第1章

计算机系统结构导论

自第一台电子计算机诞生以来，计算机技术一直处于发展和变革之中。回顾计算机半个多世纪的发展历程，计算机系统性能的提高，主要是依靠电子器件技术和计算机系统结构的不断发展。本章介绍了计算机系统结构的基本概念和计算机的多级层次结构；分析了计算机系统结构、计算器组成与计算机实现的含义、研究的内容以及三者之间的相互关系；讨论了计算机系统并行性发展的技术途径与分类；研究了系统结构的一般性能评价标准。

1.1 计算机系统结构的基本概念

1.1.1 计算机系统的层次结构

计算机系统(computer system)由硬件(hardware)和软件(software)组成。从计算机语言的角度，可以把计算机系统按功能划分成多级层次结构，如图 1.1 所示。

这个层次模型中的每一级都对应一个机器，其组成如图 1.2 所示。这里的“机器”只对一定的观察者存在，它的功能体现在广义语言上，对该语言提供解释手段，然后作用在信息处理或控制对象上，并从对象上获得必要的状态信息。作为某一层次的观察者，只需通过该层次的语言了解和使用计算机，而对其他层次机器是如何工作和实现功能的并不关心。

某级机器具备了上述功能，能将本级机器的语言转换为下级机器能够识别和处理的形式，就完成了本级机器的实现。层次结构中的 M0 级机器为硬联逻辑，M1 级机器由硬联逻辑实现，M2 级机器由微程序(固件)实现，M3 级至 M6 级主要由软件实现。我们将主要由软件实现的机器称为虚拟机器，以区别由硬件或固件实现的实际机器(物理机器)。

M0 级为硬联逻辑，是实现微指令本身的控制逻辑。

M1 级是微程序机器级，这级的机器语言是微指令集。程序员用微指令编写的微程序一般是直接由硬联逻辑解释实现的，M0 级和 M1 级构成机器的硬件内核。

M2 级是传统机器级，这级的机器语言是该机的指令系统。机器语言程序员用这级指令系统的机器指令集编写的程序由 M1 级的微程序进行解释。

计算机系统中也可以没有 M1 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。我们目前使用的 RISC 技术就是采用这样的设计思想，处理器的指令集全部用硬件直接实现，以提高指令的执行速度。

