高等学校教材

计算机系统结构

苏东庄 主编

西北电讯工程学院出版

高等学校教材

计算机系统结构

苏东庄　主编

西北电讯工程学院出版社

1984

内 容 简 介

本书讲述计算机系统结构的基本概念、基本原理、基本结构和分析方法。

全书共分八章。第一章讲述计算机系统层次结构，计算机系统结构、组成、实现的定义和相互关系，系列机概念，应用与器件的影响，计算机设计自动化。第二章讲述数据表示的确定，寻址方式，指令系统的设计。第三章讲述重迭、流水控制方式。第四章讲述总线、中断、I/O处理机。第五章讲述存贮层次，地址映象，替换算法，虚拟存贮器，Cache存贮器。第六章讲述故障检测与定位，纠错码，可靠性模型。第七章讲述并行结构，多机系统，并行处理机，多处理机。第八章讲述硬件描述语言和计算机系统结构性能评价。

本书可作为有关专业的教材和科技人员的参考书。

高等学校教材

计 算 机 系 统 结 构

苏东庄 主编

西北电讯工程学院出版社出版

西北电讯工程学院印刷厂印刷

陕西省新华书店发行　　各地新华书店经售

开本 787×1092 1/16　印张 22 10/16　字数 550千字

1984年11月第一版　1984年11月第一次印刷　印数1-30,000

统一书号：15322·5　　定价：3.80元

出 版 说 明

根据国务院关于高等学校教材工作分工的规定，我部承担了全国高等学校工科电子类专业课教材的编审、出版的组织工作。从一九七七年底到一九八二年初，由于各有关院校，特别是参与编审工作的广大教师的努力和有关出版社的紧密配合，共编审出版了教材159种。

为了使工科电子类专业教材能更好地适应社会主义现代化建设培养人才的需要，反映国内外电子科学技术水平，达到“打好基础、精选内容、逐步更新、利于教学”的要求，在总结第一轮教材编审出版工作经验的基础上，电子工业部于一九八二年先后成立了高等学校《无线电技术与信息系统》、《电磁场与微波技术》、《电子材料与固体器件》、《电子物理与器件》、《电子机械》、《计算机与自动控制》，中等专业学校《电子类专业》、《电子机械类专业》共八个教材编审委员会，作为教材工作方面的一个经常性的业务指导机构，并制定了一九八二～一九八五年教材编审出版规划，列入规划的教材、教学参考书、实验指导书等共217种选题。在努力提高教材质量，适当增加教材品种的思想指导下，这一批教材的编审工作由编审委员会直接组织进行。

这一批教材的书稿，主要是从通过教学实践、师生反映较好的讲义中评选择优和从第一轮较好的教材中修编产生出来的。广大编审者，各编审委员会和有关出版社都为保证和提高教材质量作出了努力。

这一批教材，分别由电子工业出版社、国防工业出版社、上海科学技术出版社、西北电讯工程学院出版社、湖南科学技术出版社、江苏科学技术出版社、黑龙江科学技术出版社和天津科学技术出版社承担出版工作。

限于水平和经验，这一批教材的编审出版工作肯定还会有许多缺点和不足之处，希望使用教材的单位、广大教师和同学积极提出批评建议，共同为提高工科电子类专业教材的质量而努力。

电子工业部教材办公室

前　言

本教材是高等院校工科电子类计算机专业统编教材之一，由计算机与自动控制教材编审委员会计算机编审小组评选审定，并推荐出版。

本教材是按计算机编审小组审定的编写大纲进行编写和审阅的，由西北电讯工程学院苏东庄主编，清华大学薛宏熙主审。

本教材是按“研究软、硬件功能分配以及如何最佳、最合理地实现分配给硬件的功能”这个方向来编写的，着重于基本概念、基本原理、基本结构和分析方法。虽然在讲述时联系了实际机器，但并不是围绕某种机型或过多地从具体实现上的细节来讲述。本教材力求反映近十几年来在系统结构上的重要进展以及今后可能的发展。

本课程应在“计算机(组成)原理”、“程序设计语言”和“数字逻辑”等课程之后开设。学生最好有“数据结构”方面的知识。本课程可以在“操作系统”、“编译原理”课程之后，或与它们同时开设。本书的第六、七、八章是按学生不选修“容错与诊断”、“纠错码”、“并行处理计算机结构”和“计算机系统性能评价”等课来写的。本课程的参考教学时数为80～100学时。

