计算机系统结构

（上　册）

北　京　工　业　大　学

计算机科学系

1981.1

译 者 前 言

本书原名为“结构化的计算机体系”（*Structured Computer Organization*），作者是荷兰 *Vrije* 大学的安德鲁 S·塔恩伯姆（*Andrew S. Tanenbaum*）。作者为本书所加注的小标题就是“一个程序设计者应该掌握哪些有关系统结构的知识”。这本书本身就是一本教科书，同时也是一本以结构程序设计为主导思想而编写的计算机总体设计方面的书。

“计算机系统结构”也称“计算机总体结构”或“计算机体系结构”。“计算机体系结构”（*Computer architecture*）这个术语是由 IBM 公司的一些计算机设计师在 60 年代早期提出来的。当时 IBM 公司对系列机的汇编语言设计提出的要求是，每种机器在汇编语言级必须要具有共同的功能，但对程序设计人员所看不到的那些硬件，并不提出具体的规定。对那些硬件来说，可以根据产品的运算速度和成本要求进行选取。这样，系列机的设计就可以达到它的设计者所希望的那种能共享软件的要求，即系列机有了相同的“体系结构”。这种在设计 IBM360 系列机时所采用的思想后来也被好几家计算机公司所采用。

当时 IBM 公司的 G. M. *Amdahl*，G. A. *Blaauw* 和 F. P. *Brooks* 等对计算机体系结构作了如下的定义：“是从程序员的角度所看到的系统的属性。也就是说，是概念上的结构和功能的行为，它不同于数据流程和控制的组织，不同于逻辑设计以及物理实现方法。”这就是说，计算机结构是指计算机设计的基本思想和由此产生的逻辑上的构造。至于构成计算机的元件是什么，计算速度是多少，物理构造如何，价格和功能如何等等，亦即计算机的硬件和软件是什么，则不加过问。

换句话说，“体系结构”是指计算机为了满足某种要求而采用的设计方案。严格说来，它是指所选用的指令形式和联接输入输出设备的能力。“体系结构”相同的机器则能完成相同的程序，并可接上相同的输入输出设备。机器的硬件结构通常可由一个方框加以说明。例

如，数据通用公司的 Nova 1200 机和 Nova 800 机。由于具有相同的指令结构和同样的输入输出设备，所以可以说它们有相同的体系结构。但这两种机器的方框图，即硬件结构是不同的，也就是说，机器的"执行过程"是不同的。

因此，在实际设计计算机体系结构时，如果只考虑计算机用在什么领域和以何种使用方法为主，而不考虑在计算机中可能使用的硬件和软件技术，则所设计的体系结构就会无法实现。从这个意义上讲，体系结构与硬件技术和软件技术有着密切的关系，可以说体系结构是硬件和软件的根本。体系结构通过硬件技术和软件技术的组合而具体化，从而就可以得到实际的硬件和软件。因此，也往往把计算机体系结构看作是计算机的基本设计。

在计算机系统中，有关体系结构的概念可以细分为以下三个部分：

1. 计算机体系功能：指计算机的功能描述（例如数据格式、操作代码、寻址方法等），以及这些功能的逻辑实现（但不包括具体的逻辑线路和工程实现）。文献中常用 Architecture 一词表示。

2. 计算机系统组织：指计算机主要部件（如存贮模块、中央处理机、输入输出处理机）的互连方式，控制方式和信息的流动方式，它说明了计算机功能在部件一级的逻辑实现。文献中常用 Organization 一词表示。

3. 计算机系统配置：是在计算机功能和组织确定之后，针对一定的应用确定各种设备的类型和数量，来组成一台具体的计算机。文献中常用 Configuration 一词表示。

根据这三个部分的概念，具体地说，计算机系统结构大至包括下面一些内容：

(1)、有关指令形式—— 指令格式、寻址方式。

(2)、有关数据—— 数据格式、代码。

(3)、有关处理器与输入输出设备的联系—— 再定位、引进引出、存贮保护、分段、分页等。

(4)、有关硬件与软件的联系—— 硬件/软件的功能分配。

(5)、有关处理器的构造—— 非纽曼构造、多处理机、复合计算机

等。

(6)、有关人和机器的联系——可靠性、可维修性、人机接口、扩展性、兼容性等。

从以上几方面可以看出，计算机系统结构所要解决的问题是大系统的问题，其中包括软件和硬件。作为一门课程来讲，它既不同于纯粹的硬件课程，也不同于纯粹的软件课程。学习本课程的目的，就是为了使学生对计算机软件和硬件能有一个比较完整、比较全面的认识，为能更好地应用和设计计算机打下基础。