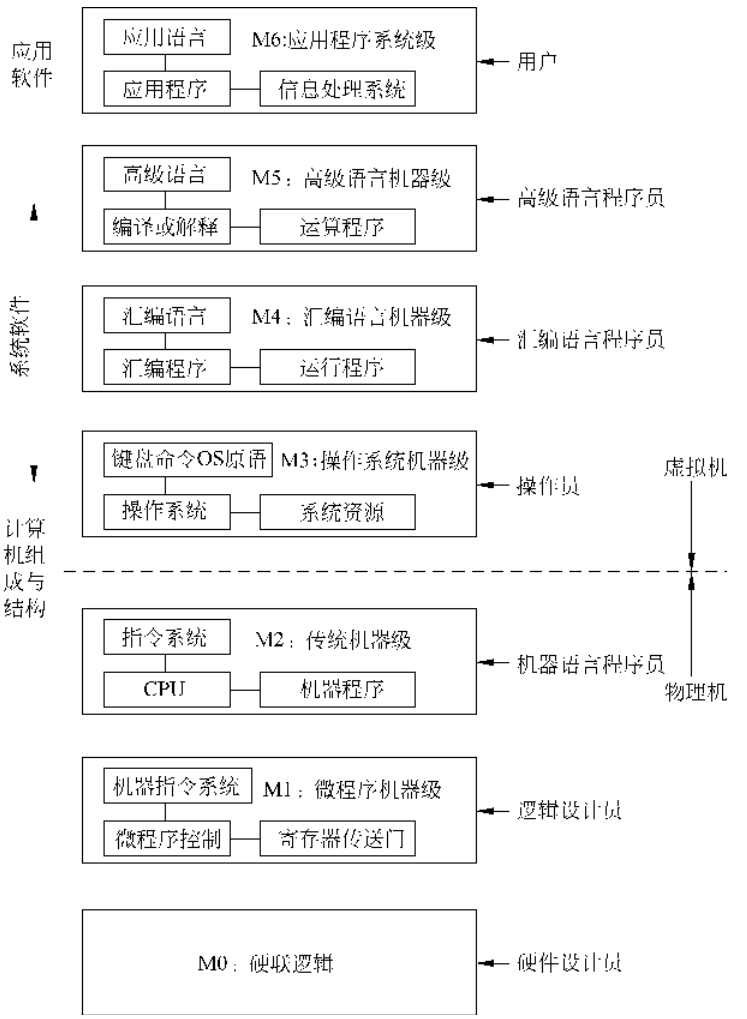

图 1.1　计算机系统层次结构

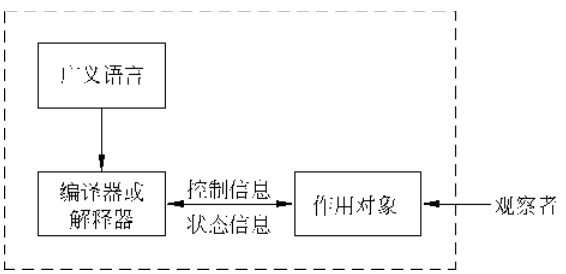

图 1.2　一级机器的组成

M3 级是操作系统机器级。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。这级的机器语言中的多数指令是传统机器的指令。此外，这一级还提供操作系统级指令，以实现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、存储体系管理、多线程管理及设备管理等。

M4 级是汇编语言机器级。这一级的机器语言是汇编语言。用汇编语言编写的程序首先被翻译成 M3 级或 M2 级语言，然后再由相应的机器进行解释。完成汇编语言程序翻译的程序称为汇编程序。

M5 级是高级语言机器级。这一级的机器语言就是各种高级语言。用高级语言编写的程序一般由编译程序翻译成 M4 级或 M3 级机器上的语言。个别的高级语言也用解释的方法实现。

M6 级是应用程序系统级。这一级的机器语言是应用语言，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。用应用语言编写的程序一般是由应用程序包翻译到 M5 级上。

把计算机系统按功能划分成多级层次结构，有利于正确理解计算机系统的工作，明确软件、硬件和固件在计算机系统中的地位和作用；有利于理解各种语言的实质及其实现；有利于探索虚拟机器新的实现方法，设计新的计算机系统。从学科领域来划分，大致可以认为：M0～M2 级属于计算机组织与结构范围，M3～M5 级属于系统软件范围，M6 级是应用软件。但各级之间可能存在某些交叉。

各虚拟机器级的实现主要靠翻译(translation)或解释(interpretation)，或者是这两者的结合。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。

软件和硬件在逻辑功能上是等效的。从原理上讲，同一逻辑功能既能用软件实现，也可以用硬件或固件实现，只是性能、价格以及实现的难易程度不同而已。一般来说，硬件实现的特点是速度快，但灵活性较差，且增加硬件成本；软件实现的特点是灵活性较好，硬件成本低，但实现速度慢。计算机系统采用何种实现方式，要从效率、速度、价格、资源状况、可靠性等多方面全盘考虑。在满足应用的前提下，软、硬件功能分配的原则主要是看能否充分利用硬件、器件技术的现状和进展，是否有利于各种组成、实现技术的采用以及是否能对各种软件的实现提供较好的硬件支持。由此对软件、硬件及固件的取舍进行综合平衡，使计算机系统达到较高的性能价格比。

从目前软、硬件技术的发展速度及实现成本上看，随着器件技术，特别是半导体集成技术的高速发展，以前由软件实现的功能，会越来越多地由硬件来实现。总体来说，软件硬化是目前计算机系统发展的主要趋势。

1.1.2　计算机系统的结构、组成与实现

1. 计算机系统结构

计算机系统结构(computer architecture)也称为计算机体系结构，从 20 世纪 70 年代开始被广泛采用。由于器件技术发展迅速，计算机硬、软件界面在动态变化，至今对计算机系

