

21 世纪 高等学校本科系列教材

总主编 吴中福

计算机系统结构

(20)

主 编 李恒甫



重庆大学出版社

计算机系统结构

主 编 李恒甫

重庆大学出版社

内容简介

计算机系统结构是计算机科学与技术专业的一门专业课程。本书把巨大中小微型计算机系统结构结合在一起,系统介绍计算机系统结构的基本概念、基本原理、基本结构、先进的系统结构技术,注重实际,加强基础,以提高对计算机总体结构的分析能力。

全书内容新颖、适应性强,主要内容分8章,包括系统结构基本概念、存储系统、输入输出系统、指令系统、标量处理机、向量处理机、并行处理技术和新型计算机系统结构等。

本书作为计算机科学与技术专业大学本科50~70学时教材,也可供从事计算机硬件、计算机软件 and 计算机应用开发工作的人员参考。

图书在版编目(CIP)数据

计算机系统结构/李恒甫主编. —重庆:重庆大学出版社,2001.9

计算机科学与技术专业本科系列教材

ISBN 7-5624-2351-2

I. 计... II. 李... III. 计算机体系结构—高等学校—教材 IV. TP303

中图版本图书馆 CIP 数据核字(2001)第 06414 号

计算机系统结构

主 编 李恒甫

责任编辑 彭 宁 柏子康

*

重庆大学出版社出版发行

新华书店经销

重庆大学建大印刷厂印刷

*

开本:787×1092 1/16 印张:16 字数:399

2001年11月第1版 2001年11月第1次印刷

印数:1~5000

ISBN 7-5624-2351-2/TP·307 定价:23.00元

前言

本书是 21 世纪计算机科学与技术专业本科系列教材之一。该书编写时主要考虑:

1. 计算机系统结构应是一门关于计算机系统总体设计的课程。为了便于学习和掌握系统结构设计技术,需要系统介绍计算机系统结构,包括系统结构基本概念、基本原理、基本技术、高性能计算机系统结构和新型计算机系统结构。

2. 计算机系统结构发展迅速,新的系统结构不断涌现,在内容上应有一定深度和广度。因此,教材中尽可能引用新型计算机系统结构和先进技术,使之成为读者学习和应用时的有益参考。

3. 力求做到内容精炼,系统性强,概念清晰,重点突出。可作为 50~70 学时讲授。采用 50 学时,带 * 号的章节可以不讲,仅作参考教材使用。

全书共分 8 章,基本内容和学习重点如下:

第 1 章 计算机系统结构基本概念。使读者了解计算机系统结构的含义、内容和实现,影响计算机系统结构的因素,促进系统结构发展的原因、途径和计算机系统设计技术。

第 2 章 存储系统。学习多级存储系统,包括存储系统设计原理,并行存储器、高速缓冲存储器、虚拟存储器和相联存储器的存储原理和设计方法。

第 3 章 输入输出系统。学习输入输出系统基本概念,基本输入输出方式,通道处理机,外围处理机(PPU)和总线系统。

第 4 章 指令系统。学习数据表示,寻址技术,指令格式的优化设计,指令系统功能设计,重点学习 CISC 和 RISC 的基本技术和基本方法。

第 5 章 标量流水处理技术。学习先行控制技术,流水线处理机,超标量与超流水线处理机。

第 6 章 向量处理机。学习向量流水处理的条件和基本要求,向量处理机,提高向量处理速度的技术,向量化编译程序和附加式数组处理机。

第 7 章 并行处理技术。学习并行处理技术的发展,SIMD 并行

处理机,计算机互连网络,并行存储器的无冲突访问,多处理机,并行机的发展,包括 MPP 大规模并行处理机及机群系统。

第 8 章 新型计算机系统结构。学习数据流计算机,归约机和智能计算机。

本书由李恒甫主编,其中第 1、2、3、8 章由李恒甫编写,第 4 章和第 1 章部分内容由杨旗编写,第 5、7 章由张锦雄编写,第 6 章由李宇明编写,李恒甫负责全书统稿。

由于编者水平有限,书中难免存在一些缺点和错误,殷切希望广大读者批评指正。

目录

第1章 计算机系统结构的基本概念	(1)
1.1 计算机系统结构	(1)
1.2 影响系统结构的主要因素	(5)
1.3 系统结构发展的基本途径	(7)
1.4 计算机系统设计技术	(8)
习 题 1	(13)
第2章 存储系统	(16)
2.1 存储系统设计原理	(16)
2.2 并行主存系统	(21)
2.3 高速缓冲存储器(Cache)	(24)
2.4 虚拟存储器	(34)
2.5 高速缓冲存储器 - 主存 - 辅存三级层次	(46)
2.6 存储保护及控制	(48)
2.7 相联存储器	(51)
习 题 2	(55)
第3章 输入输出系统	(56)
3.1 输入输出系统基本概念	(56)
3.2 基本输入输出方式	(59)
3.3 通道处理机	(66)
3.4 外围处理机(PPU)的输入/输出系统	(73)
3.5 总线系统	(76)
习 题 3	(86)
第4章 指令系统	(88)
4.1 数据表示	(88)
4.2 寻址技术	(92)
4.3 指令格式的优化设计	(98)
4.4 指令系统功能设计	(103)