至于如何去阐述有关计算机系统结构的各部分的组成内容，本书不是把与上述有关的那些系统性不强的各部分内容，散列在“计算机系统结构”这个总标题之下，一个一个地去分析、研究，也不是大致按在“计算机原理”和“计算机组成”这两门课程中那样的讲授次序去分析、研究，而是按一种先进的思想——分层结构思想，去组合、分析、研究这些内容。正是因为本书对计算机系统结构的研究采用了分层结构思想，所以使得复杂的计算机结构——软件和硬件的组合体，在我们面前呈现得是那样层次清楚、结构严谨。因此可以说，这种研究方法是对计算机系统结构这门科学的一种深化和发展。

分层结构的思想，即分层虚拟机的观点，是荷兰人 *Dijsktra* 在 *1967* 年提出来的。*Dijsktra* 把升了操作系统设计中传统的模块接口方法，采用层次结构方法实现了一个操作系统，结果使操作系统的正确性得到了极大的提高。此后分层结构的思想，在计算机科学中得到了广泛的应用，尤其在软件设计方面就更是如此。

本书是以 IBM *370*，CDC *Cyber 70* 和 DEC PDP—*11* 等多层计算机为实例，通过对分层结构的论述，广泛地讨论了有关计算机系统结构的各种先进课题，如被程序设计，汇编语言，操作系统，以及虚拟存贮器，并行处理，指令设计等。所以本书对计算机总体设计及程序设计者都有一定的参考价值。

根据上述情况，当然主要还是由于目前尚未编写出适于计算机软件专业的“计算机系统结构”这门课程的教科书，所以就翻译了这本书，用来暂做我校计算机软件专业的“计算机系统结构”课的教材。

全书共八章，分上下两册付印，上册的内容是：(1)、引论；(2)、计算机系统的体系结构；(3)、普通机器层；(4)、微程序设计层。下册的内容是：(5)、操作系统的计算机层；(6)、汇编语言层；(7)、多层计算机；(8)、供深入学习的参考知识。

限于译者水平，译文中一定有不少错误，敬请读者批评指正。

计算机软件教研室　　董英斌

一九八〇年10月

原　序

很久以前，计算机是很简单的。这些早期的机器执行着为数不多的初级指令，用户也直接使用这些初级指令来写程序。但这都是很久以前的事情了。现在，一台现代计算机已是一个十分复杂的实体，它通常是由六个或更多个能够在其上面编程序和进行研究的不同的层次组成的。实际上，现在已经很难说清楚"硬件"部分从何处结束，"软件"部分从何处开始。

许多大学在早期的课程中，都有一个汇编语言程序设计和计算机组成的课程。在那个时候，给学生讲授怎样使用汇编语言来为一台专用的计算机编写程序是合适的。但这也是很久远的事情了。这些课程必须跟上迅速变化的客观进程，因此现在就必须要对已广泛涉及到的计算机结构方面的一系列的课题加以介绍，但在几年之前，在这些课题当中，就有不少课题还仅为从事这方面研究和实验的人们所知。分段虚拟存贮器（*Segmented Virtual memories*），并行处理（*Parallel processing*），竞态条件（*Race Conditions*），微程序设计（*Microprogramming*），可变结构机器（*VariableArchitecture machines*），存贮转交网络（*Storeand forward networks*），自虚拟机器（*Self—Virtualizing machines*），以及超高速缓冲存贮器（*Cache memories*），这些只是那众多课题当中的一些例子。

做为一本教科书，本书用于介绍汇编程序设计语言和计算机结构的课程。学习这门课程的唯一先提条件，就是一门有关计算机科学导论性质的课程（或者同等学历的实践经验），其中要包括一些象FORTRAN，COBOL，或PL/1这样的高级语言。由于不需数学和工程的基础，使得本书适合于大学二年级的程度使用。本书几乎把ACM（美国计算机协会）课程表0 8里的整个B 2课程全部包括在内了，并且由于我把上面所提到的些有关计算机结构方面的内容也写进本书了，于是本书就如同那些内容一样而变得重要了。本书也可以用于有关计算机结构和机器语言程序设计方面所推荐的COSINE课程。