统结构定义的理解仍未统一。

1964 年 C. M. Amdahl 在介绍 IBM360 系统时提出“计算机体系结构是程序设计者所看到的计算机的属性,即概念性结构与功能特性”。然而,从计算机系统的层次结构概念出发,处于不同层次的程序设计者所看到的计算机属性显然是不一样的。在计算机技术中,对这种本来存在的事物或属性,但从某种角度看却好像不存在的现象称为透明性(transparency)。通常,在一个计算机系统中,低层机器的概念性结构和功能特性对高层机器的程序设计者往往是透明的。所谓“系统结构”是指计算机系统中各级之间界面的定义及其上、下级的功能分配。层次结构中的各级机器都有自己的系统结构。Amdahl 提出的系统结构是指传统机器级的系统结构,即机器语言程序设计者或编译程序设计者所看到的计算机物理系统的抽象或定义。在此界面之上包括计算机系统所有软件的功能,而界面之下则是计算机系统硬件和固件的功能。故这个界面实际上是计算机软件与硬件之间的分界面。在本课程中计算机系统结构研究的是对传统机器级界面的确定以及软、硬件之间的功能分配。

对于目前的通用型机器,计算机系统结构研究的内容一般包括:

(1) 数据表示。即硬件能直接识别和处理的数据类型和格式等。

(2) 寻址方式。包括最小寻址单位,寻址方式的种类、表示和地址计算等。

(3) 寄存器组织。包括操作数寄存器、变址寄存器、控制寄存器及某些专用寄存器的定义、数量和使用约定。

(4) 指令系统。包括机器指令的操作类型和格式,指令间的排序方式和控制机构等。

(5) 存储系统。包括最小编址单位、编址方式、存储容量、最大可编址空间等。

(6) 中断机构。包括中断的类型、中断分级、中断处理程序的功能和入口地址等。

(7) 机器工作状态。如管态、目态等的定义和切换。

(8) I/O 系统。包括 I/O 设备的连接方式、主机与 I/O 设备之间的数据传送方式和格式、传送的数据量以及 I/O 操作的结束与出错标志等。

(9) 信息保护。包括信息保护方式和硬件对信息保护的支持等。

这些就是机器语言程序员为了使其编写的程序能在机器上正确运行,所需要了解和遵循的计算机属性。

2. 计算机组成

计算机组成(computer organization)是计算机系统结构的逻辑实现,包括机器内部的数据流和控制流的组成以及逻辑设计等。计算机组成的任务是在计算机系统结构确定分配给硬件系统的功能及其概念结构之后,研究各组成部分的内部构造和相互之间的联系,以实现机器指令级要求的各种功能和性能。这种相互联系包括各功能部件的配置、相互连接和相互作用。各功能部件的性能参数相互匹配,是计算机组成合理的重要标志,因而相应地就有许多计算机组织方法。例如,为了使存储器的容量大、速度快,研究出了层次存储系统和虚拟存储技术。在层次存储系统中,又有高速缓存、多体交叉编址存储、多寄存器组和堆栈等技术。为了使输入/输出设备与处理机间的信息流量达到平衡,研究出了通道、外围处理机等技术。为了提高处理机速度,研究出了先行控制、流水线、多执行部件等技术。在各功能部件的内部结构研究方面,产生了许多组合逻辑、时序逻辑的高效设计方法和结构。例如,

在运算器方面，出现了多种自动调度算法和结构等。

计算机组成的设计是按希望达到的性能价格比，最佳、最合理地把各种设备和部件组成计算机，以实现所确定的计算机系统结构。一般计算器组成设计要确定的内容应包括：

(1) 数据通路的宽度。指在数据总线上能一次并行传送的信息位数。

(2) 专用部件的设置。包括设置哪些专用部件，如乘除法专用部件、浮点运算部件、字符处理部件、地址运算部件，以及每种专用部件的个数。这些都取决于机器所需达到的速度、专用部件的使用频度及允许的价格等因素。

(3) 各种操作对部件的共享程度。若部件共享程度高，则价格便宜，但会由于共享部件的分时使用而降低操作的速度；若设置多个功能部件降低共享程度，通过增加并行度以提高速度，则系统的价格会随之升高。

(4) 功能部件的并行度。如功能部件的控制和处理方式是采用顺序串行方式，还是采用重叠、流水、分布处理方式。

(5) 控制机构的组成方式。如控制机构是采用硬联控制还是微程序控制，是采用单机处理还是多机处理或功能分布处理。

(6) 缓冲和排队技术。包括在部件之间如何设置及设置多大容量的缓冲器来弥补它们的速度差异；在安排等待处理事件的顺序时，采用随机、先进先出、先进后出、优先级、循环队等方式中的哪一种。

