场两个方面都领先传统的 CISC 微处理器。

1.2 计算机体系结构的发展

一种成功的指令系统（instruction set architecture，ISA，又称为指令集结构）必须能够适应硬件技术、软件技术及应用特性的变化。设计者尤其要注意计算机应用领域、使用方法及实现技术的变化和发展趋势，这样，一个新型指令集结构才可能会有数十年的使用寿命，如 IBM AS/400 计算机指令系统的核心技术从 1964 年至今一直在使用，而现今广泛使用的 x86 指令集来源于 Intel 公司 1974 年正式上市的 8 位微处理器芯片 8080。为延长采用某种体系结构的计算机使用寿命，该体系结构必须能够适应技术的变化。计算机的设计受两方面因素的影响：一方面是计算机现在和未来的使用方法（软件技术），另一方面是实现技术。体系结构的发展伴随软件、应用和实现技术的发展而变化。

1.2.1 计算机分代、分型与分类

一般认为计算机到目前为止已经发展了 5 代。这 5 代计算机分别具有明显的器件、体系结构技术和软件技术的特征。表 1.3 列出了其中的典型特征。

表 1.3 5 代计算机的特征

分　代	器　件	体系结构技术	软件技术	国外，国内典型机器
第一代（1945—1954）	电子管和继电器	存储程序计算机、程序控制 I/O	机器语言和汇编语言	普林斯顿 ISA、ENIAC、IBM 701，331（901）*
第二代（1955—1964）	晶体管、磁芯、印刷电路	浮点数据表示、寻址技术、中断、I/O 处理机	高级语言和编译、批处理监控系统	Univac LARC、CDC 1604、IBM 7030，441B
第三代（1965—1974）	SSI 和 MSI、多层印刷电路、微程序	流水线、Cache、先行处理、系列计算机	多道程序和分时操作系统	IBM 360/370、CDC 6600/7600、DEC PDP-8，151-IV
第四代（1975—1990）	LSI 和 VLSI、半导体存储器	向量处理、分布式存储器	并行与分布处理	Cray-1 、IBM 3090 、DEC VAX9000，银河-I
第五代（1991—）	微处理器、高密度电路	指令级并行、SMP、MP、MPP、计算机网络	大规模、可扩展并行与分布处理	SGI Jaguar、IBM Roadrunner，天河 1 号

*国内典型机器均以国防科技大学（军事工程学院）研制的机器为例。

计算机曾经根据价格分为 5 个档次，包括巨型机（supercomputer）、大型机（mainframe）、中型机、小型机（minicomputer）和微型机（microcomputer）。

概括地分析计算机体系结构的进展，可以发现，计算机体系结构的成就不只是表现在巨、大型机上，而且在中、小、微型计算机中也愈来愈多地被采用，同时还包括技术和性能的“下移”。图 1.6 给出了计算机系统性能随时间“下移”的示意。

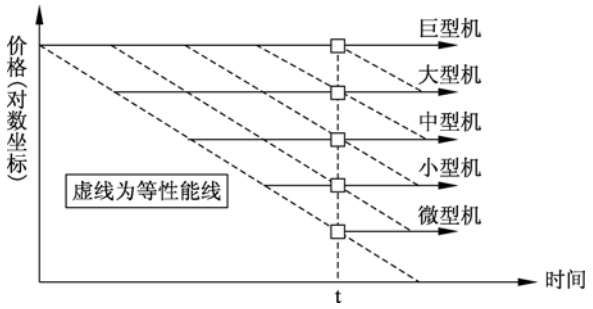

图 1.6　计算机性能下移示意图

20 世纪 60 年代出现在大型计算机上的流水线处理技术、Cache 技术、虚拟存储器技术、向量处理技术等，目前已经普遍使用在 PC 上。目前使用的 Intel i7 处理器，其处理能力达到数十亿次，达到 20 世纪 80 年代初期巨型机的性能，i7 中多 CPU 采用的 SMP 技术也是 20 世纪 80 年代甚至 90 年代高性能高档服务器才采用的技术。从计算机专业教材内容来看，关于流水线处理技术、Cache 技术、虚拟存储器技术、向量处理技术等内容，在 2000 年以前，是计算机体系结构课程讨论的，而目前有很多计算机组成原理教材就包含了这些内容。这就是计算机技术和性能的“下移”对高等学校的教材和教学带来的影响。

根据当前的计算机应用市场的现状和价格特征，通常把计算机分为服务器、桌面系统和嵌入式计算 3 大领域，后面会分析这 3 大领域的一些重要特征。巨型机目前还是一个非常活跃的领域，对整个计算机技术发展和市场有相当的影响，由于追求的指标和技术的特殊性，现在一般把巨型机看成一个以高性能计算为目标的服务器领域分支。

从上述体系结构进展的情况可以看出，随着器件提供的设计空间的增大，新型体系结构的设计，一方面是合理地增加计算机系统中硬件的功能比例，使体系结构对操作系统、高级语言甚至应用软件提供更多更好的支持；另一方面则是通过多种途径提高计算机体系结构中的并行性等级，使得凡是能并行计算和处理的问题都能并行计算和处理，使这种体系结构和组成对算法提供更多更好的支持。但要记住，有生命力的计算机系统所采用的无论是哪种措施，都要在满足应用要求的前提下，使系统具有合理的性能价格比。

1.2.2　软件的发展

软件技术最重要的发展趋势之一就是程序及数据所使用存储器容量的不断增大。程序所需的存储器容量平均每年递增 1.5%～2%，也就是说，计算机的地址位以每年 1/2～1 位的速度递增。如此高的增长速度主要由两方面原因造成：一方面是程序的需要，另一方面是 DRAM 技术的发展。DRAM 技术能够不断降低存储器的位成本。对地址空间增长速度估计不足，是一种指令集结构被淘汰的最主要原因。

1．计算机语言和编译技术

随着计算机系统和应用领域的发展，人们已设计出一系列的计算机语言。从面向机

器的语言（如机器语言、汇编语言），到各种高级程序设计语言（如 Java、C/C++、FORTRAN、Python），到各种面向问题的语言或者叫应用语言（如面向数据库查询的 SQL 语言、面向数字系统设计的 VHDL 语言、面向人工智能的 PROLOG 语言）。人们可以根据应用需求，选择或者设计符合自己要求的计算机语言。总体上看，计算机语言是由低级向高级发展的，高一级语言的语句相对于低级语言功能更强，更便于应用，但又都以低级语言为基础。

在过去 30 年里，软件技术的另一个重要发展趋势是标准的高级语言广泛使用，在很多应用领域取代了汇编语言，这种变化使编译器更加重要。只有编译器与计算机系统紧密联系，才能够生产出更具竞争力的机器。编译器已成为用户与计算机的主要界面，而 C/C++成为 UNIX（类）/Linux 操作系统“内嵌”的语言。

作为最主要的用户界面，编译技术正在逐渐增加新的功能，不断提高程序在机器上运行的效率。编译技术的改进既包括传统的优化技术，也包括为适应、完善流水线及存储系统而进行的改进。如何分配编译器与硬件所负担的工作，如何让编译器理解体系结构的变化的特点或优化的目标，使改进的处理器能够有效运行，一直是体系结构和编译两个领域中最热门的共同技术话题之一。所以，体系结构研究人员要对编译技术和编译工具非常熟悉。

2．操作系统

操作系统是计算机资源管理系统，包括 CPU 管理（进程管理）、存储管理和设备管理，同时提供用户界面（用户管理），对于今天的应用，特别是 Internet 等网络应用，如物联网（the Internet of things）以及目前被广泛关注的人机物网络（cyber physical space，CPS），安全管理成为操作系统关注的技术热点。与应用领域相对应，今天的操作系统包括嵌入式操作系统、桌面操作系统和服务器操作系统。

对操作系统的支持，一直是体系结构研究的重点领域，在计算机体系结构教材中，人们会发现大量的与操作系统有关的功能和内容。体系结构对操作系统提供哪些支持，往往是体系结构和操作系统研究人员共同关注和设计的。表 1.4 给出了操作系统对体系结构设计的一些常见要求。

表 1.4　操作系统与计算机系统设计

操作系统类型	当 前 实 例	对计算机系统设计的部分要求
嵌入式系统	WinCE、VxWorks、EPOS、Linux	实时性、低功耗、简单的存储和设备管理
桌面系统	Windows、Mac OS、Linux	图形处理（用户界面）、完善的设备管理和安全
服务器	UNIX 类、Windows、Linux	可靠性、可扩展性，完善的设备管理和安全

其实不同操作系统对体系结构的要求有较大的差距，人们在实际使用计算机时，会使用计算机对操作系统兼容性的概念，即某种操作系统可以兼容一些体系结构的计算机，如 Linux 可以兼容包括 x86 处理器在内的很多 RISC 处理器，Windows NT 可以运行在 x86 和 Alpha 两种处理器上，而 Windows 7 则只能运行在 x86 处理器上。

3．软件工具和中间件

软件工具和中间件对体系结构的影响，更多地体现在对计算机信息处理能力的需求上，包括更快的反应速度、更多的信息存储、更快的网络服务等。一般认为，中间件占用 10%～30%的系统资源，如果计算机性能不够强劲，就不能获得满意的服务。