本书每一章的内容都是完整的，因此也可以把它们当作有关专门课题的参考文献使用。

本书主要以分层结构的形式，讲述结构化的计算机体系。最底下的一层是真正的硬件，它的功能就是执行被称做为微程序的解释程序。微程序所解释的语言，就是大部分人所称为的“机器语言”，也就是在生产厂商所提供的有关机器资料的说明书中所描绘的那种“机器语言”。这种机器语言实际上就是从由电子线路直接执行的那种语言里所分离出来的一个层次。我们把最底下的这一层叫作“微程序设计层”（第四章），把由它所支持的那一层叫作“普通机器层”（第三章）。

大多数计算机都有一个操作系统，它运行在普通机器层上。操作系统为用户提供了一台“扩充的”或“虚拟的”机器，这种“扩充的”或“虚拟的”机器具有在普通机器层上所得不到的一些指令和灵活性。其中包括文件处理指令（*File manipulation instructions*），并行处理多重任务（*Parallel processing Multitasking*）指令和虚拟存贮器的指令。对操作系统的用户来说，这样一组可以使用的特性和指令，就可以被定义为一个新的层次：操作系统机器层（*the Operating system machine level*）（第五章）。

第四层是符号汇编语言层（*the Symbolic assembly language level*）。第二层和第三层是由解释来支撑的，第四层和它们不同，它是由一个称做汇编程序（*Assembler*）的程序来支撑的（第六章），这个汇编程序把第四层上的程序翻译成在它下面那一层上的程序。本书未能包括那些更高的层次，那些更高的层次都是由面向问题的语言所定义的。

第七章分析了整个多层机器的结构和应用，而第3，4，5，6各章所分析的都是单独的一个层次。第八章是对更深入研究的一个导论。

本书从头到尾都把IBM 370，CDC Cyber 70，DEC PDP——11型的计算机当做实例进行分析。使用这些机器只是为了阐述的目的，并不意味着它们比其它型号的机器有什么优越。

使用PL/1的简单的子集（*Subset*）所写的少数算法，对那些

熟悉 FORTRAN 或 ALGOL 的人来说，是一看就会明白的。之所以使用 PL/1 是因为它既应用得很广泛又适于结构程序。我原先选择的是 ALGOL 68，但遗憾的是只有少数二年级的学生才了解 ALGOL68。

本书是多人努力的结果。我多次地感觉到，与其说我是作者，倒不如说我是编者。有两个人与其它人相比尤其值得一提。杰克·阿伦恩（Jack Alanen）完整地阅读了本书第一稿，并使许多错误很本未被打字。他还对书中的内容和叙述提出了大量的建议。我特别要感谢杰克的是，他教会了我许多有关教学的知识。其后，基姆·卡斯特洛（Kim Gostelow）又继续仔细地作了检查，进行了评论，纠正了一些错误，并且又设法把我那乏味的文体改得多少象个英语的样子。因此在手稿每一个完整的自然段里，几乎都有建议和修改的痕迹。我对他们二位是深信不疑的。

瑞恩德·范·德·里特（Reind Van de Riet）和米切尔·塔恩伯姆（Mitchell Tanenbaum）也阅读和评论了整个手稿，并从不同的角度上给我提出了意见。John W. Carr III, Arnie Falick, Dick Grune, Jim Van Keulen, Bob Rosin, Carel Stilleboer 和 Wayne Wilner，他们对各章节部分地分别提出了建议和帮助。我还要特别感谢我的学生，尤其是 Arie de Bruin, Wim Harmsen, Ad König, Sape Mullender, Johan Stevenson 和 Hans Van Vliet，他们给我提供了教学反应。Homburg—Knaepe 太太对几稿的打字都完成得很好。

我十分感谢国际商业机器公司（IBM），控制数据公司（CDC），数字设备公司（DEC）以及 Burroughs 公司，他们允许我利用他们拥有版权的出版物，如：IBM公司的“IBM System/370 Principles Of Operatios Manual”和 IBM3125 CPU的工程图纸；CDC公司的“Control Data Cyber 70 Models 72, 73, 74 Computer System Reference Manual”；DEC公司的：“PDP—11/45 Processor Handbook”和 PDP—11/40的工程图纸；Burroughs 公司的：“B1700 System Reference Manual”。对这些计算机的任何讲述方面的错误都由

我个人负责。

最后我要感谢苏珊娜对我的鼓励、支持和帮助，特别要感谢，由于我在编写这本书时，为自我强制而远离人间生活的那一段很长的时间里，她那宽容的态度。我还要感谢她以 *uitgc perste sinaa-sappel* 使我长时间的工作有所中断。