习 题 4	(115)
第 5 章 标量流水处理技术	(118)
*5.1 先行控制技术	(118)
5.2 流水线处理机	(127)
5.3 超标量处理机与超流水线处理机	(144)
习 题 5	(154)
第 6 章 向量处理机	(159)
6.1 向量流水处理的条件和基本要求	(159)
6.2 向量处理机的结构	(161)
6.3 提高向量处理速度的技术	(163)
6.4 向量化编译程序	(165)
6.5 附加式数组处理机	(165)
习 题 6	(168)
第 7 章 并行处理技术	(170)
7.1 并行处理技术的发展	(170)
7.2 SIMD 并行计算机	(172)
7.3 计算机互联网络	(177)
7.4 并行存储器无冲突访问	(192)
7.5 多处理机	(194)
7.6 并行机的发展趋势	(209)
习 题 7	(224)
第 8 章 新型计算机系统结构	(226)
8.1 数据流计算机	(226)
8.2 面向函数程序设计的归约机	(239)
8.3 智能信息处理的智能计算机	(243)
习 题 8	(248)
参考文献	(249)

第 1 章

计算机系统结构的基本概念

1.1 计算机系统结构

1.1.1 计算机系统层次结构

根据冯·诺依曼型计算机结构可知,计算机系统是由硬件器件和软件组成的。现代计算机中按其软硬件功能可把计算机系统划分成多级层次结构,如图 1.1 所示。其目的有三:①有利于正确理解计算机系统的工作,明确软件、硬件、固件在计算机系统中的地位 and 作用;②便于理解各种语言的实质和实现;③便于探索虚拟机新的实现方法、设计新的计算机系统。

多级层次结构中每一级对应一种虚拟机器,其作用和组成如图 1.2 所示。在这里,虚拟机器只对一定观察者存在。它的功能由广义语言实现。广义语言有两种执行方式。一种经解释器直接解释执行;另一种经编译器编译执行,然后作用在信息处理和控制对象上。某一层次的观察者,只是通过本层次语言来了解和使用计算机,不必关心内层机器如何工作,如何实现各自的功能。

图 1.1 中的第 0 级机器由硬件实现,第 1 级机器由微指令(固件)实现,第 2 级至第 6 级机器由软件实现。为区别硬件或固件实现的机器叫作实际机器,我们把由软件实现的机器称为虚拟机。虚拟机器建立在实际机器之上,由实际机器解释执行。

第 0 级和第 1 级即 M_0 和 M_1 ,具体实现机器指定功能的中央控制部分。它根据各种指令操作所需要的控制时序,配备一套微指令,由微程序控制信息在各寄存器之间传送,这就是第 1 级机器。实现这些微指令本身的控制时序所需的逻辑线路,由硬联逻辑实现,它就是第 0 级机器,是机器的硬件内核。

第 2 级即 M_2 ,是传统机器语言机,其机器语言是该机的指令系统。机器语言程序员使用本级指令系统编写的机器语言程序由第 1 级微程序进行解释。

第 3 级即 M_3 ,是操作系统机。操作系统程序员使用操作系统指令,这些指令大部分是传统机器的指令,少部分是操作系统级专用指令或操作原语。例如打开文件,读/写文件、关闭文件等指令。用操作系统级语言编写的程序中,属于传统机器指令的直接由微程序实现,属于操作系统专用指令的,由操作系统解释执行。实际上操作系统是运行在第 2 级上的解释程序。

第 4 级即 M_4 ,是汇编语言机,该级的机器语言是汇编语言程序。汇编语言程序员使用汇编语言编写的程序,首先翻译成第 3 级或第 2 级语言,然后再由相应的虚拟机器进行解释。完成

翻译的程序叫做汇编程序。

第 5 级即 M_5 , 是高级语言机, 该级机器语言是各种高级语言。高级语言程序是用高级语言编写的程序, 通常是由编译程序翻译为第 4 级或第 3 级上的语言, 个别的高级语言也用解释方法实现。

第 6 级即 M_6 , 是应用语言机。本级机器语言是应用语言。这种语言使非计算机专业人员也能直接使用计算机。只要在用户终端上使用键盘或其他方式发出服务请求, 就能进入第 6 级的信息处理系统。信息处理系统由高级语言进行解释。