1.2.3 应用的发展

1971 年的微处理器和 1981 年 PC 的出现以后，计算机的应用领域迅速扩大，同时也出现不断分化的趋势，处理器和计算机系统的设计往往因应用、需求和计算特征的不同而变化。近几年比较明显的趋势是计算机技术和市场分化成为桌面计算、服务器和嵌入式计算 3 个部分，这 3 个不同的领域应用需求的特点对计算机系统设计的影响巨大。

桌面计算市场是销售额最大的市场，是对性能价格比要求最为苛刻和敏感的市场，也是技术更新最快的市场，PC 就是这个市场的典型代表。近几年来，各种体系结构的创新技术、规范等一般总是在桌面计算市场上先出现，同时这个市场也称为产品的“价格杀手”，由于需求量巨大，竞争激烈。

服务器市场对计算机的要求是可用性、大容量和可扩展性。诸如 WWW 服务、电子邮件、电子商务、电子政务等重要服务，可用性永远是第一位的，一旦系统停止工作，不但会影响公众形象、损害网站的信用，而且会造成极大的、直接的和一系列间接的经济损失。因此，设计服务器的首要指标就是其 RSA（reliable/serviceable/available）指标。同时，由于应用的发展，服务器不但要考虑容量，还要考虑容量和有效服务能力的扩展性。

嵌入式计算，又称为嵌入式控制，其最大特征是看见的是一台设备，而不是一台计算机，计算机作为控制部件存在于设备之中。从 IC 智能卡、微波炉、移动电话、遥控器、机顶盒（set-top box）到机器人、汽车、飞机、卫星等，嵌入式计算机无所不在。嵌入式计算与解决的应用问题密切相关，需求千差万别，一般而言，嵌入式计算最重要的要求是实时性（real-time），也就是在规定的时间内响应请求并给出所需结果。另外被关注的指标还包括价格、功耗、应用成本、体积、工艺等，很多指标是随应用系统要求而变化的。

表 1.5 列出了这 3 类应用领域的一些典型特性。

表 1.5 桌面计算、服务器和嵌入式计算应用领域的一些典型特性

特　　征	桌面计算	服 务 器	嵌入式计算（只考虑 32 位和 64 位）
系统价格（美元）	500～5 000	5 000～5 000 000	10～100 000
每个处理器价格（美元）	50～500	200～10 000	0.01～100
关键指标	性能价格比、图形性能	吞吐率（服务流量）、可用性、可扩展性	随应用领域需求而变化，主要有：成本、功耗、实时性等

1.2.4 集成电路的发展

现代计算机实现技术的基础核心是以晶体管为基本单元的平面集成电路。1947 年，

固体物理学家 William B. Shockley、John Bardeen 和 Walter Brattain 在贝尔电话实验室共同发明了晶体管，并因此分享了 1956 年诺贝尔物理奖；1958 年，Jack Kilby 在得州仪器（TI）公司发明了集成电路理论模型，并因此获得 2000 年诺贝尔物理奖；1959 年，Bob Noyce 在仙童（Fairchild）公司发明了平面集成电路，1961 年开始批量生产。1965 年，时任仙童公司研发实验室主任的摩尔（Gordon Moore）在《Electronics》上撰文，认为集成电路密度大约每两年翻一番，这就是著名的摩尔定律。四十多年来，摩尔定律不但印证了集成电路技术的发展，也印证了计算机技术的发展。图 1.7 展示了内存芯片和 Intel 微处理器的发展变化。

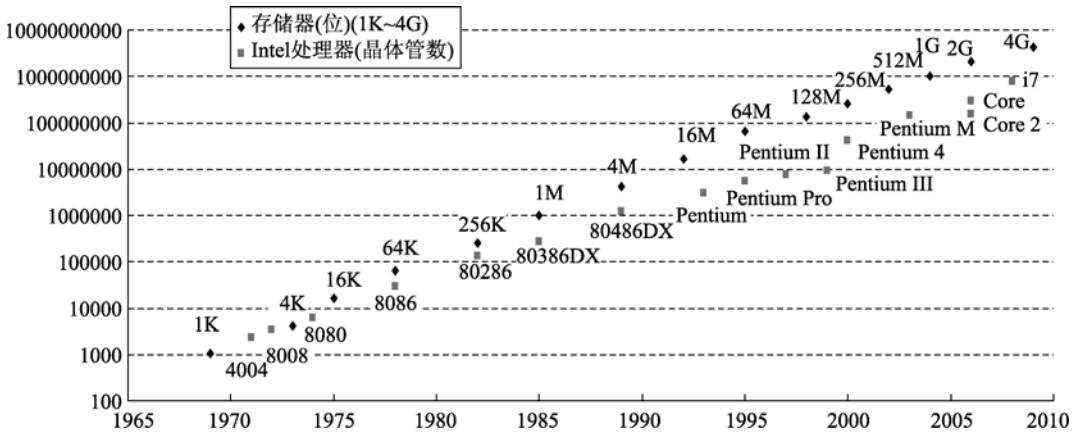

图 1.7　内存芯片密度和 Intel 微处理器集成度的发展

计算机系统的设计者不仅需要了解摩尔定律，更需要掌握技术的发展，尤其要注意实现技术日新月异的变化，其中有 4 种实现技术的变化发展极快，而这些技术对于当代计算机的发展发挥着非常关键的作用。

（1）逻辑电路。综合起来看，单芯片上的晶体管数量以每年 40%～55%的速率增长，其中，芯片上晶体管的密度每年大约增加 35%，芯片尺寸也在以每年 10%～20%速度增加。例如目前最新的 Intel Nehalem i7-980x，采用 32mm 工艺，有 11.7 亿只晶体管，管芯面积 248mm^2，工作频率 3.33GHz。

（2）半导体 DRAM（动态随机访问存储器）。芯片密度每年增长速率略低于 60%，平均 3 年增长 4 倍，存储周期时间的减少比较缓慢，大约是每 10 年减少 1/3，DRAM 通过接口的变化改善带宽。目前，DRAM 的设计和制造技术领先于逻辑电路，这主要是因为 DRAM 元件内单元电路晶体管数量少及 DRAM 采用专用技术产生。

（3）磁盘。最近磁盘密度以每年约 50%的速度增长，几乎每 3 年增长 4 倍。1990 年以前磁盘密度每年增长 25%，平均 3 年增长 2 倍，这表明未来几年内磁盘技术的发展将使磁盘密度继续保持高增长率。磁盘存取时间在过去 10 年内下降 1/3。

（4）网络。现代计算机网络都是计算机通过通信线路和交换中心设备连接的，因此网络的性能与通信线路和交换设备的性能有关。从性能指标来看，决定网络性能的关键指标有两个：网络带宽和网络延迟。多年来，网络技术一直是以带宽的突飞猛进为标志

的，以太网（Ethernet）从 10Mbps 发展到 100Mbps 用了约 10 年的时间，再发展到 1000Mbps（1Gbps）用了约 5 年的时间，又经过不到 3 年时间，目前 10 000Mbps（10Gbps）已经开始使用。同时，通信线路的介质由以同轴电缆为主发展到双绞线、光缆，近期的技术热点又发展到无线通信、自由空间激光通信等。

通过仔细研究可以发现，以上 4 个方面都是摩尔定律的某种表现形式。这些快速变化发展的技术对微处理器和计算机系统的设计具有很大影响。由于技术和工艺的不断提高，现代微处理器一般只有 5 年左右的使用寿命，甚至在一种产品的生产周期过程中（两年设计，两年生产），像 DRAM 这样的核心技术也在不断变化。设计人员必须经常采用最新技术进行计算机设计，因为最新技术的产品具有最高性价比和性能优势。值得注意的是，计算机产品的技术和性能的变化是骤变，而非渐变。例如，DRAM 技术是 3 年提高 4 倍，而不是 18 个月增长 2 倍；以太网的带宽是按照 10 倍增长。这种技术的跳变导致产品的跳变，从而使得原先不可能完成的技术变得可以实现。例如，当 MOS 技术在 20 世纪 80 年代初达到单芯片集成 25 000～50 000 个晶体管时，单片 32 位微处理器便实现了，在工艺成熟之后，其性价比便飞速提高。这种情况在计算机发展史上十分常见。

1.2.5 计算机体系结构的发展

存储程序计算机体系结构这些特点奠定了现代计算机发展的基础。但是，在冯·诺依曼等人提出这种结构时，由于受到当时硬件价格昂贵的限制，故使硬件完成的功能尽量简单，而把更多的功能交由软件完成。随着计算机应用领域的扩大，高级语言和操作系统的出现，这种功能分配的状况引起了愈来愈多的矛盾，迫使人们不断地对这种体系结构进行改进。

1．分布的 I/O 处理能力

存储程序计算机以运算器为中心，所有部件的操作都由控制器集中控制，这一特点带来了慢速输入输出操作占用快速运算器的矛盾。此时的输入输出操作和运算操作只能串行，互相等待，大大影响了运算器效率的发挥。尤其是计算机的应用进入商业领域之后，数据处理要求频繁地进行输入输出操作，上述缺点显得更加突出。

