“十二五”普通高等教育本科国家级规划教材

21世纪大学本科
计算机专业系列教材

张晨曦 骆焦煌
刘 依 沈 立 编著

计算机系统结构学习指导与题解（第2版）

http://www.tup.com.cn

- 根据教育部“高等学校计算机科学与技术专业规范”组织编写
- 与美国 ACM 和 IEEE *Computing Curricula 2005* 同步

清华大学出版社

“十二五”普通高等教育本科国家级规划教材

21世纪大学本科计算机专业系列教材

计算机系统结构学习指导与题解

（第2版）

张晨曦 骆焦煌 刘依 沈立 编著

清华大学出版社

北京

内容简介

本书是普通高等教育“十二五”国家级规划教材《计算机系统结构》(套书)中的一册,是《计算机系统结构教程(第2版)》(清华大学出版社)的配套教材。全书共分为14章,内容覆盖面广,包括计算机系统结构的基础知识、指令系统的设计、流水线技术、向量处理机、指令级并行性及其开发——硬件方法、指令级并行的开发——软件方法、存储系统、输入输出系统、互连网络、多处理机、多核架构与编程、机群系统、阵列处理机、数据流计算机。每一章都由4节组成,分别是基本要求与难点、知识要点、习题以及题解。知识要点给出了各章的精华和要点。习题的类型有概念题、选择题、填空题、简答题和应用题。对于应用题,书中给出了详细的求解过程。

本书概念清晰,重点难点突出,覆盖面广,题型多样,是一本很有用的学习辅导书。本书可作为计算机系统结构课程(上课或自学)的学习辅导书,也可作为计算机专业硕士研究生入学考试的复习指导书。

图书在版编目(CIP)数据

计算机系统结构学习指导与题解/张晨曦,骆焦煌等编著. —2版. —北京:清华大学出版社,2018
(21世纪大学本科计算机专业系列教材)
ISBN 978-7-302-49605-2

Ⅰ. ①计… Ⅱ. ①张… ②骆… Ⅲ. ①计算机体系结构—高等学校—教学参考资料 Ⅳ. ①TP303

中国版本图书馆CIP数据核字(2018)第028846号

责任编辑:魏江江　薛　阳
封面设计:何凤霞
责任校对:梁　毅
责任印制:丛怀宇

出版发行:清华大学出版社
网　　址:http://www.tup.com.cn, http://www.wqbook.com
地　　址:北京清华大学学研大厦A座　　**邮　　编**:100084
社 总 机:010-62770175　　**邮　　购**:010-62786544
投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈:010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载:http://www.tup.com.cn,010-62795954
印 装 者:三河市君旺印务有限公司
经　　销:全国新华书店
开　　本:185mm×260mm　　**印　　张**:19　　**字　　数**:464千字
版　　次:2009年10月第1版　2018年8月第2版　　**印　　次**:2018年8月第1次印刷
定　　价:49.00元

产品编号:057275-01

前　言

本书是普通高等教育“十二五”国家级规划教材《计算机系统结构》(套书)中的一册,是《计算机系统结构教程(第2版)》(清华大学出版社)的配套教材。

计算机系统结构是计算机专业及相关专业的一门重要的专业课程。本书是专门为该课程编写的辅导书,既可作为该课程的教学参考书,也可作为自学该课程的辅导书,还可作为计算机专业硕士研究生入学考试的复习指导书。