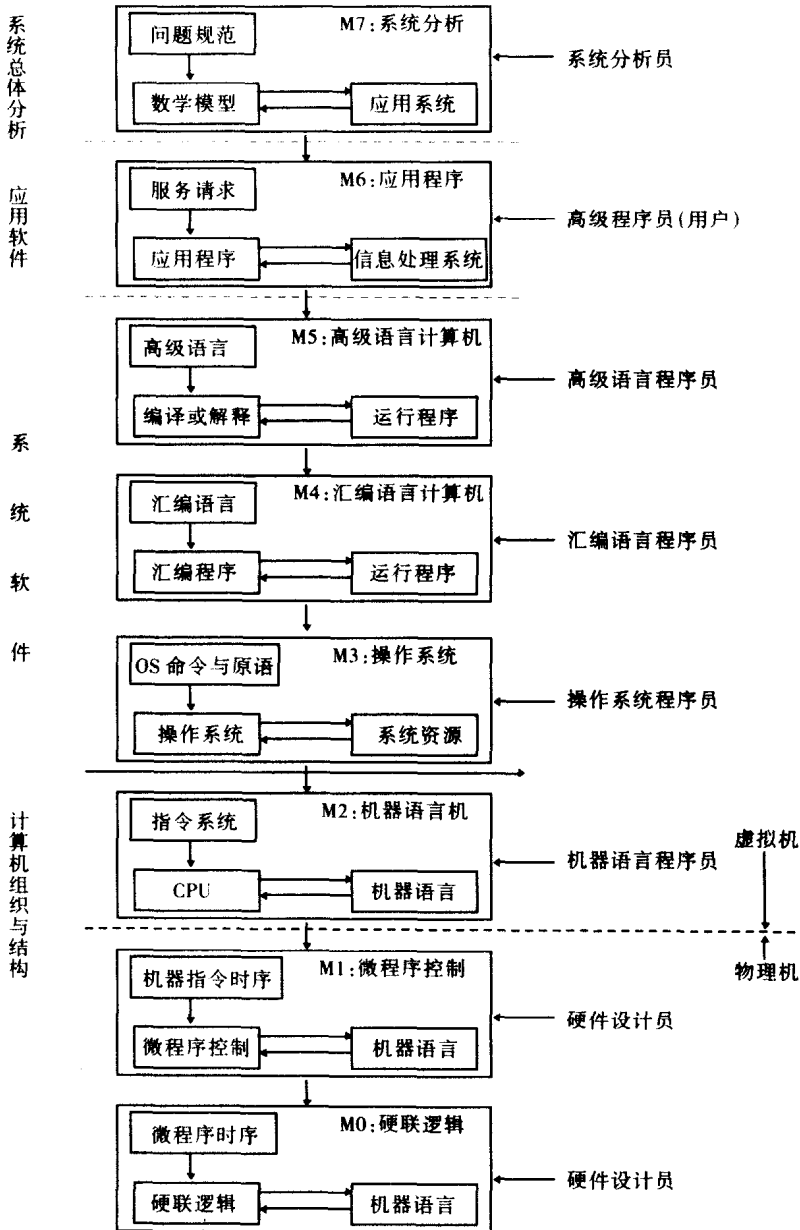


图 1.1 计算机多级层次结构

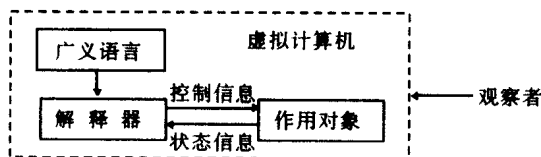


图 1.2 虚拟器的作用和组成

计算机多级层次结构除了按功能划分以外,也可按学科领域划分。按后者划分,则可将第0级至第1级认为是计算机组织与结构所讨论的范围。第3级至第5级是系统软件,第6级是应用软件的范围。当然,这种分层方法并不是严格定义的,它们之间往往存在一定的交叉。例如,第0级要求一定的数字逻辑基础,第2级涉及汇编语言程序设计的内容;第3级与计算机系统结构密切相关。在特殊计算机系统中,有些级别也有可能不存在。

1.1.2 计算机系统结构、组成和实现

(1) 计算机系统结构

计算机系统结构也称计算机体系结构。系统结构这个词,是Amdahl在1964年介绍IBM360系统时提出的,自70年代起这个概念已被广泛采用。它所研究的内容既涉及计算机硬件,也涉及计算机软件,现已成为连接硬件和软件的一门学科。

人们早期给计算机系统结构定义的内容比较窄。认为计算机系统结构是程序员(即机器语言或汇编语言)所看到的一个计算机系统的属性,即概念性结构和功能特性。这个定义只局限于系统软件设计者所看到的计算机属性,即指令系统及实现指令系统的硬件,例如通用寄存器、数据表示、程序状态字、存储器的组成和寻址技术、中断、输入输出机制等,这些内容对使用高级语言的程序员来说是看不到的,是透明的。在多级层次结构中,不同层次的程序设计者所看到的计算机属性是不同的。对于高层次的程序员一般不需了解低层次程序员所需了解的机器属性。