为了克服这一缺点，人们先后提出各种输入输出方式，如图 1.8 所示，从而引起计算机概念性结构发生了相应的变化。

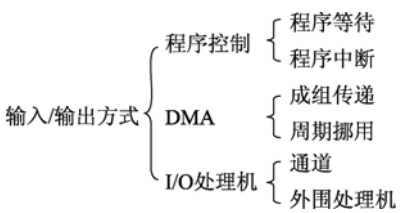

图 1.8 各种输入/输出方式

在程序等待方式之后，很快出现了程序中断的概念，并将它应用于输入输出操作（如 1954 年的 UNIVAC 1103 机）。这时，CPU 执行到一条输入输出操作指令后，可以不必等待外部设备回答的状态信息而继续执行后续指令。直到外部设备准备好发送/接收数据，向 CPU 发出中断请求，CPU 响应中断请求后才进行输入输出。这时，计算机结构仍以运算器为中心，但增加了

中断接口部件。程序中断概念的引入可以使 CPU 与外部设备在一定程度上并行工作，提高了计算机的效率，并且可以实现多种外部设备同时工作。中断技术已经成为现代计算机操作系统的技术基础。

随着外部设备种类和数量的增多，中断次数过于频繁，从而浪费了大量 CPU 的时间。对于成块（组）地进行传送的输入输出信息，出现了 DMA（直接存储器访问）方式。为了实现这种方式，需要在主存和设备之间增加 DMA 控制器（数据通道），从而形成了以主存以中心的结构。由 CPU 向 DMA 控制器寄存器置好初始参数后，仍可继续执行其后续指令。当外设准备好发送或接收数据时，便对 CPU 发 DMA 请求，使输入输出设备经数据通道直接与主存成组地交换信息。直到数据通道控制完成这一块数据传送后，才向 CPU 发结束信号，CPU 就可以进行接收或发送数据后的处理工作。

采用 DMA 方式，每传送完一组数据就要中断 CPU 一次。如果进一步使该部件能自己控制完成输入输出的大部分工作，从而使 CPU 进一步摆脱用于管理、控制 I/O 系统的沉重负担，这就出现了 I/O 处理机方式。I/O 处理机几乎把控制输入输出操作和信息传送的所有功能都从 CPU 那里接管过来，独立出去。I/O 处理方式有通道方式和外围处理机（I/O 子系统）方式两种。最早采用通道方式的是 IBM 360/370 系统。

2．保护的存储器空间

虽然传统存储程序计算机的存储程序原理现在仍为大多数计算机所采用，但是否把指令和数据放在同一存储器中，这一点不同计算机却有不同的考虑。指令和数据存放在同一个存储器中，会带来以下一些好处：指令在执行过程中可以被修改，因而可以编写出灵活的可修改的程序；对于存取指令和数据仅需一套读写和寻址电路，硬件简单；不必预先区分指令和数据，存储管理软件容易实现；程序和数据可以分配于任何可用空间，从而可更有效地利用存储空间等。然而，在实际应用中也看到：自我修改程序是难以编制、调试和使用的。一旦程序出错，进行程序诊断也不容易。程序的修改不利于实现程序的可再入性（reenterability）和程序的递归调用。在开发指令级并行时还可以看到：程序可修改也不利于重叠和流水方式的操作。因此，现在绝大多数计算机都规定：在执行过程中不准修改程序。这需要通过存储管理硬件的支持，由操作系统来实现。现代操作系统都实现了这种管理。

3．存储器组织结构的发展

按地址访问的存储器具有结构简单、价格便宜、存取速度快等优点。但是在数据处理时，往往要求查找具有某种内容特点的信息。在随机存储器中，虽然可以通过软件，按一定的算法（顺序查找、对分查找或采用 Hash 技术等）完成查找操作，但由于访问存储器的次数较多而影响了计算机系统的性能。按内容访问的相联存储器（content addressed memory，CAM），把查找、比较的操作交由存储器硬件完成。如果让相联存储器除了完成信息检索任务外，还能进行一些算术逻辑运算，则就构成了以相联存储器为核心的相联处理机。为了减少程序运行过程中访问存储器的次数，人们早在 1956 年的

Pegasus 计算机上采用了通用寄存器的概念。它不但把变址寄存器和累加器结合起来使用，提供了多累加器结构，而且使中间结果不必访问存储器。这个概念为现代计算机普遍采用，通用寄存器的数量也由几个增加到十几个或几十个，有些 RISC 处理器中增加到几百个。为了进一步减少访问存储器的次数和提高存储系统的速度，人们又提出了在 CPU 和主存之间设置高速缓冲存储器 Cache。这种技术当初是在大型机上采用的，而现在 PC 也使用了多至 1MB 的两级或者三级 Cache 存储器。

4．并行处理技术

传统的存储程序计算机解题算法是顺序型的，即使问题本身可以并行处理，由于程序的执行受程序计数器控制，故只能是串行、顺序执行。因此，如何挖掘传统机器中的并行性，一直是计算机设计者努力的方向。人们通过改进 CPU 的组成，如采用重叠方式，先行控制、多操作部件甚至流水方式把若干条指令的操作重叠起来；或者在体系结构上把本来可并行的计算机的题目使之能并行计算，如对向量的计算就可以用一条向量指令并行地对向量的各个元素进行相同的运算。更进一步，如果把一个作业（程序）划分成能并行执行的多个任务（程序段），把每个任务分配给一个处理机执行，则构成了多机并行处理系统。这种多机并行处理系统由于有很高的并行性，而成为提高计算机系统速度最有潜力的手段之一。

5．指令集的发展

指令集是传统机器程序员所看到机器的主要属性。指令仍由操作码和地址码两部分组成，它在两个方面对计算机体系结构设计产生重大影响：一是指令集的功能，二是指令的地址空间和寻址方式。

自 20 世纪 50 年代开始，计算机系统中指令的种类和数量逐渐增加，后来“软件危机”出现，人们也认为大量功能丰富的由硬件实现的指令，不但可以缓解“软件危机”，而且可以提高计算机系统的性能。到 20 世纪 80 年代，一个指令系统中所含指令的数目可达 300～500 条，这种计算机称为复杂指令集计算机（complex instruction set computer，CISC）。后来的研究表明，过多的和过于复杂的指令难于理解、不好使用，大量种类指令的实际利用率非常低，同时，由于复杂功能指令的实现复杂，指令译码电路庞大，反而降低了计算机系统的性能。到 1979 年，由 D. A. Patterson 等人提出了精减指令系统计算机（reduced instruction set computer，RISC）的设想，把指令系统设计成只包含那些使用频率高的少量指令，并提供一些必要的指令以支持操作系统和高级语言。按照这个原则而构成的计算机称为精简指令集计算机 RISC。RISC 的技术思想已经成为当代计算机设计的基础技术之一。

早期计算机地址码是直接指出操作数地址的（直接寻址），这是一种简单快速的寻址方法。随着计算机存储器容量的扩大，指令中地址码的位数已不能满足整个主存空间寻址的要求，使直接寻址方式受到很大限制。现代计算机指令地址码部件一般给出的是形式地址，而由各种寻址方式，按照规定的算法把形式地址变换成访问主存的有效地址，如变址寻址（1949 年 EDSAC）、间接寻址（1958 年 IBM 709）、相对寻址（如

PDP-11）等，由于多道程序的要求，出现了基址寻址（如IBM 360），为了对虚拟存储器进行管理，出现了页式寻址（1959 年 Atlas）。现代的微、小型计算机由于指令字长较短，一般都具有多种灵活的寻址方式。多种寻址方式在带来灵活性的同时也带来了设计、生产和使用的复杂性。所以采用 RISC 技术的计算机一般只有常用的几种寻址方式。

1.2.6 并行处理技术的发展

研究计算机体系结构的目的是提高计算机系统的性能。开发计算机系统的并行性，是计算机体系结构的重要研究内容之一。本节首先叙述体系结构中的并行性概念，然后从单机系统和多机系统两个方面对并行性的发展进行归纳，以期认识计算机体系结构中并行性发展的全貌。

1．并行性概念

并行性（parallelism）是指在同一时刻或是同一时间间隔内完成两种或两种以上性质相同或不相同的工作。只要时间上互相重叠，就存在并行性。严格来讲，把两个或多个事件在同一时刻发生的并行性叫做同时性（simultaneity）；而把两个或多个事件在同一时间间隔内发生的并行性叫做并发性（concurrency）。以 *n* 位并行加法为例，由于存在着进位信号的传播延迟时间，全部 *n* 位加法结果并不是在同一时刻获得的，因此并不存在同时性，而只存在并发性的关系。如果 *m* 个存储器模块能同时读出信息，则属于同时性。以后，除非特殊说明，本书不严格区分是哪种并行性。

计算机系统中的并行性有不同的等级。从执行程序的角度看，并行性从低到高可分为以下 5 个等级。

（1）指令内部并行：指令内部的微操作之间的并行，以微操作为调度单位。其实现技术主要是采用硬件。这些研究内容主要涉及计算机的组织和逻辑设计的内容，本教材不系统讨论这方面的问题。