本教材是在国防工业出版社1981年出版的《计算机系统结构》的基础上修编而成的。参加当时编写的有：西北电讯工程学院的苏东庄、张志华、李学干、马玉珍；清华大学的金兰；华东师范大学的张东韩和唐培艺，由苏东庄主编。担任主审的是清华大学的房家国和薛宏熙。参加此次修编的除苏东庄外，还有李学干、金兰（第七章）、梁新来（第六章）、马玉珍（第八章），李学干承担了全书的文稿整理工作。

这里特别要感谢本版主审人薛宏熙同志，他进行了认真细致的审稿，并提出了宝贵的修改意见。凡在本书编写和审阅过程中提出过意见和建议的同志，在此一并向他(她)们表示衷心的感谢。

本书在编写中得到了西北电讯工程学院教材科、图书馆的大力支持。有关领导部门、电子工业部教育局以及研究所和兄弟院校的很多领导、专家、教授和同志们对本书的编写曾给予了热情的鼓励，并提出过许多宝贵的意见。我们在此表示诚挚的感谢。

恳切希望读者继续对本书的缺点和错误予以指正。

编　者

1984年

目　录

前　言

第一章　计算机系统结构的基本概念

习题

主要参考文献

第二章　指令与寻址

习题

主要参考文献

第三章　控制方式

第四章 输入输出系统

第五章 存贮体系

第六章 可靠性技术

第七章 多机系统

第八章 描述与评价

第一章　计算机系统结构的基本概念

本章首先讲述计算机系统层次结构和本书所用的计算机系统结构的定义，叙述系统结构、组成与实现的定义和三者的关系；而后简述程序可移植性要求以及应用与器件对系统结构的影响；并稍许讲讲计算机设计自动化。目的是在学习后面各章之前，能对计算机系统结构有一个基本的了解。

§1　计算机系统层次结构

1.1　软件与硬件

计算机系统是由硬件和软件组成的。

一方面，软件随着计算机系统使用等的需要而不断扩大。明显的例子是操作系统的日益庞大，因为使用者日益希望只需发出少数几条命令，就能使计算机系统执行复杂的动作。另一方面，软件和硬件的分界面又在动态地不断变化着。例如，早期的机器没有乘、除法指令，乘、除法运算是通过子程序（软件）实现，后来的机器则把这部分软件"硬化"，即用硬件来实现乘、除法运算。前几年，同一种运算或操作（例如浮点运算、字符行运算等），在微型机和一般小型机中是用软件来实现的，而在大、中型机中则是用硬件来实现。又如，一般机器只有整数（定点数）、实数（浮点数）和逻辑数的数据表示，复杂的数据结构，如向量、数组等只能用软件实现，但目前已有一些机器把这些软件硬化，使机器具有向量、数组硬件表示及相应的向量、数组运算。其它如中断处理、存贮管理等的软、硬功能分配也是随不同时期、不同机器而动态地在改变着，这个特点是学习本课程时要始终注意的。

由上可见，软件和硬件在逻辑功能上是等效的。由软件实现的操作（如编译操作、操作系统的基本操作等）在原理上是可以硬化成由硬件来实现；同样，由硬件实现的操作（如某条机器指令所完成的操作，条件码的置定等等）在原理上也是可以用软件的模拟来实现。由自动机的基本理论可知，只要机器有"相减"及"转移"这两种指令就可用于算题、求解。这样，具有相同功能的计算机系统，其软、硬件功能分配可以在很宽的范围内变化，如图 1.1 所示。选择什么样的分配比例，主要应取决于在现有硬件状况（主要是逻辑和存贮的器件状况）下的性能价格比。提高硬件功能的比例可以提高解题速度，减少所需的存贮容量，但会提高硬件的成本以及降低硬件的利用率和计算机系统的灵活性与适应性；相反，提高软件功能的比例可以降低硬件的造价，提高灵活性和适应性，但解题速度要下降，软件设计费用和所需的存贮器容量要增加。目前计算机系统的软、硬件功能分配对现有硬件状况、运算方法和解题算法是适应的。但这决不是说只能如此，尤其是在今后硬