随着计算机技术的发展,计算机中硬件与软件的结合越来越紧密,如超长指令字(VLIW)计算机并行操作功能的发挥与优化编译程序的效能密切相关。因此计算机系统结构的含义扩大为:不仅包括软件开发人员所关心的指令系统和实现模式,而且还应包括计算机设计人员所看到的属性。为了区别这两种属性,人们把软件开发人员看到的属性称为系统结构,把计算机设计人员看到的属性称为计算机组成和实现。

(2) 计算机的组成和实现

计算机系统结构的主要任务是确定软件、硬件的界面和进行软、硬件功能分配。计算机组成就是计算机系统结构的逻辑实现,组成的主要任务是在系统结构确定分配给硬件子系统的功能及其概念性结构之后,研究各组成部分的内部构造和相互联系,以实现机器指令级的各种功能和特性。包括各功能部件的配置、相互连接和相互作用,机器内的数据流和控制流的组成与逻辑设计,着重于机器内各事件的排序方式与控制机构等各部件的功能及各部件的联系。计算机组成设计的目的是按期望的性能价格比,以最佳、最合理的方式把各种设备和部件组成计算机,以实现所定的系统结构。组成设计的内容包括数据通路宽度的确定,各种操作对功能部件共享程序度的确定,专用部件的确定,功能部件的并行性确定,缓冲器和排队的确定,控制机构的设计,可靠性技术的确定等。

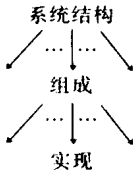


图 1.3 系统结构、组成和实现的相互关系

计算机实现,是指计算机组成的物理实现。它包括处理机、主存等部件的物理结构,器件的集成度和速度,信号传输,器件、模块、插件、底板的划分与连接,专用器件的设计,电源及电源冷却、封装等技术。它着重于器件技术和微组装技术。

(3) 计算机系统结构、组成和实现间的关系

计算机系统结构、计算机组成和计算机实现是三个不同的概念。系统结构所关心的是计算机系统结构的软、硬件界面,如何合理地进行硬件软件功能分配,为软件设计者提供更实用的机器;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。它们之间既有区别,又相互关联,如图 1.3 所示。系统结构、组成和实现三者的区别,除它们各自所包含的内容不同外,它们的设计目标也是各自不同的。系统结构设计主要围绕概念性结构进行,它关心的是采用怎样的软硬件功能分配才能为软件设计人员提供更合适的机器;计算机组成设计主要围绕提高速度进行,关心的是采用何种组成技术才能获得比较高的机器速度;计算机实现则围绕器件技术和微程序组装技术进行,它关心的是器件设计和选定,逻辑电路设计和微程序组装技术的选定。系统结构、组成和实现三者的关系可用下述三例说明。

例 1, 系统结构设计者设计指令系统;组成设计者则把指令的实现,分解为取指、取数、执行和结果送存等具体操作步骤,并按顺序、重叠、并行等多种组成方式来提高指令执行速度;计算机实现则按选定的组成方式确定指令功能的具体电路、器件设计和组装技术。

例 2, 系统结构设置一条乘法指令;计算机组成在选择乘法器时有两种组成方式。一种采用专用乘法器实现,另一种利用加法器采用重复加和右移操作实现;计算机实现,则按确定的组成方式进行器件选择以确定乘法器、加法器的物理实现。

例 3, 计算机系统的系统结构确定主存系统容量和编址方式;逻辑实现的系统结构可以采取逻辑结构措施和采用多体交叉并行主存系统来提高主存速度;物理实现的系统结构则按确定的组成技术,进行存储器选定,逻辑电路设计,微组装技术选定。

系统结构、组成和实现三者之间的关系可用图 1.3 说明,一定的系统结构,决定一定的组成,一定组成决定一定实现技术。不同系统结构要求不同的组成和实现,反之,组成和实现技术的发展又会促进系统结构的发展。在一种系统结构下有多种组成选择,一种组成也有多种实现。同一系统结构由于采用的组成技术和实现工艺不同,其性能相差可达 100 倍以上,这就构成各种性能不同,档次不同的系列机。例如 IBM360、370、4300 系列。同一系列内各档机的系统结构是相同的。其它系列机还有 PDP-11 系列、VAX 系列等。另一方面,由于组成和实现技术的发展,反过来又会促进系统结构发展。例如高速缓冲存储器,最初是作为组成和实现技术提出来的,因为它对系统结构影响较大,后来被作为系统结构的一部分而发展了系统结构。由图 1.3 还可发现,组成设计上取决于系统结构,不受实现技术限制。但它不是被动的,例如 Cache 存储器一样。