全书共有14章,各章内容如下。

第1章讲述计算机系统结构的概念以及系统结构和并行性的发展,并介绍定量分析基础。

第2章是指令系统的设计,讲述计算机指令系统设计的各个方面。

第3章是流水线技术,讲述流水线的基本概念、分类和性能分析、非线性流水线的调度,介绍流水线中的相关和冲突问题及其解决方法,并讨论流水线的实现。

第4章是向量处理机,讲述向量处理机的结构、特点和性能评价。

第5章和第6章分别讲述如何用硬件和软件的方法来开发指令级并行。

第7章是存储系统,讲述Cache的基本知识以及提高Cache性能的方法,并对并行主存和虚拟存储器做了简要的讨论。

第8章是输入输出系统,讲述总线及其与CPU/存储器的连接、通道处理机及其流量分析,并详细论述了廉价磁盘冗余阵列RAID。

第9章是互连网络,讲述互连函数、互连网络的特性参数、静态互连网络、动态互连网络以及消息传递机制等。

第10章是多处理机,讲述对称式共享存储器系统结构、分布式共享存储器系统结构、多Cache一致性、同步、同时多线程以及MPP。

第11章是多核架构与编程,讲述Intel x86和ARM11 MPCore多核架构以及基于多核的并行程序设计。

第12章是机群系统,讲述机群的结构、软件模型以及机群的分类。

第13章是阵列处理机,讲述阵列处理机的操作模型、结构和特点以及并行算法。

第14章是数据流计算机,讲述数据流计算机模型、数据流程序图和数据流语言、静态数据流计算机结构以及动态数据流计算机结构。

每一章都由4节组成,分别是基本要求与难点、知识要点、习题以及题解。知识要点给出了各章的精华和要点。习题的类型有概念题、选择题、填空题、简答题和应用题。对于应用题,书中给出了详细的求解过程。

本书概念清晰,重点难点突出,覆盖面广,题型多样,是一本很有用的学习辅导书。

由于作者水平有限,书中难免有疏漏和不妥之处,敬请读者批评指正。

作　者

2017年7月

目　录

第1章　计算机系统结构的基础知识

1.1　基本要求与难点

1.1.1　基本要求

(1) 掌握计算机系统结构的基本概念，如：计算机系统的多级层次结构，虚拟机，计算机系统结构的定义，透明性，系列机等。

(2) 理解计算机系统的多级层次结构。

(3) 掌握计算机系统结构、计算机组成和计算机实现的定义，了解其关系。

(4) 了解计算机系统结构的分类方法。

(5) 掌握计算机系统设计的定量原理，能熟练地应用 Amdahl 定律和 CPU 性能公式进行定量分析。

(6) 理解进行计算机系统设计的 3 种主要方法，重点掌握“由中间开始”的设计方法。

(7) 理解进行计算机系统性能评测的基准测试程序法，了解如何进行计算机系统性能的比较。

(8) 理解冯·诺依曼结构的特点及其改进。

(9) 掌握实现软件移植的 3 种方法，理解系列机的概念及其根本特征。

(10) 理解并行性的定义以及并行性的不同等级，掌握提高并行性的 3 种技术途径。

(11) 了解单机系统中和多机系统中并行性的发展。

1.1.2　难点

(1) 计算机系统结构、计算机组成和计算机实现三者的定义与关系。如何判断某种技术是属于哪个方面的？如何判断其透明性？

(2) Amdahl 定律和 CPU 性能公式，如何应用它们进行定量分析？

1.2　知识要点

1.2.1　计算机系统结构的基本概念

1. 计算机系统的层次结构

可以把计算机系统按功能划分成**多级层次结构**：微程序机器级，传统机器语言机器级，

操作系统机器级,汇编语言机器级,高级语言机器级,应用语言机器级等。在这6级层次中,最下面的两级机器是用硬件/固件实现的,称为**物理机**;而上面4层一般是由软件实现的,称为**虚拟机**。

各机器级的实现主要靠翻译或解释,或两者的结合。一般来说,上述6级层次中的下面3级是用解释实现,而上面3级则是用翻译的方法实现。**翻译**是先用转换程序把高一级机器上的程序转换为低一级机器上等效的程序,然后再在这低一级机器上运行,实现程序的功能。**解释**则是对于高一级机器上的程序中的每一条语句或指令,都是转去执行低一级机器上的一段等效程序,执行完后,再去高一级机器取下一条语句或指令,再进行解释执行,如此反复,直到解释执行完整个程序。

2. 计算机系统结构的定义

计算机系统结构(Computer Architecture)的经典定义是1964年Amdahl在介绍IBM360系统时提出的:计算机系统结构是指传统机器程序员所看到的计算机属性,即概念性结构与功能特性。计算机系统结构的实质是确定计算机系统中软硬件的界面,界面之上是软件实现的功能,界面之下是硬件和固件实现的功能。

传统机器级所存在的差别对于高级语言程序员来讲是"看不见"的。在计算机技术中,把这种本来存在的事物或属性,但从某种角度看又好像不存在的概念称为**透明性**。

在J. L. Hennessy和D. A. Patterson编写的*Computer Architecture: A Quantitative Approach*一书中,把系统结构定义为囊括计算机设计的3个方面:指令系统结构,组成,硬件。我们不妨将之理解为广义的系统结构定义。

3. 计算机组成和计算机实现

计算机组成指的是计算机系统结构的逻辑实现,包含物理机器级中的数据流和控制流的组成以及逻辑设计等。它着眼于物理机器级内各事件的排序方式与控制方式、各部件的功能以及各部件之间的联系。