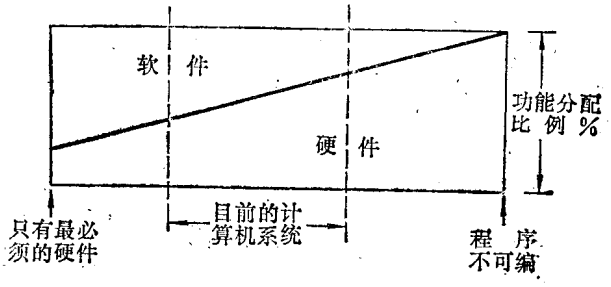

图 1.1　计算机系统的软、硬件功能分配

件价格随着超大规模集成电路技术的发展而日益下降，将会破坏目前的这种适应性。

那么，对于由紧密相关的硬件和软件组成的计算机系统，我们可以用什么样的观点，从整体上去认识和分析它呢？一种观点是把计算机系统看成是按功能划分的多级层次结构。这和软件的发展过程是一致的。

1.2 计算机系统的多级层次结构

早先的机器语言（即机器指令系统）由于受电子器件数量的限制，只能是最基本的和简单的，使用者必须编写出用二进制表示的程序。这很不方便，也很不适应于解题的需要及计算机使用范围的扩大。因此，在五十年代出现了符号式程序设计语言(汇编语言)。尽管汇编语言程序的每个语句和机器指令基本上仍是一一对应，但却方便多了。这样对程序员来讲，可以把汇编语言看成是改进了的机器语言，只是没有实际的机器硬件与它直接对应，它的实现是经汇编程序翻译成真正存在的机器语言。我们可以把这想象成是在实际机器级（其机器语言由硬件实现）之上出现新的“虚拟”机器级，其机器语言为汇编语言，如图1.2所示。整个汇编语言（L2）程序（源程序）先经汇编程序变换成等效的机器语言（L1）程序（目标程序），而后再在实际机器上执行目标程序以获得结果。这样一种先把源程序变换成目标程序，而后再在机器上执行目标程序以获得结果的技术称为翻译。

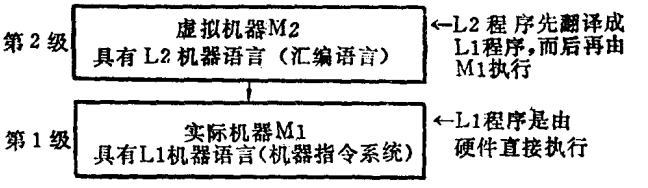

图 1.2 汇编虚拟机器的实现

由于汇编语言的语法、语义结构仍然和机器语言的基本一样，而且和解题所需的仍然差别较大，因此紧接着汇编语言，又出现了面向题目的高级语言，如FORTRAN、ALGOL等。同理，我们可以设想成在汇编语言级之上又出现了高级语言级，它的实现是先经编译程序把高级语言程序翻译成汇编语言程序（或是机器语言程序），逐级向下地实现，如图1.3所示。程序员在用高级语言编制程序时，就如同面对着其机器语言为高级语言的这样一种虚拟机器。

这种虚拟机器的垂直式层次概念，对于理解各种语言的实质及其实现是有好处的。高级

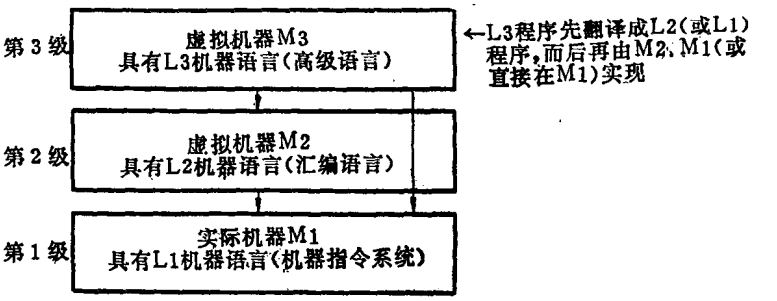

图 1.3 高级语言虚拟机器的实现

语言程序先翻译再执行的这样一种实现技术，虽然已经持续了二十多年，但它是立足于当时的硬件条件（指硬件状况只适合提供简单的机器语言）。显然，随着我们对程序设计语言和编译技术认识的深入以及超大规模集成电路技术的发展和应用，我们应该探索并一定能够找到实现高级语言的更好的办法。