（2）指令级并行（instruction level parallel，ILP）：并行执行两条或多条指令，以基本块中的指令为调度单位。其实现技术需要大量硬件，同时需要得到编译的支持才能够发挥多指令流出的效能。本书第 3 章、第 4 章将集中深入讨论有关技术专题。

（3）线程级并行（thread level parallel，TLP）：并发执行多个线程，通常是以一个进程内控制派生的多个线程为调度单位。其实现的技术难点在于编译技术，同时简洁、高效的硬件支持必不可少，否则严重影响执行效率。作为技术前沿，本书第 4 章、第 5 章和第 7 章将涉及有关技术问题。

（4）任务级或过程级并行：并行执行两个或多个过程或任务（程序段），以子程序或进程为调度单位。软、硬件综合的复杂系统，其技术难点在于并行算法、程序设计和硬件高效支持，系统必须提供一些程序设计手段。本书在第 7 章研究有关的技术问题。

（5）作业或程序级并行：在多个作业或程序间的并行，以程序或作业为调度单位。

其特点与任务级并行相似。本书第 7 章涉及有关内容。

在单处理机系统中，这种并行性升到某一级别后（如任务、作业级并行），则要通过软件（如操作系统中的进程管理、作业管理）来实现。而在多处理机系统中，由于已有了完成各个任务或作业的处理机，其并行性是由硬件来实现的。因此，实现并行性也有一个软硬件功能分配问题，往往也需要折中考虑。

从处理数据的角度，并行性从低到高可以分为以下 4 个等级。

（1）字串位串：同时只对一个字的一位进行处理。这是最基本的串行处理方式，不存在并行性。

（2）字串位并：同时对一个字的全部位进行处理，不同字之间是串行的。这里已开始出现并行性。

（3）字并位串：同时对许多字的同一位（称为位片）进行处理。这种方式有较高的并行性。

（4）全并行：同时对许多字的全部或部分位进行处理。这是最高一级的并行。

在一个计算机系统中，可以采取多种并行性措施。既可以有执行程序方面的并行性，又可以有处理数据方面的并行性。当并行性提高到一定级别则称之为进入并行处理领域。例如，执行程序的并行性达到任务或过程级，或者处理数据的并行性达到字并位串一级，即可认为进入并行处理领域。所以，并行处理（parallel processing）是信息处理的一种有效形式。它着重挖掘计算过程中的并行事件，使并行性达到较高的级别。并行处理是硬件、体系结构、软件、算法、语言等多方面综合研究的领域。

1966 年，Michael J. Flynn 依据计算机中同时存在的指令流和数据流的数量，提出了 Flynn 分类法（Flynn's taxonomy）。Flynn 分类法按照指令的数据的关系，将计算机从并行处理的角度划分为以下 4 类模型。

（1）单指令流单数据流（single instruction single data stream，SISD）：指令和数据都不存在并行，完全串行执行，早期简单的单处理器计算机都是这样工作的。

（2）单指令流多数据流（single instruction multiple data streams，SIMD）：一个指令流同时对多个（组）数据进行相同的操作。典型机器包括阵列机（array processor）和图形处理器（GPU）。

（3）多指令流单数据流（multiple instruction single data stream，MISD）：多个指令流在一个数据流上进行操作。这种情况并不常见，多处理器容错结构的计算机系统可以算是一种，这种容错结构一般用在可靠性和实时性要求都非常高的环境中。

（4）多指令流多数据流（multiple instruction multiple data streams，MIMD）：多个独立的处理器在不同的数据上执行不同的指令，这是很常见的并行处理模型。

图 1.9 是 Flynn 分类法的 4 种概念模型的形象说明。

Flynn 分类的概念在目前体系结构研究中使用较多，特别是 SIMD 和 MIMD 两种并行计算模型是现在并行计算的主要模型。同时，Flynn 分类还产生了一些延伸的概念：按照程序（program）进行分类，可以形成单程序多数据流（SPMD）和多程序多数据流（MPMD），按照线程（thread）进行分类，就有单线程多数据流（STMD）和多线程多数据流（MTMD），等等。

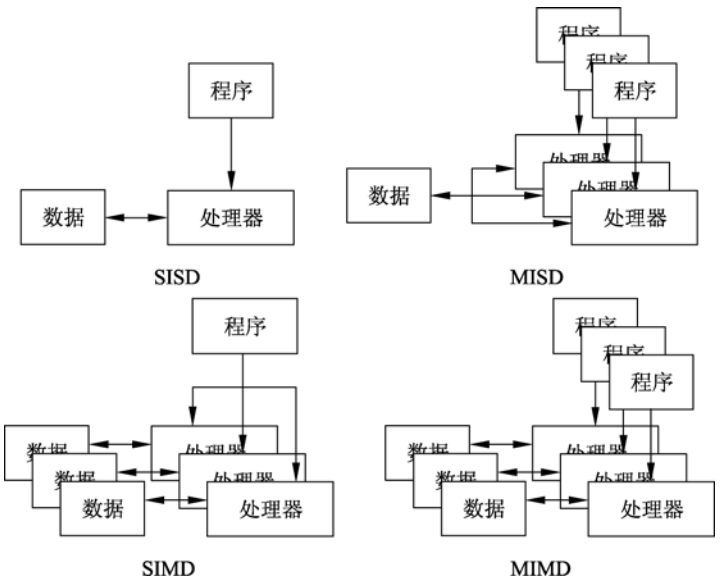

图 1.9　Flynn 分类法的 4 种概念模型

2．提高并行性的技术途径

计算机系统中提高并行性的措施多种多样，但就其思想而言，都可纳入下列 3 种途径。

（1）时间重叠（time-interleaving）：在并行性概念中引入时间因素，即多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分，以加快硬件周转而赢得速度。时间重叠原则上不要求重复的硬件设备。流水线技术是时间重叠的典型实例。

（2）资源重复（resource-replication）：在并行性概念中引入空间因素，是根据“以数量取胜”的原则，通过重复地设置资源，尤其是硬件资源，以大幅度提高计算机系统的性能。随着硬件价格的降低，这种方式在单处理机上广泛采用，而多处理机本身就是资源重复的结果。

（3）资源共享（resource-sharing）：这是一种软件方法，它使多个任务按一定时间顺序轮流使用同一套硬件设备。例如，多道程序、分时系统就是遵循资源共享这一途径产生的。资源共享既降低了成本，又提高了计算机系统中设备的利用率。网络打印技术就是一种分布系统中资源共享的典型实例。

下面先分析单机系统中并行性的发展。在发展高性能单处理机过程中，起着主导作用的是时间重叠这个途径。实现时间重叠的基础是部件功能专用化（functional specialization）。即把一件工作按功能分割为若干相互联系的部分，把每一部分指定给专门的部件完成；然后按时间重叠原则把各部分执行过程在时间上重叠起来，使所有部件依次分工完成一组同样的工作。例如，将执行指令的过程分解为取指令、指令译码、指令执行、访问存储部件和写结果 5 个步骤（见图 1.2），每个步骤使用专门化的部件，就需要以下 5 个专用的部件，即取指令部件（IF）、指令译码部件（ID）、指令执行部件（EX）、访问存储部件（M）和写结果部件（WB）。把它们的工作按某种时间重叠关系重叠起来，

构成计算机中央处理机的指令执行流水线（或称为指令流水线），使得在处理机内部能同时处理多条指令，从而提高处理机的速度，如图 1.10 所示。这种流水化处理技术开发了计算机系统中的指令级并行。

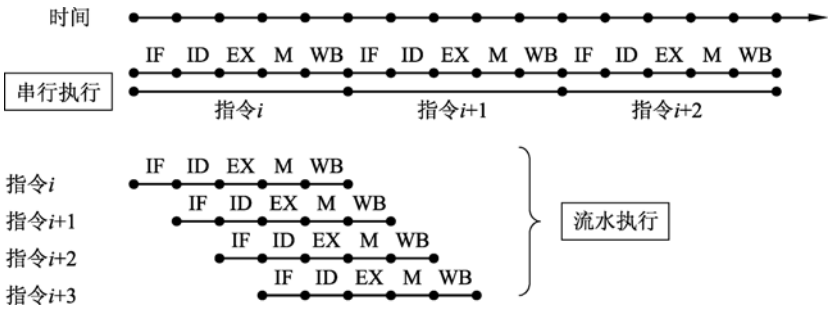

图 1.10　指令的串行执行和分解成 5 个步骤的指令流水执行

在单处理机中，资源重复的运用已经普遍起来。不论是在非流水线处理机（如CDC 6600），还是在流水线处理机（如IBM 360/91）中，多操作部件和多体存储器都是成功应用的结构形式。在多操作部件处理机中，通用部件被分解成专门部件，例如加/减法部件、乘法部件、逻辑运算部件等。一条指令所需的操作部件只要没有被占用，就也可以开始执行，这就是指令级并行。进一步，可以重复设置多个相同的处理单元，在同一个控制器指挥下，按照同一条指令的要求，对向量的各元素同时进行操作，这就是并行处理机。它在指令内部实现了对数据处理的全并行，从而把单处理机的并行性又提高一步，进入了并行处理领域。并行处理机本身还是单处理机。如果再进一步提高其并行性，使其达到任务级并行，则每个处理单元都必须是独立处理机，这就进入多处理机领域。如果进一步提高其并行性，使其达到任务级并行，则每个处理单元都必须有自己的控制器，能独立地解释和执行指令，成为一台独立完整的处理机，这就进入了多处理机范畴。这种多处理机系统称为对称型（symmetrical）或同构型多处理机系统（homogeneous multiprocessor system）。它们由多个同类型，至少是由同等功能的处理机组成，同时地处理同一作业中能并行执行的多个任务。