1.2 影响系统结构的主要因素

对系统结构发展影响的因素,包括软件、应用、需求、器件、价格和技术发展,其中影响最大的因素主要是软件、应用和器件3个因素。

1.2.1 软件对系统结构发展的影响

软件对系统结构发展的影响包括两个方面:一方面软件的专用算法,促进了专用计算机系统结构的发展,例如,MPP位平面阵列处理机,BSP科学计算机等。另一方面软件又制约系统结构发展。这是因为系统结构所提供的软件运行环境有一定生存期。系统结构改变了,软件运行环境也将随之改变,软件运行环境改变了,原来的程序就不能在机器上正确执行。为了适应系统结构的变化,软件设计者就不得不重新设计或研制大量的新配套软件。事实上由于软件生产率比较低,每10年才提高两倍多,而且也不应该在短时间内频繁地按新的系统结构、新的指令系统去重新设计其系统软件和应用软件。解决这一问题的有效办法是维持系统结构的运行环境不变,采取软件可移植性(即软件兼容性)技术,使同一软件可以应用于不同的运行环境。

实现软件兼容性的方法有3种:

1)所有系统所有软件采用统一的高级语言。由于语言的标准化问题,一下子难于统一,目前这一方法还难于实现。

2)采用系列机思想。就是为了使软件长期稳定,在相当长的时期内保持系列机的系统结构基本不变。硬件设计者根据机器速度、性能、价格的不同,选择不同器件,采用不同的硬件技术,组成技术和实现技术,研制和提供不同档次的机器。我们把同一公司内具有相同系统结构的各档机器称为系列机;把不同公司厂家生产的具有同一系统结构的计算机,称为兼容机。系列机的思想,成功地解决了系统结构的发展与软件的矛盾,也对计算机工业的迅速发展起了很好的推动作用。

3)采用模拟与仿真技术,实现不同系统结构机器间的软件兼容。其中仿真,就是用一种机器的微程序直接解释执行另一种机器指令系统的方法,它适用于使用频繁且易于仿真的机器指令;模拟,就是用一种机器的机器语言所编写的模拟程序,直接解释他机的机器语言的方法,它适合于移植运行时间短、使用次数少,在时间关系上没有约束和限制的软件。对于不同系列间的软件移植,往往通过仿真和模拟两种方法并用来实现。由于软件的兼容性要求,因而制约了系统结构的发展。为了解决软件可移植性与系统结构发展的矛盾,可以通过采用不同的组成技术和不同的实现技术,发展系列机来解决。

1.2.2 应用对系统结构的影响

随着计算机应用领域的不断扩大和深入,对计算机系统结构的设计也提出广泛要求。如高速运算、大存储容量和大的I/O吞吐率。系统结构为各种不同的应用提供支持

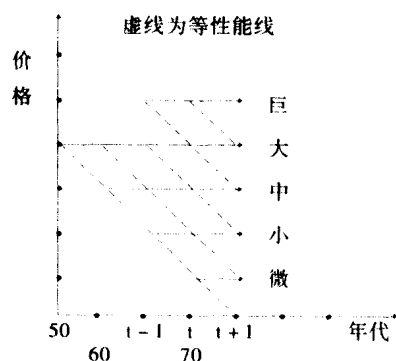


图 1.4 计算机等级与价格、性能关系

而促进了它自身的发展。例如计算机应用从最初的纯科学计算逐步向更高级、更复杂的应用发展,经历了从数据处理、信息处理、知识处理和智能处理 4 级逐步上升的阶段。

如图 1.4 所示,计算机经过前几十年的发展已形成巨型机、大型机、中型机、小型机、微型机等多功能、多系列的通用计算机。巨、大、中型计算机中,其中巨型机、大型机属于高性能计算机,以满足高处理能力的应用需要。对高性能计算机,需要不断研制新的计算机系统结构和组成技术,以满足其高速或高性能的要求。例如,中央处理机的重叠、流水和并行处理技术,存储系统中的超高速缓冲存储器和虚拟存储层次,通道 I/O 处理机和外围处理机方式,各种从系统结构上提高系统可靠性的技术,多处理机技术,采用高级数据表示的向量机、阵列机等。

其他,如中、小、微各型计算机所具备的性能和系统结构,是从巨、大型机随时间动态下移,即低档机承袭高档机的系统结构和组成。小、微型机继承巨、大、中型机的系统结构。图 1.4 中,斜线为等性能线,巨、大、中、小、微型机沿此等性能线随时间下移。水平线为各型计算机各自自在一定的系统结构和组成技术上随时间继续发展,发展了又下移至中、小、微型机上。因此,中、小、微机的设计原则是采用承袭和发展的策略。“承袭”就是保持系统结构基本不变,“发展”就是发展硬件功能。