这种层次概念还可引伸、应用于机器内部。对于采用微程序控制的机器，每条机器指令对应一串微指令(一段微程序)，而每条机器指令是通过这一串微指令的执行来实现。这样，我们又可以把图 1.2 中的实际机器级分解成如图 1.4所示的二级层次结构。微指令 所执行的是最基本的操作，如寄存器间的数据传送、数据经过加法 器 或 乘 法器、主存与寄存器间的数据传送等等。但是，在这里并不是如同汇编语言或高级语言程序那样，先把高一级机器语言的程序翻译成低一级语言的等效程序，然后再执行它，而是每条机器指令由一串微指令实现，实现完某条机器指令后，再由程序内的下条机器指令控制，进入实现它的另一串微指令。这种实现过程称为解释，如图 1.5 所示。这个过程也可理解为在 Mi 机器（这里是微程序机器 M0）上，用一串 Mi 机器指令(这里是微指令)的执行来仿真比它高级的机器的一条语句(这里是比M0 高一级的 M1 机器指令)。用微程序解释机器指令意味着把软件技术引入到传统机器的内部，这对提高硬件的规整性，以适应大规模集成电路技术的需要是很有益处的。

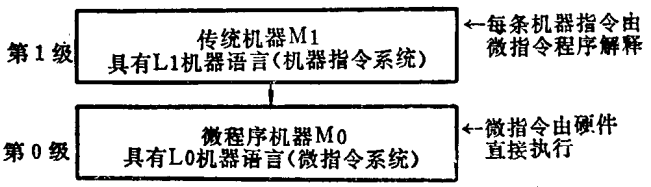

图 1.4　机器指令由微指令程序解释

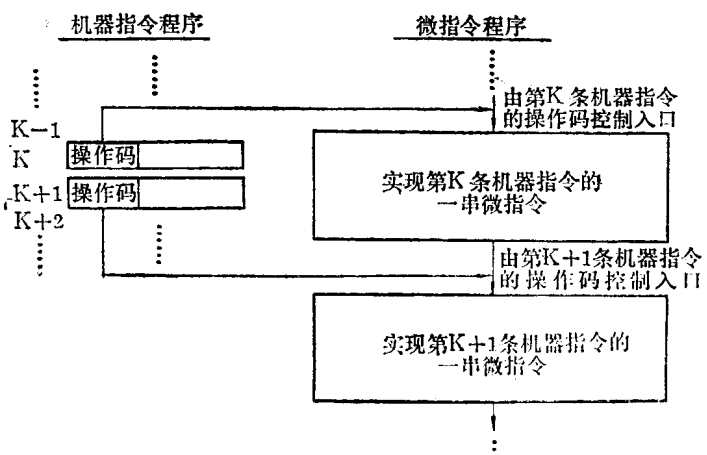

图 1.5　用微指令程序解释机器指令

翻译和解释是语言实现的两种基本技术，对于微程序控制的机器，其高级语言的实现，这两种技术都要用到。即先把高级语言程序经编译程序翻译成机器（传统的）语言程序，而后再经微程序对每条机器指令进行解释来实现。还要注意到，由高级语言程序变换成机器语言程序的过程中也不是全部只采用翻译技术，而往往是翻译和解释这两种技术并用。一般是先把高级语言源程序翻译成易于执行的某种中间形式，而后再用机器语言程序解释它。至于机器语言程序的每条机器指令，又需经微程序解释。

上面分析了可以把计算机系统看成由高级语言虚拟机器、汇编语言虚拟机器、传统机器语言机器和微程序语言机器所构成的多级层次结构。那么，操作系统在这个层次结构中应处于什么位置呢？这个问题比较复杂。

操作系统实质上是传统机器的引伸，它提供了传统机器所没有的，但为汇编语言和高级语言的使用和实现所需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

贮体系以及多道程序和多重处理所用的某些操作等等。这些操作（可以看成是操作系统的指令系统，其中包含作业控制语言等）和数据结构在已有的机器上大多是经机器语言程序的解释来实现。从上述分析看，操作系统级应是位于传统机器级之上，汇编语言机器级之下。当然，有些机器语言指令，如某些具体的 I/O 操作指令是要被操作系统“挡”住的，但大部分机器语言指令，如运算类指令等等，却是原样不动地穿过操作系统级，而且也包括在操作系统级的指令系统之内。