(7) 预估、预判技术。如采用何种原则来预测未来的行为，以优化性能和优化处理。

(8) 可靠性技术。如采用怎样的冗余技术和容错技术来提高可靠性。

3. 计算机实现

计算机实现(computer implementation)是计算机组成的物理实现，包括处理机、主存等部件的物理结构，器件的集成度和速度，器件、模块、插件、底板的划分与连接，专用器件的设计，微组装技术，信号传输，电源、冷却及整机装配技术等。它着眼于器件技术和微组装技术，其中，器件技术在实现技术中起着主导作用。

4. 计算机系统结构、组成和实现三者的关系

计算机系统结构、计算机组成和计算机实现是三个互不相同的概念。计算机系统结构是计算机系统的软、硬件的界面；计算机组成是计算机系统结构的逻辑实现；计算机实现是计算机组成的物理实现。它们各自包含不同的内容，但又相互联系且相互影响。

指令系统的定义属于系统结构。指令的实现，如取指、译码、取操作数、运算、送结果等具体操作的安排及其时序属于组成。而实现这些指令功能的具体电路、器件设计及装配技术等则属于实现。

指令系统中是否包含乘、除法指令属于系统结构。而乘、除指令是用专门的乘法器、除法器实现，还是用加法器以累加配上右移或左移操作实现是属于组成。乘法器、除法器或加法器的物理实现，如器件选择及所用的微组装技术等属于实现。

在主存系统中，主存容量与编址方式(即按位、按字节还是按字访问)的确定属于系统结构。而主存的速度、逻辑结构等属于组成。至于存储器芯片选定、逻辑电路的设计、主存部件组装连接等则属于实现。

具有相同系统结构的计算机可因性价比要求不同而采用不同的组成技术。例如，具有相同指令系统的计算机，指令的读取、译码、取操作数、运算、存结果既可以采用顺序方式进行解释，也可以采用流水方式让它们在时间上重叠进行来提高速度。又如乘法指令可以利用专用乘法器来实现，也可以通过加法器重复相加、移位来实现，这主要取决于对速度的要求、乘法指令出现的频度和所采用的乘法运算方法。显然，前一种方法可以有效地提高乘法运算速度，而后一种方法则可以降低系统的价格。

同样，一种计算机组成也可以采用多种不同的计算机实现。例如，在主存器件的选择上，可以选择 TTL 型的器件，也可以采用 MOS 型器件；既可以采用单片 VLSI 集成电路，也可以采用多片 LSI 或 MSI 集成电路组成；既可以选择响应时间速度较快的芯片，也可以选择响应速度较慢的芯片。这实际上是在速度、价格等因素之间进行取舍。换句话说，采用什么样的实现技术主要考虑所要达到的性能价格比及器件技术的现状。

计算机实现是计算机系统结构和计算机组成的基础。计算机实现，尤其是器件技术的发展对计算机系统结构有着很大的影响。例如，器件的发展使系统结构由大型机下移到小型机及微机的速度加快，早期用于大型机的各种数据表示、指令系统、操作系统很快应用到了小型机以及微机上。而计算机组成也会影响计算机系统结构，目前 PC 机中的 CPU 已经普遍采用了早期在大型机中才使用的超标量技术，并引入了 VLIW 技术，有些机器还使用了超流水线技术。

系统结构的设计必须结合应用考虑，为软件和算法的实现提供更多更好的支持。同时，还要涉及可能采用和准备采用的组成技术。即计算机系统结构的设计应考虑减少对各种组成及实现技术的使用限制，在一种系统结构中，应允许有多种不同的组成和实现技术，既能方便地在低档机器上用简单、低成本的组成实现，也能在高档机器上以较高的成本、复杂的组成实现。例如，在 IBM370 系列机中，由低到高有不同档次的机器，它们的中央处理器都具有相同的基本指令系统，只是指令的分析、执行方式不同，在低档机器上用顺序方式处理，在高档机器上用并行方式处理。又如，在数据通路宽度的组成和实现上，不同档次的机器可以分别采用 8 位、16 位、32 位和 64 位。IBM370 系列机采用通道方式进行输入/输出，其组成又可以分为在低档机器中采用的结合型通道和在高档机器中采用的独立型通道。