计算机实现指的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,模块、插件、底板的划分与连接,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主导作用。

具有相同系统结构的计算机因为速度、价格等方面要求的不同,可以采用不同的计算机组成。而同一种计算机组成又可以采用多种不同的计算机实现。系列机的出现充分反映了这一点。系列机是指由同一厂家生产的具有相同系统结构但具有不同组成和实现的一系列不同型号的计算机。

4. 计算机系统结构的分类

1) Flynn分类法

Flynn分类法是按照指令流和数据流的多倍性进行分类。Flynn分类法中定义:

指令流:计算机执行的指令序列。

数据流:由指令流调用的数据序列。

多倍性:在系统最受限的部件上,同时处于同一执行阶段的指令或数据的最大数目。

Flynn分类法把计算机系统的结构分为以下4类。

(1) 单指令流单数据流(Single Instruction stream Single Data stream,SISD);

(2) 单指令流多数据流(Single Instruction stream Multiple Data stream,SIMD);

(3) 多指令流单数据流(Multiple Instruction stream Single Data stream,MISD);

(4) 多指令流多数据流(Multiple Instruction stream Multiple Data stream,MIMD)。

SISD 是传统的顺序处理计算机;SIMD 以阵列处理机为代表;MISD 只是一种人为的划分,目前没有实际的机器;多处理机属于 MIMD 结构。

2) 冯氏分类法

冯氏分类法是用系统的最大并行度对计算机进行分类。最大并行度 P_m 定义为:计算机系统在单位时间内能够处理的最大的二进制位数。

按照这种分类法,可以把计算机分成以下 4 类具有不同最大并行度的计算机系统结构:字串位串,字串位并,字并位串,字并位并。

3) Handler 分类法

这种分类方法把计算机的硬件结构分成 3 个层次,并考虑它们的可并行-流水处理程度。

$$t(\text{系统型号}) = (k \times k', d \times d', w \times w')$$

这 3 个层次是:

(1) 程序控制部件(PCU)的个数 k;

(2) 算术逻辑部件(ALU)或处理部件(PE)的个数 d;

(3) 每个算术逻辑部件包含基本逻辑线路(ELC)的套数 w。

1.2.2　计算机系统的设计

1. 计算机系统设计的定量原理

1) 以经常性事件为重点

这是计算机设计中最重要、最广泛采用的设计原则。它是指要对经常发生的情况采用优化方法的原则,因为这样能得到更多的总体上的改进。这里的优化是指分配更多的资源、达到更高的性能等。

2) Amdahl 定律

Amdahl 定律告诉我们:当对一个系统中的某个部件进行改进后,所能获得的整个系统性能的提高,受限于该部件的执行时间占总执行时间的百分比。它可以用来具体地计算:当对计算机系统中的某个部分进行改进后,所能获得的加速比的大小。

这个加速比的大小与两个因素有关。一个是可改进比例,记为 Fe。另一个是可改进部件改进以后性能提高的倍数,简称部件加速比,记为 Se。改进后程序的总执行时间为:

$$T_n = T_0\left(1 - \mathrm{Fe} + \frac{\mathrm{Fe}}{\mathrm{Se}}\right)$$

其中,T_0 为改进前整个程序的执行时间,1−Fe 为不可改进比例。

改进后,整个系统的加速比为:

$$S_n = \frac{T_0}{T_n} = \frac{1}{(1 - \mathrm{Fe}) + \dfrac{\mathrm{Fe}}{\mathrm{Se}}}$$

当 $Se \to \infty$ 时，则 $S_n = 1/(1-Fe)$。即如果只针对整个任务的一部分进行改进和优化，那么所获得的加速比不超过 $1/(1-Fe)$。