1.2.3 器件对系统结构发展的影响

器件的发展是推动计算机系统结构和组成发展的主要因素。器件的发展,不仅对高性能计算机是必需的,而且对加速巨、大、中、小、微型计算机系统结构的下移速度也是重要的。器件的影响主要包括器件的功能和器件的集成度。器件的功能也与集成度有关。器件的集成度越高,器件的速度和机器的主频越高。

首先,器件的变革对机器速度影响极大。例如,最初的电子管机,计算机的计算时间单位为 ms,相应中央处理机的速度不足千次;20 世纪 70 年代中 LSI 机,解题速度提高了两个数量级,时间单位为 μs ,相应中央处理机速度为百万次;现在采用超大规模集成电路 VLSI 计算机为 ns,相应中央处理机速度为亿次;若用约瑟夫逊器件则为 ps,相应处理机速度将达数百亿次甚至千亿次。

其次,器件是系统结构及其组成技术的基础。例如,微程序技术,流水技术,Cache,相联处理机,向量机,数组机,数据库机等都有赖于器件技术的进展。VLSI 技术的发展,带来了集成度的提高,体积的缩小,可靠性高和价格便宜,使系统结构的下移速度加快,促进了多处理机的分布处理、智能终端和智能计算机的实现。

最后,器件的发展还影响算法、语言和软件的发展。由于器件技术发展了,促进了计算机系统结构发展,出现了并行处理机,多处理机和数据流计算机等新型计算机结构,将推动并行算法、并行语言、并行处理应用软件和并行操作系统的发展。

1.3 系统结构发展的基本途径

1.3.1 推动系统结构发展的原因

推动系统结构发展的原因包括：

- 1) 要求增大系统的处理能力。随着计算机应用领域的扩大，一般都迫切要求提高系统的处理能力，降低机器价格。从 20 世纪 50 年代起，计算机性能价格比是每 10 年提高约 100 倍。
- 2) 要求提高软件生产率。由于软件开发周期与机器发展速度差距很大，计算机程序开发的生产率每 10 年只提高 2 倍左右，随着应用领域的扩大，要求能够有大幅度提高软件生产率的结构。
- 3) 不断改进系统结构，以适应新的应用领域要求。
- 4) 改进人机接口，促进人机交往的智能化。
- 5) 从系统结构技术上满足高可靠性要求，促进容错计算机的发展。

1.3.2 系统结构发展的基本途径

器件的发展对推动计算机的发展起了重大作用。但是单纯依靠改进工艺技术来提高逻辑电路的开关速度，达到提高计算机速度的潜力已越来越小，这就促使人们从系统结构上来提高计算机系统性能。

发展系统结构的主要途径，一是改进和发展冯·诺依曼型机的系统结构，二是采用并行处理技术。并行处理技术，包括并发技术、重叠技术和并行技术，如图 1.5 所示。

并发技术是通过资源共享、分时技术和虚拟技术，使多道程序同时在一台处理机上并发执行；重叠技术是将一个操作或一个处理分解成若干微过程，然后采用先行控制技术使这些微处理过程在时间上重叠处理，实现指令流水线和存储器并行；采用多功能部件并行和运算流水技术。