资源共享的概念，在单处理机系统中实质上是用单处理机模拟多处理机的功能，形成虚拟机的概念。比如分时系统，在多终端的情况，每个终端上的用户感到好像自己有一台处理机一样。远程终端的出现，改变了计算机系统地理上和逻辑上“集中”的局面，开始向“分布”方向发展。当计算机之间互相连接，分工合作时，则进入了多机系统，这种多机系统称为分布处理系统（distributed processing system）。

下面回顾多机系统中并行性的发展。多机系统也遵循着时间重叠、资源重复和资源共享的技术途径，向着 3 种不同的多处理机方向发展。但在采取的技术措施上与单机系统稍有差别。

为了反映多机系统的各机器之间的物理连接的紧密程度和交互作用能力的强弱，人们引进耦合度的概念。多机系统的耦合度，可以分为最低耦合系统、松散耦合系统、紧密耦合系统等几类。

（1）最低耦合系统（least coupled system）：是耦合度最低的系统。除通过某种中间存储介质之外，各计算机之间没有物理连接，也无共享的连机硬件资源。

（2）松散耦合系统（loosely coupled system）或间接耦合系统（indirectly coupled system）：一般是通过通道或通信线路实现计算机间互连，共享某些外围设备（如磁盘、磁带等），机间的相互作用是在文件或数据集一级进行。松散耦合系统常表现为两种形式：一种是多台计算机和共享的外围设备连接，不同机器之间实现功能上的分工（功能专用化），机器处理的结构以文件或数据集的形式送到共享的外围设备，供其他机器继续处理；另一种是计算机网，通过通信线路连接，以求得更大范围内资源共享。

（3）紧密耦合系统（tightly coupled system）或直接耦合系统（directly coupled system）：一般是指机间物理连接的频带较高，它们往往通过总线或高速开关实现互连，可以共享主存。由于具有较高的信息传输率，从而为快速并行处理一个作业或多个任务创造了条件。

在单机系统中，要做到时间重叠必须有多个专用功能部件，即把某些功能分离开由专门部件去完成。而在多处理机中是将处理功能分散给各专用处理机完成，即功能专用化。各处理机之间按照时间重叠原理工作。早期，是把一些辅助性功能由主机分离出来，交给一些较小的专用计算机完成。如输入/输出功能的分离，导致了通道和专用外围处理机发展。它们之间往往采取松散耦合方式，形成各种松散耦合系统。这种趋势发展下去，许多主要功能如数组运算、高级语言编译、数据库管理等也逐渐分离出来，交由专用处理机完成，机间的耦合程序也逐渐加强，发展成异构的多处理机系统。

最早的多机系统并不是为了提高速度，而是为了在关键性的工作中保证系统的可靠性。通过设置多台相同类型的计算机，使系统工作的可靠性在处理机一级得到提高。各种不同的容错多处理机系统方案，对机间互联网络的要求是不同的，但正确和可靠性是最起码的要求。如果提高对互联网络的要求，使其具有一定的灵活性、可靠性和可重构性，则发展成一种可重构系统（reconfigurable system）。在这种系统中，平时几台计算机都正常工作，像通常的多处理机系统一样。但到故障阶段，就要使系统重新组织，降低档次继续运行，直到故障排除为止。随着硬件价格的降低，现在人们更多的是通过多处理机的并行处理，提高整个系统的速度。这时，对机间互联网络的性能提出了更高要求。高带宽、低延迟、低开销的机间互联网络，是高效实现程序段或任务一级并行处理的前提条件。为了使并行处理的任务能在处理机之间随机地进行调度，就必须使各个处理机具有同等的功能，成为同构型多处理机。表 1.6 对上述 3 种多处理机进行了简单的比较和总结。由表可以看出，分布式系统与其他两类多处理机系统在概念上是存在交叉的。无论是单机系统还是多机系统，它们都是按不同的技术途径向 3 种不同类型的多处理机发展。

这些多处理机系统的有关技术问题，本书在第 7 章中将进行比较全面的介绍。

3．并行计算的应用需求

应用需求永远是计算机系统性能提高的最大动力。当前许多领域的发展已经依赖于计算，特别是高性能计算，而并行计算则是满足这些永无止境计算能力要求的几乎唯一

表 1.6 3 种类型多处理机比较

项 目	同构型多处理机	异构型多处理机	分布处理系统
目的	提高系统性能（可靠性、速度）	提高系统使用效率	兼顾效率与性能
技术途径	资源重复（机向互连）	时间重叠（功能专用化）	资源共享（网络化）
组成	同类型（同等功能）	不同类型（不同功能）	不限制
分工方式	任务分布	功能分布	硬件、软件、数据等各种资源分布
工作方式	一个作业由多机协同并行地完成	一个作业由多机协同串行地完成	一个作业由一台处理机完成，必要时才请求其他机器协作
控制形式	常采用浮动控制方式	采用专用控制方式	分布控制方式
耦合度	紧密耦合	紧密、松散耦合	松散、紧密耦合
对互联网络的要求	快速性、灵活性、可重构性	专用性	快速、灵活、简单、通用

可行的出路。在科学理论领域，包括物理学、化学、数学、材料科学、生物学、天文学、地球科学等，在工程技术领域，包括石油、化工、汽车、航空、航天、制药、气象、能源、基因等，对并行计算都有着无尽的期待。同样，科学计算结果的处理，如可视化技术，也是计算量巨大的。另外一个重要领域就是娱乐业，1995 年的“玩具总动员”（Toy Story）开创了全计算机动画电影的先河，2010 年的“阿凡达”（Avatar）则把计算量巨大的 3D 视频带到了人们的生活中，与此同时，越来越精细、逼真的计算机游戏，极大地拉动了计算机图形学、高档 CPU、高性能 GPU 甚至影响了计算机系统的发展。

目前，人们已经看到的需要 TFLOP 计算能力和 TB 存储能力的应用领域包括：重大机械问题、全球气象变化、基因/蛋白质功能、湍流、机械动力学、洋流、量子色动力学、复杂巨系统建模、深层空间问题等。所以，人们对并行技术的研究还将继续深入下去，来满足人类在了解自然和持续发展的过程中对计算能力的需求。

1.3 计算机系统设计和分析

1.3.1 成本与价格

1．计算机系统的成本和价格

首先，组装一台作为图形工作站使用的较高档配置 PC，看看硬件各部件价格，表 1.7 列出了价格细节，当然，其中每个部件的选择都有更改调整的余地。但是，在大多数配置中，显示器系统在总成本中都占有较大的比重，在面向图形应用的工作站中更是如

此，本配置中显卡加显示器的价格达到总价格的 41.6%。

表 1.7　一台组装三屏 PC 图形工作站及其各个部件的价格分布

配　　件	品牌/型号	价格* ￥	总价格中的比例
处理器	Intel Core i5 750（盒）	1380	11.5%
主板（含基本 I/O）	技嘉 GA-P55-UD3L	1099	9.2%
存储器	南亚 2GB DDR3 1333 ECC（两条）	890	7.5%
硬盘	西部数据 RE3 1TB×2（RAID 1）	1860	15.6%
显示卡	蓝宝石 HD5770 1GB GDDR5	1199	10.0%
显示器	优派 VX2240w（DVI）×2+戴尔 P2210（DP）	3790	31.6%
光存储	三星 TS-H622A	185	1.5%
键盘和鼠标	微软极动套装黑色版	160	1.4%
机箱	Tt M5	480	4.0%
电源	Tt 金刚 KK500A	450	3.8%
音箱	麦博梵高 FC550 升级版	468	3.9%
总价		11 961	100%

*此价格为 2010 年 1 月的媒体报价。

众所周知，组装计算机的价格，尤其是部件的价格，是经常变化的。但是从总体看，价格变化的趋势是不断下降的。计算机的价格是与成本紧密相关的，影响价格的因素通常就是那些影响成本的因素，成本会影响计算机中部件的选择。价格与成本是不同的概念。部件的成本会限制设计，但成本并不代表用户必须付出的价格，成本在变成实际价格之前会出现一系列的变化，系统设计者必须清楚设计方案对最终的销售价格的影响。一般情况下，成本的变化反映到价格上将放大 3～4 倍。价格上扬时计算机销售势头就不好，产量就会下降，成本就会增大，并导致价格进一步增长。因此，小小的成本变化会对产品的市场产生很大的影响。