3) CPU 性能公式

执行一个程序所需的 CPU 时间可以按下式计算：

$$\text{CPU 时间} = \text{IC} \times \text{CPI} \times \text{时钟周期时间}$$

其中，IC 为所执行的指令条数。CPI(Cycles Per Instruction)为每条指令的平均时钟周期数，它可以根据下式计算得出：

$$\text{CPI} = \text{执行程序所需的时钟周期数} / \text{所执行的指令条数}$$

CPU 的性能取决于以下 3 个参数。

(1) 时钟周期时间：取决于硬件实现技术和计算机组成；

(2) CPI：取决于计算机组成和指令系统的结构；

(3) IC：取决于指令系统的结构和编译技术。

CPU 设计中还经常用到下面计算 CPU 时钟周期总数的方法：

$$\text{CPU 时钟} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{IC}_i)$$

其中，n 为指令的种数，IC_i 为第 i 种指令出现的次数，CPI_i 为第 i 种指令所需的平均时钟周期数。这时的 CPU 性能公式为：

$$\text{CPI} = \frac{\text{时钟周期数}}{\text{IC}} = \frac{\sum_{i=1}^{n} (\text{CPI}_i \times \text{IC}_i)}{\text{IC}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \frac{\text{IC}_i}{\text{CI}} \right)$$

$$\text{CPU 时间} = \text{CPU 时钟周期数} \times \text{时钟周期时间} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{IC}_i) \times \text{时钟周期时间}$$

其中，(IC_i/IC) 反映了第 i 种指令在所执行的指令总数中所占的比例。

4) 程序的局部性原理

程序的局部性原理是指：程序执行时所访问的存储器地址不是随机分布的，而是相对地簇聚。现在常用的一个经验规则是：程序执行时间的 90% 都是在执行程序中 10% 的代码。

局部性包括时间局部性和空间局部性。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。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。

2. 计算机系统设计者的主要任务

计算机系统设计者的任务包括指令系统的设计、数据表示的设计、功能的组织、逻辑设计以及其物理实现等。设计一个计算机系统大致要完成以下 3 个方面的工作。

1) 确定用户对计算机系统的功能、价格和性能的要求

总的来说，计算机系统设计者的目标是设计出能满足用户的功能需求、有较长的生命周期且又具有很高的性能价格比的系统。要考虑以下具体的功能需求。

(1) 应用领域。首先要考虑的是：是专用还是通用？是面向科学计算还是面向商用处理？如果是专用，就需要对应用领域进行深入的研究，以确定什么样的计算任务是关键的，

具有什么计算特点等，然后对其进行优化设计。

(2) 软件兼容。软件兼容是指一台计算机上的程序不加修改就可以搬到另一台计算机上正常运行。兼容性方面的考虑对于设计新的计算机系统有很大的影响。

(3) 操作系统需求。

(4) 标准。

2) 软硬件功能分配

软件和硬件在实现功能上是等价的。进行优化设计必须考虑软硬件功能的合理分配。对于任何一种功能来说，用软件实现的优点是设计相对容易、修改简单，而且可以减少硬件成本。但其缺点是所实现的功能的速度较慢。用硬件实现的优点是速度快、性能高，但它修改困难，灵活性差。所以优化设计时要在软硬件之间进行折中和取舍。

3) 设计出生命周期长的系统结构

一个成功的系统结构应该有较长的生命周期，它应该能经得住软、硬件技术的发展和应用的变化。因此，设计者要特别注意计算机应用和计算机技术的发展趋势，设计出具有一定前瞻性的系统结构，以使得它具有较长的生命周期。

3. 计算机系统设计的主要方法

从多级层次结构出发，计算机系统可以有由上往下、由下往上和从中间开始 3 种不同的设计方法。

1) “由上往下”设计

由上往下设计是先从层次结构中的最上面一级开始，首先确定应用级机器的属性，然后再逐级往下设计，每级都考虑怎样优化上一级的实现。这种方法很适合于专用机的设计，而不适合通用机的设计。

2) “由下往上”设计

这种方法是从层次结构的最下面一级开始，逐层往上设计各层的机器。在硬件技术飞速发展、而软件技术发展相对缓慢的今天，这种设计方法已经难以适应计算机系统的设计要求，很少被采用了。

3) “由中间开始”设计

软、硬件设计分离和脱节是上述“由上往下”和“由下往上”设计方法的主要缺点。要解决这个问题，就必须综合考虑软、硬件的分工，从中间开始设计。这里“中间”是指层次结构中的软硬件的交界面，即在传统机器级与操作系统机器级之间。采用这种方法时，首先要进行软、硬件功能分配，确定好这个界面。然后从这个界面开始，软件设计者开始往上设计操作系统、汇编、编译系统等，硬件设计者开始往下设计传统机器级、微程序机器级、数字逻辑级等。软件和硬件并行设计可以缩短设计周期，设计过程中可以交流协调，是一种交互式的、很好的设计方法。