应当看到，系统结构、组成和实现所包含的具体内容在不同时期或随不同的计算机系统会有所变化。在某些计算机系统中作为系统结构的内容，在另一些计算机系统中可能是组成和实现的内容。软件的硬化和硬件的软化都反映了这一事实。随着各种新技术的出现和发展，特别是器件技术的发展，将许多功能集成在单一芯片之中，使系统结构、组成和实现融合于一体，系统结构、组成和实现三者之间的界限越来越模糊。

计算机系统结构设计的任务是进行软、硬件的功能分配，确定传统机器级的软、硬件界面。但作为“计算机系统结构”这门学科来讲，实际上也包括组成方面的内容。因此，它研究的是软、硬件的功能分配以及如何最佳、最合理地实现分配给硬件的功能。我们也可以把着眼于软、硬件功能分配和确定程序设计者所看到的机器级界面的计算系统结构，称为从程序设计者看的计算机系统结构；把着眼于如何最佳、最合理地实现分配给硬件的功能的计算机组成称为从计算机设计者看的计算机系统结构。

1.1.3　计算机系统的特性

计算机系统在功能和结构方面都具有多层次的特性。从影响计算机系统结构的其他因素考虑，有必要讨论计算机系统的一些重要特性。

1. 计算机等级

计算机系统通常被分为巨型、大型、中型、小型、微型等若干等级。但随着技术进步，各等级的计算机性能指标都不断提高，如果按性能指标来划分计算机等级，那么一台计算机的等级将随时间而下移。各型机器的性能、价格随时间变化的趋势大致可用图 1.3 来说明，其中虚线称为等性能线。由图 1.3 可见，各型机器所具备的性能是随时间动态下移的，但价格却在相当长一段时间内基本不变，因此，有人就主张用价格来划分机器的不同等级。

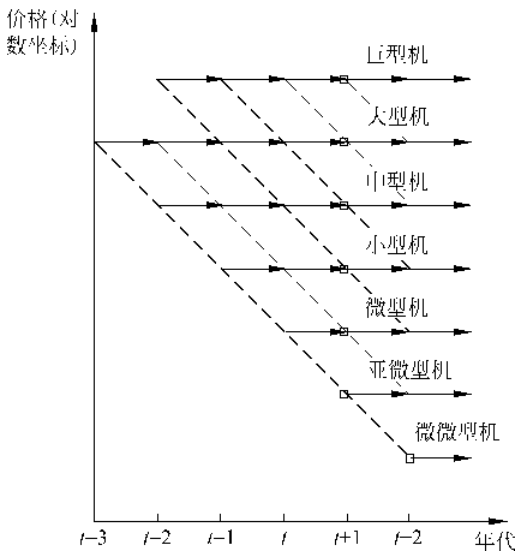

图 1.3　计算机性能下移示意图

由此可见，计算机工业在处理性能和价格的关系上可以有两种途径：一种是维持价格不变，充分利用器件技术等的进展不断提高机器的性能，即沿图 1.3 中的水平实线发展；另一种是在性能基本不变的情况下，利用器件技术等的进展不断降低机器的价格，即沿图 1.3 中虚线往下发展。基于这种思想，不同等级的计算机可采用不同的发展策略：

(1) 在同等级范围内以合理的价格获得尽可能好的性能，逐渐向高档机发展，称为最佳性能价格比设计。

(2) 维持一定适用的基本性能而争取最低价格，称为最低价格设计。其结果往往是从低档向下分化出新的计算机等级。

(3) 以获取最高性能为主要目标而不惜增加价格，称为最高性能设计。其结果是产生当时最高等级的计算机。

第一类设计主要针对大、中型计算机用户的需要，设计生产出性能价格比更好的中型计算机和超小型计算机；第二类设计以普及应用计算机为目标，设计生产数量众多的微、小型

计算机；第三类设计只满足少数用户的特殊需要。

从系统结构的观点来看，各型计算机的性能随时间下移，实质上是在低档（型）机上引用甚至照搬高档（型）机的系统结构和组成。这种低档机承袭高档机系统结构的状况正符合小型机和微型机的设计原则，即充分发挥器件技术的进步，以尽可能低的价格在低档机上实现高档机已有的结构和组成。不花很大力量专门去研究和采用新的系统结构和组成技术，这将有利于计算机工业的快速发展和计算机应用的广泛普及。从计算机技术发展过程可以看到，系统结构和组成下移的速度越来越快，例如，超高速缓冲存储器和虚拟存储器从大型机下移到小型机所花的时间不到六年，巨型阵列机问世不过七年，小型机上就有了可扩充的高速阵列处理部件。