安德鲁·S·塔恩伯姆

（ANDREW S. TANEBAUM）

课 程 指 导 序 言

使用本书的任何一门课程，都要把汇编语言程序设计的练习当作一个重要的部分。在开课的第一天，就应该给学生一份有关文献的详细目录，这个文献目录是在所要使用的那台计算机上，调试汇编程序语言时需用的，同时还要对那台计算机做一个概括的介绍，对运行程序的详细管理也要有一个简单的解释——到什么地方去寻找终端设备或者作孔机，需要什么样的做业控制卡片，如何使用编译程序等等。学生们应该做的是穿孔和执行程序，因为没有必要为了介机器的详细情况而浪费时间。

下一步就是让学生们对程序做些一般的变换。例如，使用一种简单的代用记号，就可以把一个对信息进行编码和译码的文件程序变换成可以处理数字和文字的程序。在对一个或两个简单的程序做过若干次连续的更难一些的变换之后，就要让学生为变换一个更难的程序做好准备。

有一个很好的选择，就是教师要为图 4—15 里的那个解释程序做出一个汇编语言的解释。我们希望学生将能改进由解释程序所支持的目标机器。在第四章结尾处的问题里，我提出了一些建议，但要鼓励学生们独立思考。这个练习更加明确地说明了，普通机器层确实是由在其底下的那个解释程序（微程序）所确定的，而不是由硬件所确定的。

可以做的另外一些练习则是模拟计算机网络或者虚拟存贮器的操作，或者是对一个简单的汇编程序加以改进。把工作程序当作一个基本的东西提供给学生，要比让他们完全从无做起更为可取。这样做将能使那些进度较慢的学生，避免在“你把这个操作码放在哪一行里？”这类问题上浪费更多的时间。同时也为那些学习较好的学生，提供一种方便的形式去做那些更难的课题。如给一个汇编程序增加大量的灵活性。

用一台小型计算机来依次传递经验（Hands—on experience）优于使用分时终端，而分时终端又优于成批环境（Batch environment），但这种选择通常是由设备的效能来决定的。在这门课程当中，要尽可能早一些地安排学生去阅读那些属于厂商手册里的适当的课题。

本书既可以当做集中在一个学期里所讲授的课程，也可以再加上一些本书之外的阅读材料，从容地安排成二个学期的课程。如果要改为一个学期的课程，就只能集中在主要问题上，而删略其它部分。例如：计算机结构课程的内容可以只使用第1，2，3章的一部分，和第4，7两章。强调汇编语言程序的课程可以只使用第1，2，3，6章。要是安排成两个学期的课程，就要包括所有的章节，但那些较高深的内容和附加的那些例子，要留给那些更具雄心的学生去攻读。

上册目录

第一章 引 论

一台数字计算机，通过执行人们给它的指令，来解决人们提出的问题。描述怎样执行一定任务的指令序列称为程序。每台计算机的电路都可以识别和直接执行一组限定的简单指令，在这些程序被执行之前，全部程序一定要首先转变成那些简单的指令。实际上，这些基本指令要比加两个数、检查一个数是否是零，把一块数据，从存贮器的一个地方搬到另一个地方等复杂得多。

总之，计算机的基本指令形成了一种语言，利用这种语言，人们就可以和计算机进行联系。象这样的一种语言就被称为机器语言（*machine Lanyuage*）。

当人们设计一台新计算机时，就必须确定哪些指令要包括在它的机器语言里。一般人们总是力图使这些基本指令尽可能地简单一些，并且还要和这台计算机予期的应用以及操作的要求相一致，以便尽量减少它的复杂性和在所需电子器件方面的花费。尽管大部分机器语言都是相当简单的，但对于人们使用来讲还是显得困难和累赘。

解决这个问题有两种基本方法；这两种方法都包括要设计一组新的指令码，而这组指令码和原来的那组机器指令码相比，要更适于人们的使用。总而言之，这些新的指令码也形成了一种语言，我们称之为 L_2，而机器的固有语言（*built—in machine language*）称之为 L_1。这两种方法在形式上是有差别的，用 L_2 写成的程序是由计算机来执行的，但是最后，计算机仅能执行的是用 L_1，即用它的机器语言所写成的程序。

执行用 L_2 写成的程序的一种方法，就是首先把这个程序里的每一条指令，用 L_1 里指令的一个相当的序列来进行置换。其结果就是，产生了一个新的完全由 L_1 指令组成的程序。随后计算机所执行的是这个新的 L_1 的程序，而不是原来的那个 L_2 的程序了。这种技术被称为翻译（*translation*）。

另一种方法就是用 L_1 写一个程序，这个程序把用 L_2 写的程序当作输入的数据，通过对每一条指令的依次分析，并进行运算，直接