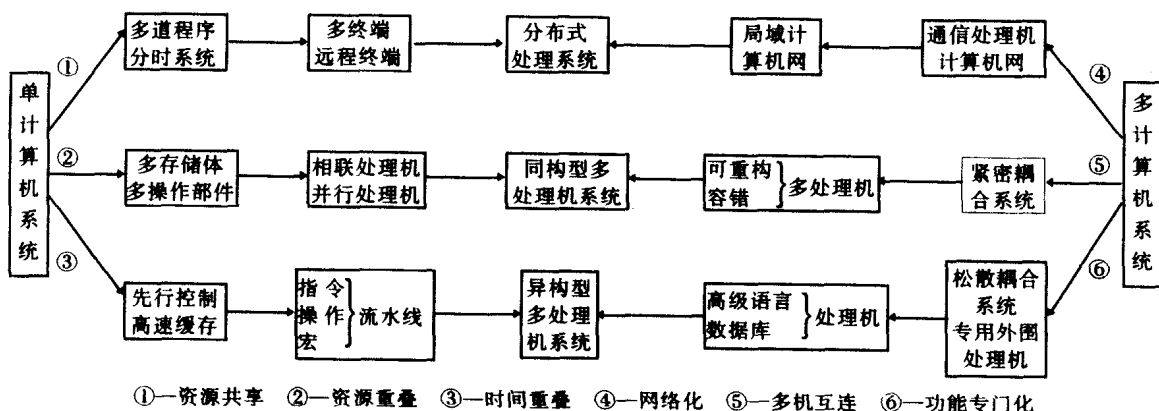


图 1.5 发展系统结构的主要途径

1.3.3 高性能和并行处理计算机的发展

高性能计算机发展的突出问题是提高计算机速度。提高单机速度的方法是从系统结构上发展并行处理技术。从系统结构上提高单机处理速度的改进包括:增设 CPU 内的通用寄存器数量和 Cache 存储器,采用多体交叉存储器,将程序和数据分别存放于不同存储体内,采用指令流水线和运算流水线,多功能部件和增设支援处理器或协处理器以分担 CPU 一些费时的 I/O 操作。在多道、分时技术和多功能运算流水线技术基础上发展向量巨型机,以及 20 世纪 80 年代发展的 RISC 技术。向量巨型机综合了单处理机上各种并行处理技术,其功能已达到极高水平,但单机性能逐渐呈现饱和趋势。高性能计算机继续发展,必须发展多处理机系统的并行处理技术。

多处理机系统并行处理的基本途径是发展并行处理机、多处理机和容错计算机。并行处理机,就是利用资源重复技术,将大量相同处理部件,在一个集中控制部件控制下,同时对各不相同的数据执行同一条指令的操作,以提高系统处理速度。所谓多处理机系统,就是用多台相对独立的计算机以松耦合技术或紧耦合技术组成的多处理机系统,在一个统一操作系统控制下同时完成各机的数据处理。这种系统除带来高速处理外,也对提高可靠性和实现资源共享带来好处。容错计算机系统,是一个紧耦合的多处理机系统。它是多处理机技术的一个重要分支。容错计算机系统结构技术的应用,目前已从实时控制扩展到银行金融业务等联机事务处理。

1.3.4 新型计算机系统结构

高性能计算机和并行处理计算机在并行处理技术方面得到了充分发展,使计算机性能达到极高的水平。但是由于它们本质上仍属于冯·诺依曼型机器结构,很难最大限度发掘其计算并行性。

由于 VLSI 技术的发展,促进了一些以 VLSI 为基础的新系统结构,或新系统结构思路的发展,突破了传统的冯·诺依曼型计算机系统结构。新型计算机系统结构的发展有 3 个发展方向。

- 1) 数据驱动的数据流计算机;
- 2) 按需求驱动的归约机;
- 3) 智能计算机。

1.4 计算机系统设计技术

1.4.1 计算机系统结构分类

研究计算机系统的分类有助于人们了解计算机系统的演变和对计算机系统的工作原理及性能的理解。

计算机系统分类的方法根据不同的观点、不同的角度有许多种。下面着重介绍 Flynn 分类法:

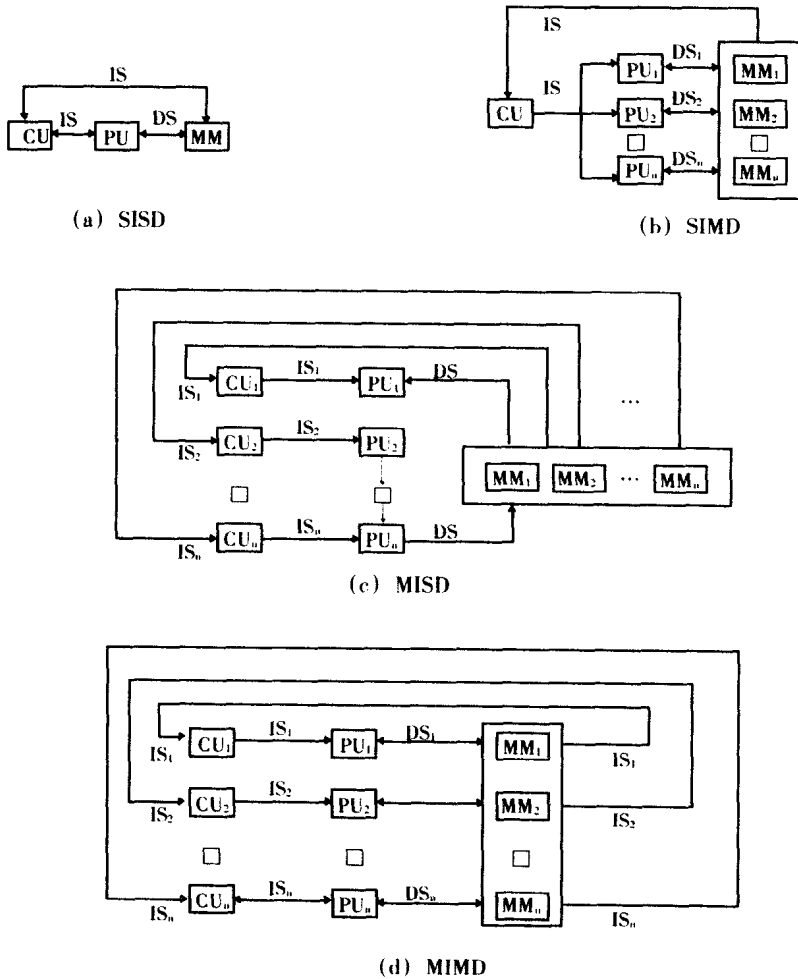
Michael J. Flynn (1966) 根据指令流和数据流的多倍性状况对计算机系统进行分类。这里指令流是指机器执行的指令序列;数据流是指由指令流调用的数据序列(包括输入数据和中间结