2. 系列机

所谓系列机的概念，是在软、硬件界面上设计好一种系统结构，然后软件设计者按此系统结构设计系统的软件；硬件设计者根据机器速度、性能、价格的不同，选择不同的器件，采用不同的硬件技术和组成与实现技术，研制并提供不同档次的机器。在系列机上必须保证用户看到一致的机器属性，例如，IBM AS400 系列，数据总线有 16、32、64 位的区别，但数据表示方式一致。

系列机之间必须保持软件兼容（software compatibility）。系列机软件兼容是指同一个软件（目标程序）可以不加修改地运行于系统结构相同的各档次机器中，而且所得结果一致。软件兼容包括向上兼容和向下兼容：向上兼容是指在低档机器上编写的软件，不加修改就可以运行于高档机器上；向下兼容则相反。一般不使用向下兼容方式。软件兼容还有向前兼容和向后兼容之分：向后兼容是指在某个时期投入市场的该型号机器上编写的软件，不加修改就可以运行于在它之后投入市场的机器上；向前兼容则相反。对系列机而言，必须保证做到软件向后兼容，同时力争做到软件向上兼容。

为了减少编写软件的工作量，降低软件开发成本，延长成熟软件的生命周期，应该注意在研究新的系统结构时，解决好软件的可移植性（portability）问题。所谓软件的可移植性，是指软件不用修改或只需少量加工就能由一台机器换到另一台机器上运行，即同一软件用于不同的环境。系列机软件兼容的特性，能够很好地解决同一系列结构内的软件可移植问题，成为当前计算机设计普遍采用的技术。

系列机为了保证软件的兼容，要求系统结构的一致，这无疑又成为妨碍计算机系统结构发展的重要因素。实际上，为适应性能不断提高和应用领域不断扩大的需要，应允许系列机中后面推出的各档机的系统结构有所发展和变化。但是，这种改变只应该是为提高机器总的性能所做的必要扩充，而且主要是为改进系统软件的性能来修改系统软件（如编译系统），尽可能不影响高级语言应用软件的兼容，尤其是不允许缩小或删改运行已有软件的那部分指令和结构。例如，在后推出的各档机器上，可以为提高编译效率和运算速度增加浮点运算指令；为满足事务处理增加事务处理指令及其所需功能；为提高操作系统的效率和质量增加操作系统专用指令和硬件等。因此，可以对系列机的软件向下兼容和向前兼容不做要求，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），但向后兼容是肯定要做到的。

不同公司厂家生产的具有相同系统结构的计算机称为兼容机（compatible machine）。它的思想与系列机的思想是一致的。兼容机还可以对原有的系统结构进行某种扩充，使

之具有更强的功能，例如，长城 0520 与 IBM PC 兼容，但有较强的汉字处理能力。

3. 模拟与仿真

系列机解决了在具有相同系统结构的各种机器之间实现软件移植的问题。为了实现软件在不同系统结构的机器之间移植，就必须做到能在一种机器的系统结构上实现另一种机器的系统结构。从计算机系统结构的层次模型来看，就是要在一种机器的系统结构上实现另一种机器的指令系统。一般可采用模拟方法或仿真方法。

要求在 A 机器上用虚拟机的概念实现 B 机器的指令系统，如图 1.4 所示，即 B 机器的每一条机器指令由 A 机器的一段机器语言程序去解释执行，从而可使 B 机器的程序能在 A 机器上运行。这种用机器语言程序解释实现软件移植的方法称为模拟(simulation)，被模拟的 B 机器称为虚拟机(virtual machine)，A 机器称为宿主机(host machine)。