构成价格的各因素可以通过占成本或价格的百分比来表示。价格与成本的差别也因销售市场的不同而不同。

简单地说，商品的标价（价格）由这样一些因素构成：原料成本、直接成本、毛利和折扣。

原料成本是指一件产品中所有部件的采购成本总和。它是价格中最明显的部分，也是对计算机系统设计影响最明显的部分。

直接成本是指与一件产品生产直接相关的成本，包括劳务成本、采购成本（如运输、包装费用）、零头（剩余的零头）及产品质量成本（如人员培训、生产过程管理等）。直接成本通常是在部件成本上增加 10%～30%。

毛利是公司开支的一部分，这一部分是无法由一件产品直接支付的，它必须均摊到每一件产品中去。毛利主要包括：公司的研发费用、市场建立费用、销售费用、生产设备维护费用、房租、贷款利息、税后利润和所得税等。原料成本、直接成本和毛利相加，

就得到平均销售价格。毛利一般占到平均销售价格的 10%～45%，具体情况取决于产品的独立性。导致低端 PC 产品制造商具有较低毛利的几个主要原因是：首先，由于产品的标准化，他们的研发费用较低；其次，他们采取非直接销售方式（通过邮购、电话订购或零售店），所以其销售成本较低；第三，因为这一类产品缺乏独特性，竞争激烈导致低价格和低利润，因而毛利也较低。

标价与平均销售价格并不一样，原因之一是公司提供了批发价格折扣，产品通过零售店进行销售，零售商也需获得利润。

正如人们所见到的，价格随竞争程度的变化而变化。公司在销售产品时可能无法取得所期望的毛利。在更坏的情况下，价格猛跌而导致无法取得利润。一家公司争夺市场份额的方法就是降低价格，从而提高产品的竞争力，如果产量增加则成本就会减小，就有可能维持其利润。它们之间的关系是极其复杂的，这里不可能深入分析。

在美国，大部分公司只将收入的 4%（商用 PC）～12%（高端服务器）用于研发。这一百分比将不会轻易随时间变化。

图 1.11 通过 PC 产品的成本和价格的分布，对上述概念进行了形象的说明。关于成本及成本/性能的问题是很复杂的，计算机系统的设计往往也不是单一目标。一种极端是巨型计算机设计，为达到高性能而不考虑成本；另一种极端是低成本机器，为达到低成本就需要牺牲一些性能，低端的 PC 就属于这一类。位于这两种极端之间的就是性能/成本设计，设计者需取得性能与成本之间的平衡。绝大部分工作站、服务器等制造商就属于这一类。在过去的几十年中，计算机尺寸变小，因此低成本设计和成本/性能设计就显得日益重要，即使是巨型计算机制造商也发觉成本问题已日益重要。

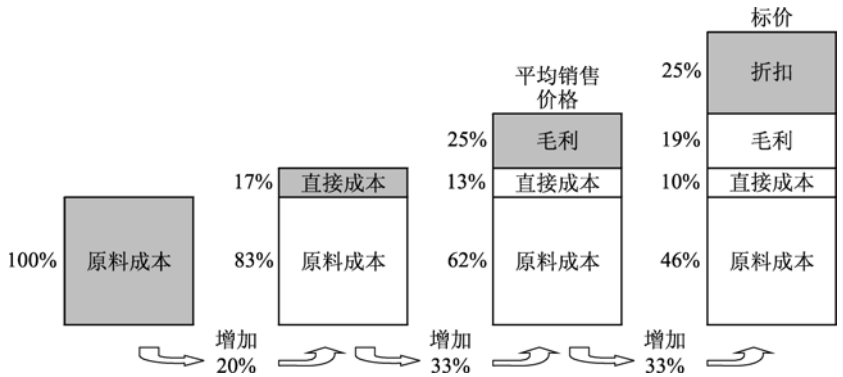

图 1.11　PC 的成本和价格

2．时间因素

对计算机系统成本产生影响的主要因素有时间、产量、商品化等因素。

对成本产生最直接影响的是时间。即使实现技术没有变动，计算机系统的制造成本也会不断下降。随着时间的推移，生产工艺会日渐稳定，产品的成品率会不断提高。产品的成本与成品率成反比。在规划产品生产周期时，成本随时间变化是非常关键的因素，这也是工业化生产的一个重要特点。我们不难理解，产品的成品率翻一番，成本将下降

一半。

产量是决定产品成本的第二个关键因素。首先，产量的增加会加速工艺的稳定；第二，产量增加就提高了生产效率，降低了成本；第三，产量增加还可降低每台单机必须加入的开发费用，从而使得单机成本下降。统计结论是：无论是集成电路芯片、印刷电路板或系统，如果产量翻一番，那么成本就会减少 10%。

商品化也是影响产品成本的重要因素，但更重要的是它影响产品的价格。商品就是指市场上销售的批量化的产品，如 DRAM、磁盘、显示器、键盘等。商品化包括建立市场和销售渠道的过程，如广告、代理、维修等。

综合产品价格变化的各个因素，最终反映到产品价格随时间变化的特性上，就是价格随时间下降的趋势。这种变化的趋势与人类学习知识时候的记忆曲线非常相似，因此人们称之为价格的学习曲线（learning curve）。虽然各种产品都有类似的价格变化曲线，但是不同产品的变化速度是不同的，图 1.12 为存储器价格变化的学习曲线。其他产品具有类似的曲线。

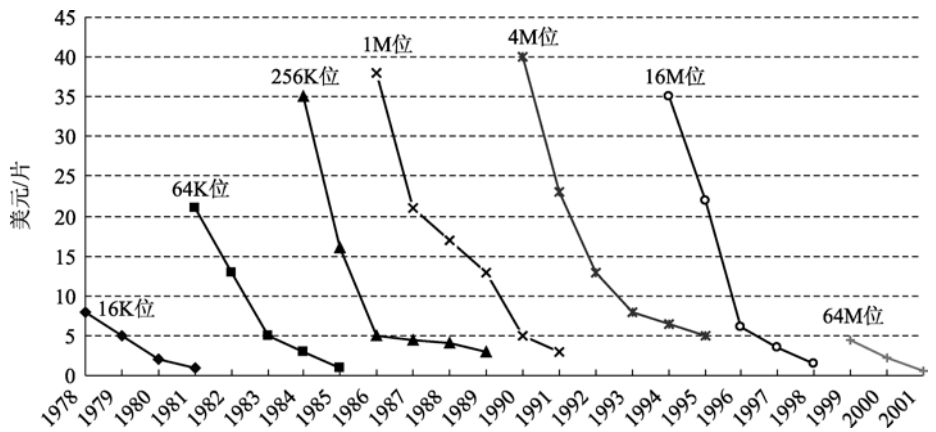

图 1.12　DRAM 价格的学习曲线

1.3.2　基准测试程序

既然性能与测试程序的执行时间相关，那么用什么程序进行测试呢？如果用户仅使用计算机完成某种特定的应用，那么这组应用程序就是评估计算机系统性能的最佳测试程序。用户通过比较在不同系统中这组应用程序的响应时间，就可以知道计算机的性能。然而，这种情况实际上比较少见。大部分人必须依靠其他测试程序，以获得机器的性能。目前常用的测试程序可以分为 5 类，下面按测试可靠性由高至低的顺序列出。

（1）实际应用程序：这是最可靠的方法。即使用户对计算机性能测试一窍不通，通过运行实际应用程序，也可以清楚地知道计算机的性能。实际应用程序是随应用而变化的，如 C 编译器，进行文本处理软件 TeX（UNIX）、MS Word（Windows、Mac OS）、WPS（DOS、Windows、Linux）等，进行 CAD 设计的工具 Spice、AutoCAD，进行图片

处理的 Photoshop（Windows、Mac OS）等。用户自己的应用系统也属于这一类程序，如 MIS、工业控制系统等。使用实际应用程序可能会引发另外一个问题，由于操作系统或者编译器的变化，导致实际应用程序不能够在测试系统中正常运行。这也就是通常所说的软件（程序）可移植性问题，引起这类问题的主要原因是由于实际应用软件和机器之间的相关性，也就是一些软件功能依赖于特定的硬件系统，如图形系统、交互设备、专用设备、特殊设备等。

（2）修正的（或者脚本化）应用程序：很多情况下，通过修正实际应用程序的部分代码或者通过脚本描述来模拟实际应用，再用它们构成测试程序。之所以这样处理程序，通常是基于以下几点考虑：一是解决程序的可迁移性问题，这一点前面已经说明了；二是为了回避程序的一些次要特性，如交互时人的因素，更加突出程序受到的特定因素的影响；三是简化一些程序的复杂性，降低测试的复杂程度，减少测试开销。

（3）核心测试程序：由从真实程序中提取的较短但很关键的代码构成。Livermore Loops 及 LINPACK 是其中使用比较广泛的例子。这些代码的执行时间直接影响到程序总的响应时间。用户不会直接使用核心测试程序，因为它的功能只是用来测试计算机性能。核心测试程序可以根据需要评价机器的各种性能，从而解释在运行真实程序时机器性能不同的原因。