果);而多倍性是指在系统最受到限制的元件上处于同一执行阶段的指令或数据的最大可能个数。这样,他按照指令流和数据流的不同方式,把计算机系统分成 4 类:

- 1) 单指令流单数据流 SISD(Single Instruction stream Single Datastream);
- 2) 单指令流多数据流 SIMD(Single Instruction stream Multiple Datastream);
- 3) 多指令流单数据流 MISD(Multiple Instruction Stream Single Datastream);
- 4) 多指令流多数据流 MIMD(Multiple Instruction Stream Multiple Datastream)。

对应于这 4 类计算机的基本结构框图如图 1.6 所示。SISD 是传统的单处理机。SIMD 是指以阵列处理机为代表的并行处理机。MISD 是属于何种机器有不同的看法,有的学者把流水线结构机器看成是 MISD 结构。MIMD 包括了大多数多处理机和多计算机系统。

1972 年,冯泽云教授提出用并行度来定量地描述各种计算机系统的冯氏分类法。



CU: 控制部件 PU: 处理部件 MM: 存储器模块 IS: 指令流 DS: 数据流

图 1.6 Flynn 分类法各类机器结构

1977 年, Wolfgang Handler 在冯氏分类法的基础上, 又根据并行度和流水线提出了另一种分类方法。

此外, 还有其他很多分类方法, 这里不详细介绍, 可参阅有关资料。

1.4.2 计算机系统设计的定量原理

在计算机系统设计中, 应遵循的几个定量原理:

1) 加快经常性事件的速度 (make the common case fast)

这是最重要也是最广泛采用的设计准则。加快对经常性事件的处理速度可以明显地提高整个系统性能。例如: 在 CPU 中进行两个数相加时, 相加结果可能溢出, 也可能不发生溢出。不溢出是经常出现的事件, 溢出是偶然发生的事件。因此, 只要对不发生溢出的情况, 加快执行, 而不必过多考虑对溢出情况的处理, 因为这种情况发生概率很小, 即使发生了, 处理得慢一些也不会对系统性能产生太大的影响。

对经常性事件, 如何加快其处理速度, 可用 Amdahl 定律解决。

2) 阿姆达尔 (Amdahl) 定律

阿姆达尔 (Amdahl) 定律是 1967 年由 IBM360 系列机主要设计者 Amdahl 首先提出来的。这个定律指出: 系统中对某一部件采用某种更快执行方式后, 整个系统性能的提高与这种执行方式的使用频率或占总执行时间的比例有关。这一定律定义了对某些功能加速后所获得的性能改进或执行时间的加速比。

$$\begin{aligned}\text{加速比} &= \frac{\text{采用改进措施后的性能}}{\text{采用改进措施前的性能}} \\ &= \frac{\text{采用改进措施前执行某任务的时间}}{\text{采用改进措施后执行某任务的时间}}\end{aligned}$$

同时, 加速比还与两个因素有关: 一是被改进部分所占时间在执行某任务的总时间的百分比 (假设用 F_c 表示); 二是采用改进措施后比没有采用改进措施前系统性能提高的倍数 (假设用 S_c 表示), 则改进后整个任务的执行时间为:

$$T_c = T_0 \left(1 - F_c + \frac{F_c}{S_c} \right)$$

其中 T_0 为改进前整个任务的执行时间。

$$S_p = \frac{T_0}{T_c} = \frac{1}{(1 - F_c) + F_c / S_c}$$

式中 $1 - F_c$ 表示不可增强部分。可以看出, 当 F_c 为 0, 即没有可改进部分时, $S_p = 1$ 。系统性能的提高受 F_c 的影响。当 $S_c \rightarrow \infty$ 时, $S_p = 1 / (1 - F_c)$, 说明可获取性能改善的极值受 F_c 的约束。

例 1: 在某一系统中, 将其中一个部件的处理速度加快到 10 倍, 但该部件的原处理时间仅为系统总运行时间的 40%, 采用加速措施后, 能使整个系统性能提高多少?

解: 由题可知: $F_c = 0.4$, $S_c = 10$, 根据 Amdahl 定律