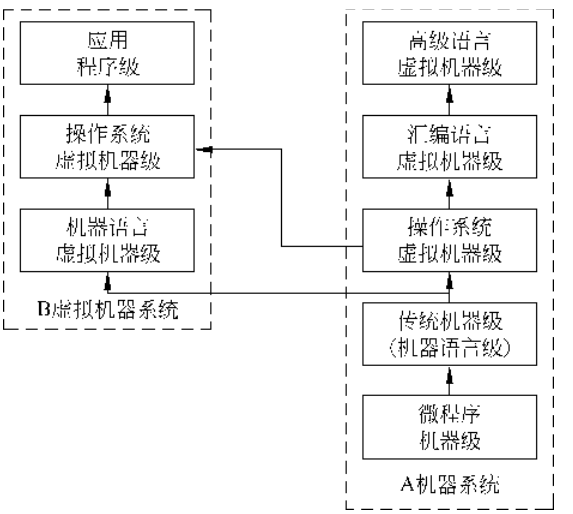

图 1.4　用模拟方法实现软件的移植

若 A 机器采用微程序控制，则被模拟的 B 机器的每条机器指令需要通过二重解释。显然，如果直接用 A 机器的微程序去解释 B 机器的机器指令就会加快解释过程，如图 1.5 所示。这种用微程序直接解释另一种机器指令系统实现软件移植的方法称为仿真(emulation)。进行仿真工作的 A 机器称为宿主机，被仿真的 B 机器称为目标机(target machine)，为仿真所编写的解释微程序称为仿真微程序。仿真与模拟的主要区别在于解释所用的语言：仿真用微程序解释，其解释程序在微程序存储器中，而模拟用机器语言解释，其解释程序在主存储器中。

为了使虚拟机的应用软件能在宿主机上运行，除了模拟虚拟机的机器语言外，还得模拟其存储体系、I/O 系统、控制台的操作，以及形成虚拟机的操作系统。让虚拟机的操作系统受宿主机操作系统的控制，如图 1.4 所示那样，实际上是把虚拟机操作系统作为宿主机的应用程序来看待。所有为模拟所编写的解释程序统称为模拟程序。

模拟程序的编写是非常复杂和费时的。由于虚拟机的每条机器指令不能直接被宿主机的硬件执行，而是由多条宿主机机器指令构成的解释程序来解释，因此，模拟的运行速度显著降低。

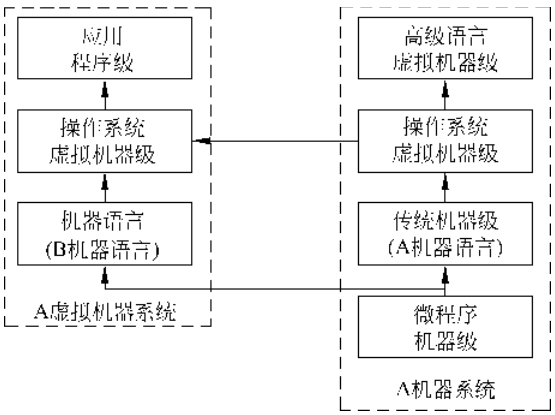

图 1.5　用仿真方法实现软件的移植

用仿真方法可以提高被移植软件的运行速度，但由于微程序机器级结构深度依赖于机器的系统结构，所以当两种机器结构差别较大时，就很难依靠仿真来实现软件移植，特别是当其 I/O 系统结构差别较大时更是如此。因此，在实际应用中，不同系列机之间的软件移植往往通过仿真和模拟两种方法并用来实现。对于使用频繁而易于仿真的机器指令，尽可能采用仿真方法以提高速度，对于使用较少且用微程序仿真难以实现的某些指令及 I/O 系统等操作则宜采用模拟方法。

1.2　计算机系统结构中并行性的发展

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。本节首先对冯·诺依曼系统结构进行分析，然后叙述体系结构中的并行性概念，再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，得到对计算机系统结构中并行性发展的全面了解和认识。

1.2.1　冯·诺依曼型计算机系统结构

冯·诺依曼型计算机由运算器、控制器、存储器、输入设备和输出设备五个部分组成，在结构上有以下特点：

(1) 机器以运算器为中心，I/O 设备与存储器之间的数据传送都要经过运算器。各部件的操作及相互之间的联系都由控制器集中控制。

(2) 采用存储程序的思想。机器各部分的操作是在事先存放于存储器中的程序的控制之下顺序执行一条条指令来完成的，而且将存储器中的指令和数据同等对待，都可以不加区别地送到运算器，因此，由指令组成的程序可以在运行过程中被修改。

(3) 存储器按地址访问。它是一个顺序、线性编址的一维空间，每个存储单元的位数是固定的。

(4) 由指令计数器指明要执行的指令在存储器中的地址。可以根据运算结果改变指令