（4）小测试程序：小测试程序通常是指代码在几十行到 100 行的具有一些特定目的的测试程序。用户可以随时编写一些这样的程序来测试系统的各种功能，并产生用户已预知的输出结果，如皇后问题、迷宫问题、快速排序、求素数、计算 π 等，当然也编写一些测试特定指标的小测试程序，如通过频繁显示模式转换测试显示特征、通过大量小文件的建立/读/写/删除等测试磁盘的速度等，这类流行的测试程序都具有短小、易输入、通用等特点，最适于进行一些基本测试。

（5）合成测试程序：设计合成测试程序的基本思想与设计核心测试程序是相同的，但是合成测试程序面向大量应用程序中操作和数据的统计特征。首先对大量的应用程序中的操作进行统计，得到各种操作比例，再按这个比例人为制造出测试程序。Whetstone 与 Dhrystone 是最流行的合成测试程序。在操作类型和操作数类型两个方面，合成测试程序试图保持与大量程序中的比例一致。用户不会自己产生合成测试程序，因为其中没有任何用户能够使用的代码。合成测试程序与实际应用相差更远，核心程序起码是从真实程序中提取出来的，而合成测试程序则完全是人为制造出来的。

要在竞争激烈的计算机市场中生存和发展，就必须努力提高计算机产品的性价比。所以，每家计算机系统设计公司都投入大量的人力、物力资源研究各种通用的测试程序，并针对测试结果，从硬件和软件两个方面对系统设计进行修改和优化，以提高计算机系统的总体测试性能。但是，通用计算机系统一般不针对某一个特定的真实程序进行设计或性能优化，因为这样做不但难度大，而且会增加成本。为了提高测试的公正性，通用测试程序往往由非商业性组织或者第三方厂商提供。通用测试程序在使用时也有明确的要求，如系统配置、数据精度、编译优化等，以期获得的测试结果具有良好的可比性。

目前有一种日渐普及的测试程序产生方法，就是选择一组各个方面有代表性的测试

程序，组成一个通用测试程序集合。这种测试程序集合称为测试程序组件（benchmark suites），它的最大优点是避免了独立测试程序存在的片面性，尽可能全面地测试了一个计算机系统的性能。

目前在评价计算机系统设计时最常见的测试程序组件是基于 UNIX 的 SPEC，它诞生于 20 世纪 80 年代，当时主要用于测试各种使用 UNIX 的工作站，其主要版本包括 SPEC89、SPEC92、SPEC95、SPEC2000 和 SPEC2006 等。本教材的很多测试数据都是基于 SPEC 的，研究计算机体系结构的很多技术文章中使用的测试结果也是基于 SPEC。

事务处理（transaction-processing，TP）测试程序主要测试在线事务处理（on-line transaction processing，OLTP）系统的性能，其核心内容是数据库访问和相关的信息决策能力。20 世纪 80 年代，相关工程技术人员成立了一个称为 TPC（Transaction Processing Council）的独立组织，1985 年发布第一个 TPC 测试程序 TPC-A，并先后发布多个修改版本并补充了 4 个不同的测试程序，构成 TPC 测试程序组件。

目前，市场上还有一大类常用的测试程序组件，用于测试基于 Microsoft 公司的 Windows 系列操作系统平台的测试组件，这一类测试平台目前已经非常完整，这里仅仅列举一些最常用的测试组件。PCMark04 包括中央处理器测试组、内存测试组、图形芯片测试组、硬盘测试组等。Business Winstone 2004 主要用于测试计算机系统商业应用的综合性能。

还有一些专门的性能指标测试程序：3DMark03 主要测试显卡性能和 DirectX 的性能。SuperPi/SuperE 是计算圆周率 π/自然指数 e 的软件，通常用来测试 CPU 的稳定性。

1.3.3 量化设计的基本原则

我们已知道如何定义、度量和比较计算机系统的性能，下面讨论计算机体系结构设计和分析中最经常使用的几条基本原则和方法。

1．大概率事件优先原则

大概率事件优先原则是计算机体系结构设计中最重要和最常用的原则。这个原则的基本思想是：对于大概率事件（最常见的事件），赋予它优先的处理权和资源使用权，以获得全局的最优结果。也就是“好钢用在刀刃上”，以达到事半功倍的效果。

在进行计算机设计时，如果需要权衡，就必须侧重常见事件，使最常发生事件（大概率事件）优先。此原则也适用于资源分配。着重改进大概率事件性能，能够明显提高计算机性能。另外，大概率事件通常比小概率事件简单，而且容易使之更快完成。例如，CPU 在进行加法运算时，运算结果无溢出为大概率事件，而溢出为小概率事件。因此，应该针对无溢出情况进行 CPU 优化设计，加快无溢出时加法计算速度。虽然发生溢出时机器速度可能会减慢，但由于溢出事件发生概率很小，所以总体上机器性能还是提高了。

在这本书中将经常看到该原则的应用。重要的是，要能够确定什么是大概率事件，同时要说明针对该事件进行的改进将如何提高机器的性能。

2．Amdahl 定律

Amdahl 定律既可以用来确定系统中对性能限制最大的部件，也可以用来计算通过改进某些部件所获得的系统性能的提高。Amdahl 定律指出：加快某部件执行速度所获得的系统性能加速比，受限于该部件在系统中所占的重要性。

首先，Amdahl 定律定义了加速比这个概念。假设我们对机器进行某种改进，那么机器系统的加速比是：

$$\text{系统加速比} = \frac{\text{系统性能}_{\text{改进后}}}{\text{系统性能}_{\text{改进前}}}$$

或者

$$\text{系统加速比} = \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}}$$

系统加速比告诉人们改进后的机器比改进前快多少。Amdahl 定律使人们能够快速得出改进所获得的效益。系统加速比依赖于以下两个因素。

（1）可改进部分在原系统计算时间中所占的比例。例如，一个需运行 60s 的程序中有 20s 的运算可以加速，那么该比例就是 20/60。这个值用“可改进比例”表示，它总是小于等于 1 的。

（2）该部分改进以后的性能提高。例如，系统改进后执行程序，其中可改进部分花费 2s 的时间，而改进前该部分需花费 5s，则性能提高为 5/2。用“部件加速比”表示性能提高比，一般情况下它是大于 1 的。

部件改进后，系统的总执行时间等于不可改进部分的执行时间加上可改进部分改进后的执行时间：

$$\text{总执行时间}_{\text{改进后}} = (1-\text{可改进比例}) \times \text{总执行时间}_{\text{改进前}} + \frac{\text{可改进比例} \times \text{总执行时间}_{\text{改进前}}}{\text{部件加速比}}$$

$$= \text{总执行时间}_{\text{改进前}} \times \left[(1-\text{可改进比例}) + \frac{\text{可改进比例}}{\text{部件加速比}} \right]$$

系统加速比为改进前与改进后总执行时间之比：

$$\text{系统加速比} = \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}} = \frac{1}{(1-\text{可改进比例}) + \frac{\text{可改进比例}}{\text{部件加速比}}}$$

实际上，Amdahl 定律还表达了一种性能增加的递减规则：如果仅仅对计算机中的一部分做性能改进，则改进越多，系统获得的效果越小。Amdahl 定律的一个重要推论是：如果只针对整个任务的一部分进行优化，那么所获得的加速比不大于 1/(1−可改进比例)。

从另外一个侧面来看，Amdahl 定律告诉人们如何衡量一个“好”的计算机系统：具有高性价比的计算机系统是一个带宽平衡的系统，而不是看它使用的某些部件的性能。

3．程序的局部性原理

程序的局部性原理是指：程序总是趋向于使用最近使用过的数据和指令，也就是说程序执行时所访问存储器地址分布不是随机的，而是相对地簇聚；这种簇聚包括指令和数据两部分。程序局部性包括程序的时间局部性和程序的空间局部性。程序的时间局部性是指：程序即将用到的信息很可能就是目前正在使用的信息。程序的空间局部性是指：程序即将用到的信息很可能与目前正在使用的信息在空间上相邻或者临近。

程序的局部性原理是计算机体系结构设计的基础之一。在很多地方，尤其在处理那些与存储相关的问题时，要使用这个原理。

4．CPU 的性能

为了衡量 CPU 的性能，可以将程序执行的时间进行分解。首先，将计算机系统中与实现技术和工艺有关的因素提取出来。这个因素就是计算机工作的时钟频率，单位是 MHz 或者 GHz；第二，可以测量执行程序使用的总时钟周期数。通过这两个参数我们就可以知道程序执行的 CPU 时间：

$$\text{CPU 时间}=\text{总时钟周期数}/\text{时钟频率}$$

这两个参数没有反映程序本身的特性，还需考虑程序执行过程中所处理的指令数，记为 IC。这样可以获得一个与计算机体系结构有关的参数，即指令时钟数（cycles per instruction，CPI）：

$$\text{CPI}=\text{总时钟周期数}/\text{IC}$$

程序执行的 CPU 时间就可以写成：

$$\text{总 CPU 时间}=\text{CPI}\times\text{IC}/\text{时钟频率}$$

这个公式通常称为 CPU 性能公式。它的 3 个参数反映了与体系结构相关的以下 3 种技术。

（1）时钟频率：反映了计算机实现技术、生产工艺和计算机组织。

（2）CPI：反映了计算机实现技术、计算机指令集的结构和计算机组织。

（3）IC：反映了计算机指令集的结构和编译技术。

通过改进计算机系统设计，可以相应提高这 3 个参数的指标，从而提高计算机系统的性能。从目前情况来看，提高某一个参数指标，不会明显地影响其他两个指标。这对于综合运用各种技术改进计算机系统的性能是非常有益的。