1.2.3　计算机系统的性能评测

1. 执行时间和吞吐率

如何评测一台计算机的性能，与测试者看问题的角度有关。对于桌面台式计算机来说，

用户关心的是执行单个程序的时间,而对于数据处理中心的管理员来说,他所关心的则是吞吐率。

在比较不同的设计方案时,经常需要对两台计算机的性能进行比较。假设这两台计算机为 X 和 Y,通常"X 的性能是 Y 的 n 倍"是指:

$$n = \frac{\text{执行时间 } Y}{\text{执行时间 } X} = \frac{\dfrac{1}{\text{性能}_Y}}{\dfrac{1}{\text{性能}_X}} = \frac{\text{性能}_X}{\text{性能}_Y}$$

目前广泛采用的一致和可靠的性能评价方法,是使用真实程序的执行时间来衡量。

我们常用"CPU 时间"来计算程序的执行时间,"CPU 时间"是指 CPU 执行所给定的程序所花费的时间,不包含 I/O 等待时间以及运行其他程序的时间。CPU 时间还可细分为用户 CPU 时间及系统 CPU 时间,前者表示用户程序所耗费的 CPU 时间,后者表示用户程序执行期间操作系统所耗费的 CPU 时间。

2. 基准测试程序

用于测试和比较性能的基准测试程序的最佳选择是真实应用程序。

为了能比较全面地反映计算机在各个方面的处理性能,通常采用整套测试程序。这组程序称为基准测试程序套件,它是由各种不同的真实应用程序构成的。基准测试程序套件的目标是尽可能全面地反映两台计算机的相对性能。

目前最成功和最常见的测试程序套件是 SPEC 系列,它是由美国的标准性能测试公司(Standard Performance Evaluation Corporation)于 20 世纪 80 年代末创建的。桌面计算机的基准测试程序套件可以分为两大类:处理器性能测试程序和图形性能测试程序。SPEC 最早创建的 SPEC89 是用于测试处理器性能的。SPEC89 后来演化出了 4 个版本:SPEC92、SPEC95、SPEC2000 和 SPEC CPU2006。

3. 性能比较

为了能更好地比较不同计算机的性能,可以采用以下 3 种方法。

1) 总执行时间

我们可以直接用计算机执行所有测试程序的总时间来进行比较,也可以采用平均执行时间来代替总执行时间。平均执行时间是各测试程序执行时间的算术平均值,即:

$$S_m = \frac{1}{n}\sum_{i=1}^{n} T_i$$

其中,T_i 是第 i 个测试程序的执行时间,n 是测试程序组中程序的个数。

如果各程序在测试程序组中所占的比重不同,就可以用加权执行时间来比较。加权执行时间是各测试程序执行时间的加权平均值,即:

$$A_m = \sum_{i=1}^{n} W_i \cdot T_i \tag{1.1}$$

其中,W_i 是第 i 个测试程序在测试程序组中所占的比重,$\sum_{i=1}^{n} W_i = 1$。T_i 是该程序的执行时间。

2）调和平均值法

如果性能是用速度（如 MFLOPS）表示，则可以采用调和平均值法进行比较。其公式为：

$$H_m = \frac{n}{\sum_{i=1}^{n} \frac{1}{R_i}} = \frac{n}{\sum_{i=1}^{n} T_i} \tag{1.2}$$

其中，R_i 表示执行第 i 个程序的速度，$R_i = 1/T_i$，T_i 为第 i 个程序的执行时间。

如果考虑工作负荷中各程序不会以相等比例出现，则可以使用加权调和平均值公式：

$$H_m = \left(\sum_{i=1}^{n} \frac{W_i}{R_i} \right)^{-1} = \left(\sum_{i=1}^{n} W_i T_i \right)^{-1} \tag{1.3}$$

3）几何平均值法

几何平均值法的基本思想来源于性能规格化的方法，即以某台计算机的性能作为参考标准，其他计算机性能则除以该参考标准而获得一个比值。其公式为：

$$G_m = \sqrt[n]{\prod_{i=1}^{n} R_i} = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{T_i}} \tag{1.4}$$

式中，R_i 表示执行第 i 个程序的速度，$R_i = 1/T_i$，$\prod$ 为连乘符号。

如果考虑工作负荷中各程序不会以相等比例出现，则可以使用加权几何平均值公式：

$$G_m = \prod_{i=1}^{n} (R_i)^{W_i} = (R_1)^{W_1} \times (R_2)^{W_2} \times \cdots \times (R_n)^{W_n} \tag{1.5}$$