下面对 CPU 性能公式进行进一步细化。假设计算机系统有 n 种指令，其中第 i 种指令的处理时间为 CPI_i，在程序中第 i 种指令出现的次数为 IC_i，则程序执行时间为：

$$\text{CPU 时间}=\Sigma(\text{CPI}_i\times\text{IC}_i)/\text{时钟频率}$$

这个公式同时还反映了计算机系统中每条指令的性能。将上面两个公式合并起来，得到

$$\text{CPI}=\Sigma(\text{CPI}_i\times\text{IC}_i)/\text{IC}=\Sigma(\text{CPI}_i\times\text{IC}_i/\text{IC})$$

其中，（IC_i/IC）反映了第 i 种指令在程序中所占的比例。上面这些公式均称为 CPU 性能公式。

CPI 的测量比较困难，因为它依赖于处理器组织的细节，如指令流。设计者经常采用指令的平均 CPI 值，该值是通过测量流水线和 Cache，然后计算得出。

与 Amdahl 定律相比，CPU 性能公式的最大优点是它可以独立涉及计算机 CPU 性能的各个要素。为使用 CPU 性能评价公式以求得 CPU 性能，需要对公式中各独立部分的性能进行测量。开发和使用测量工具，分析测量结果，然后通过权衡各个因素对系统性能的影响，对设计进行修改，是计算机体系结构设计的主要工作。在以后的内容中将看到，这些公式中的各个部分是如何一步一步地测量，然后修改设计，从而使系统性能提高的。

例 1.1 假设考虑条件分支指令的两种不同设计方法如下：

（1）CPU_A：通过比较指令设置条件码，然后测试条件码进行分支。

（2）CPU_B：在分支指令中包括比较过程。

在两种 CPU 中，条件分支指令都占用 2 个时钟周期而所有其他指令占用 1 个时钟周期，对于 CPU_A，执行的指令中分支指令占 20%；由于每个分支指令之前都需要有比较指令，因此比较指令也占 20%。由于 CPU_A 在分支时不需要比较，因此假设它的时钟周期时间比 CPU_B 快 1.25 倍。哪一个 CPU 更快？如果 CPU_A 的时钟周期时间仅仅比 CPU_B 快 1.1 倍，哪一个 CPU 更快呢？

解：不考虑所有系统问题，所以可用 CPU 性能公式。占用 2 个时钟周期的分支指令占总指令的 20%，剩下的指令占用 1 个时钟周期。所以

$$CPI_A=0.2\times2+0.80\times1=1.2$$

则 CPU 性能为：

$$\text{总 CPU 时间}_A=IC\times1.2\times\text{时钟周期}_A$$

根据假设，有：

$$\text{时钟周期}_B=1.25\times\text{时钟周期}_A$$

在 CPU_B 中没有独立的比较指令，所以 CPU_B 的程序量为 CPU_A 的 80%，分支指令的比例为：

$$20\%/80\%\times100\%=25\%$$

这些分支指令占用 2 个时钟周期，而剩下的 75%的指令占用 1 个时钟周期，因此：

$$CPI_B=0.25\times2+0.75\times1=1.25$$

因为 CPU_B 不执行比较，故：

$$IC_B=0.8\times IC_A$$

因此 CPU_B 性能为：

$$\begin{aligned}\text{总 CPU 时间}_B&=IC_B\times CPI_B\times\text{时钟周期}_B\\&=0.8\times IC_A\times1.25\times(1.25\times\text{时钟周期}_A)\\&=1.25\times IC_A\times\text{时钟周期}_A\end{aligned}$$

在这些假设之下，尽管 CPU_B 执行指令条数较少，CPU_A 因为有着更短的时钟周期，所以比 CPU_B 快。

如果 CPU_A 的时钟周期时间仅仅比 CPU_B 快 1.1 倍，则

$$\text{时钟周期}_B=1.10\times\text{时钟周期}_A$$

CPU_B 的性能为：

$$\text{总 CPU 时间}_B = IC_B \times CPI_B \times \text{时钟周期}_B$$
$$= 0.8 \times IC_A \times 1.25 \times (1.10 \times \text{时钟周期}_A)$$
$$= 1.10 \times IC_A \times \text{时钟周期}_A$$

因此，CPU_B 由于执行更少指令条数，比 CPU_A 运行更快。

1.4 小结

本章主要内容是讨论计算机体系结构的基本概念。

首先，从存储程序计算机概念的基础出发，讲述了经典计算机体系结构概念，并进一步讨论了计算机组成和计算机实现技术，详细介绍了现代计算机系统中的层次概念。在此基础上，可以更好地理解现代计算机体系结构所研究的范围和内容。

通过存储程序计算机，了解了计算机的分代和分型，研究了计算机应用需求和实现技术等方面的发展对计算机体系结构发展的促进作用，总结了计算机体系结构的生命周期。

本章简要讨论了影响体系结构设计的成本和价格因素，这些概念会加深读者对计算机体系结构技术的理解。

促进现代计算机发展的重要手段之一就是对计算机系统中采用的技术进行定量分析。本章讨论了一些对计算机系统性能进行定量分析的技术、方法、参数和指标等，并给出了贯穿全书的一些指导计算机体系结构设计的基本原则。

通过并行性技术提高计算机系统性能是体系结构研究的主要内容之一。本章介绍了并行性技术的基本概念，这些概念是学习计算机体系结构的基础。

习题 1

1．解释下列名词。

层次结构	翻译	解释	体系结构	透明性
系列机	软件兼容	兼容机	计算机组成	计算机实现
存储程序计算机	并行性	时间重叠	资源重复	资源共享
同构型多处理机	异构型多处理机		最低耦合	松散耦合
紧密耦合	响应时间	测试程序	测试程序组件	大概率事件优先
系统加速比	Amdahl 定律	程序的局部性原理		CPI
原料成本	直接成本	毛利	折扣	标价

2．假设有一个计算机系统分为 4 级，每一级指令都比它下面一级指令在功能上强 M 倍，即一条 r+1 级指令能够完成 M 条 r 指令的工作，且一条 r+1 级指令需要 N 条 r 级指令解释。对于一段在第一级执行时间为 K 的程序，在第二、第三、第四级上的一段等效程序需要执行多少时间？

3．传统的存储程序计算机的主要特征是什么？存在的主要问题是什么？目前的计算

机系统是如何改进的？

4．对于一台 400MHz 计算机执行标准测试程序，程序中指令类型、执行数量和平均时钟周期数如表 1.8 所示。

表 1.8　标准测试程序参数

指令类型	指令执行数量	平均时钟周期数
整数	45 000	1
数据传送	75 000	2
浮点	8000	4
分支	1500	2

求该计算机的有效 CPI、MIPS 和程序执行时间。

5．假设在某程序的执行过程中，浮点操作时间占整个执行时间的10%，现希望对浮点操作加速。

（1）设对浮点操作的加速比为 S_f。画出程序总加速比 Sp 和 S_f 之间的关系曲线。

（2）请问程序的最大加速比可达多少？

6．如果某一计算任务用向量化方式求解比用标量方式求解要快 20 倍，定义可用向量方式求解部分所花费的时间占总的时间的百分比为可向量化百分比。

（1）请写出加速比与可向量化百分比之间的关系表达式。

（2）画出二者之间的关系曲线。

7．计算机系统中有 3 个部件可以改进方法，这 3 个部件的部件加速比如下：

部件加速比 $_1$=30

部件加速比 $_2$=20

部件加速比 $_3$=10

（1）如果部件 1 和部件 2 的可改进比例均为 30%，那么当部件 3 的可改进比例为多少时，系统加速比才可以达到 10？

（2）如果 3 个部件的可改进比例分别为 30%、30%和 20%，3 个部件同时改进，那么系统中不可加速部分的执行时间在总执行时间中占的比例是多少？

（3）如果相对某个测试程序 3 个部件的可改进比例分别为 30%、20%和 50%，要达到最好的改进效果，仅对一个部件改进时，要选择哪个部件？如果允许改进两个部件，又如何选择？

8．假设某应用程序中有 4 类操作，通过改进，各操作获得不同的性能提高。具体数据如表 1.9 所示。

表 1.9　操作数据

操作类型	程序中的数量（百万条指令）	改进前的执行时间（周期）	改进后的执行时间（周期）
操作 1	10	2	1
操作 2	30	20	15

续表

操作类型	程序中的数量（百万条指令）	改进前的执行时间（周期）	改进后的执行时间（周期）
操作 3	35	10	3
操作 4	15	4	1

（1）改进后，各类操作的加速比分别是多少？

（2）各类操作单独改进后，程序获得的加速比分别是多少？

（3）4 类操作均改进后，整个程序的加速比是多少？