1.2.4　计算机系统结构的发展

1. 冯·诺依曼结构及其改进

最早的存储程序式计算机是美国数学家冯·诺依曼（von Neumann）等人于 1946 年总结并提出来的，它由运算器、控制器、存储器、输入设备和输出设备 5 部分构成。我们经常称之为冯·诺依曼结构计算机。虽然与冯·诺依曼结构相比，现代的计算机系统结构已经发生了很大变化，但就其结构原理来说，占主流地位的仍是改进了的冯·诺依曼结构计算机。

冯·诺依曼结构的主要特点如下。

（1）计算机以运算器为中心，采用集中控制。

（2）在存储器中，指令和数据同等对待。

（3）存储器是按地址访问、按顺序线性编址的一维结构。

（4）指令的执行是顺序的，即一般是按照指令在存储器中存放的顺序执行。

（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，地址码指明操作数地址和存放运算结果的地址。操作数的类型由操作码决定。

（6）指令和数据均以二进制编码表示，采用二进制运算。

后来的计算机针对冯·诺依曼结构的不足之处进行了不断的改进，在系统结构方面有了很大的进展。主要包括以下几个方面。

1）对输入/输出方式的改进

人们先后提出了多种输入/输出方式，包括程序中断、DMA、通道、外围处理机等。这是

把越来越多的输入/输出管理工作从CPU中分离出来,“下放”给新设置的硬件去完成。

2) 采用并行处理技术

在不同的级别采用并行技术,例如微操作级、指令级、线程级、进程级、任务级等。先后出现了向量计算机、阵列处理机、多处理机、MPP(大规模并行处理机)等各种并行处理计算机。

3) 存储器组织结构的发展

在CPU中设置了通用寄存器组,并在CPU和主存之间设置了高速缓冲存储器Cache。

4) 指令系统的发展

发展方向有两个,一个是朝**复杂指令集计算机**(Complex Instruction Set Computer,CISC)的方向发展,把越来越多的功能交由硬件实现。另一个方向是朝精简指令系统的方向发展,只设置使用频度高、功能简单的指令。1979年,D. A. Patterson等人提出了**精减指令集计算机**(Reduced Instruction Set Computer,RISC)的思想。

2. 软件对系统结构的影响

软件对系统结构有多方面的影响。下面只讨论系统结构设计要注意解决的软件可移植性问题。**可移植性**是指一个软件可以不经修改或者只需少量修改就可以由一台计算机移植到另一台计算机上运行。差别只是执行时间的不同。在这种情况下,我们称这两台计算机是**软件兼容**的。实现可移植性的常用方法有3种:统一高级语言,系列机,模拟与仿真。

1) 统一高级语言

如果各计算机能采用同一种高级语言,那么用这种语言编写的应用软件和系统软件的可移植问题就解决了。然而,到目前为止,还没有一种高级语言对各种应用是真正通用的。

2) 系列机

系列机能较好地解决软件开发要求系统结构相对稳定与器件、硬件技术迅速发展的矛盾。系列机的软件兼容有4种:向上兼容,向下兼容,向前兼容,向后兼容。**向上(下)兼容**指的是按某档计算机编制的程序,不加修改就能运行于比它高(低)档的计算机。**向后(前)兼容**是指按某个时期投入市场的某种型号计算机编制的程序,不加修改地就能运行于在它之后(前)投入市场的计算机。向后兼容是肯定要做到的,它是系列机的根本特征。

兼容机:由不同公司厂家生产的具有相同系统结构的计算机。

3) 模拟和仿真

模拟和仿真是实现软件(二进制代码)可移植性的两种常用方法。

模拟是指用软件的方法在一台现有的计算机(称为宿主机Host)上实现另一台计算机(称为虚拟机)的指令系统。通常用解释的方法来实现。除了模拟虚拟机的指令系统外,还要模拟其存储系统、I/O系统、操作系统等。这种方法的缺点是运行速度较慢,性能较差。

仿真是指用一台现有计算机(称为宿主机)上的微程序去解释实现另一台计算机(称为目标机)的指令系统。这个微程序称为仿真微程序。

仿真和模拟的主要区别在于解释执行所用的语言。仿真是用微程序解释执行,而模拟则是用机器语言程序解释执行。因此仿真的运行速度比模拟方法的快,但仿真只能在系统结构差距不大的计算机之间使用。为了取长补短,可以将这两种方法混合使用。