但是从另一方面看，操作系统还具有提高计算机系统的效率以及控制汇编语言、高级语言等的实现和作业的运行等功能。从这点看，似乎又应该把操作系统置于前述按语言结构划分的层次结构之外，并让它能作用于几乎所有的级。

另外，目前已有一些机器，尤其是今后的机器，它们的操作系统不是用汇编语言编写，而是用高级语言(不是用面向解题的算法语言，而是用面向系统软件的高级语言，如 C语言)编写。这样，操作系统级似乎又应该在高级语言机器级之上。

总之，操作系统级的位置是不能简单地指明的。不过从操作系统的基本功能来看，把它置于传统机器级与汇编语言机器级之间，基本上是能反映出其主要作用的。因此，用图 1.6 的多级层次结构来认识包括操作系统的计算机系统还是适宜的，图中每级对应一类机器（各有其自己的机器语言）。在这里，“机器”被定义为能存贮和执行程序的算法和数据结构的集合体。各级机器的算法和数据结构的实现方法不同：M0 是硬件实现；M1是微程序(固件)实现；M2到M5是软件实现。我们称由软件实现的机器为虚拟机器，以区别于由硬件或固件实现的实际机器。要注意，机器的实现和题目的得到解答不是一回事，后者是要从你用的那级开始逐级变换，直至到达最低级，它需经硬件实现才能得到；而前者主要表现为如何把该级的程序或是翻译成比它低一级的语言的程序，或是由低一级的程序所解释。因此，相邻级的语言的语

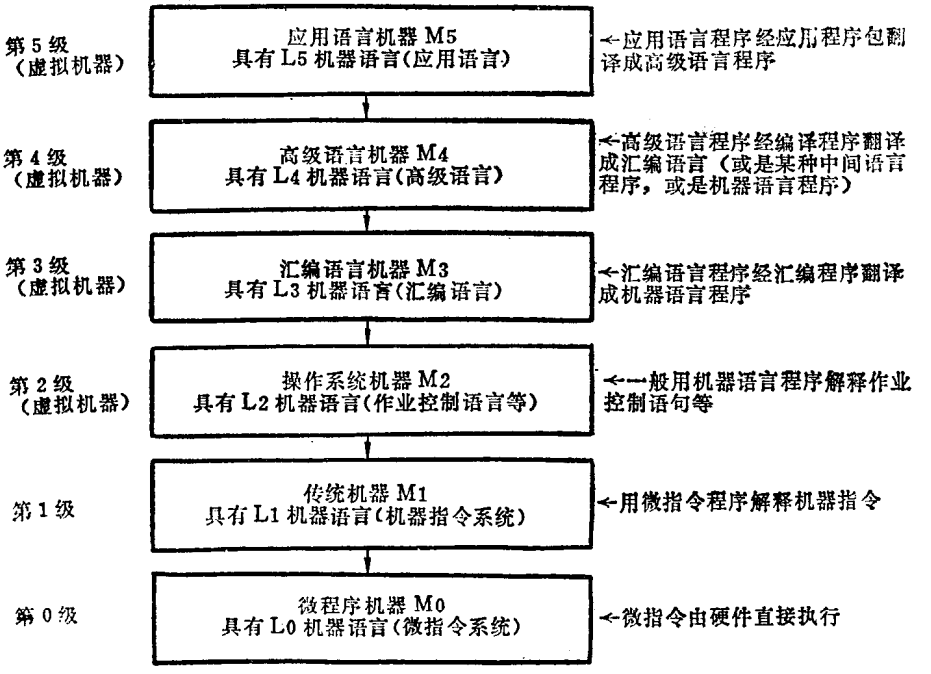

图 1.6　计算机系统的多级层次结构

义结构差别不能过大，以便于翻译或解释。当然，有的级的程序可能翻译成更低级的程序，例如高级语言程序可能直接翻译成机器语言程序；有的级的语言可能被更低级的程序所解释，例如操作系统的某些命令就可能由比它低二级的微程序解释。操作系统机器的这种例子表明虚拟机器也不一定非得全由软件实现，其中有些操作也可能是固件或硬件实现。循此，为什么就不能设想直接由微程序或硬件实现高级语言机器，或是用固件实现操作系统机器呢？

采用何种实现方式，要从整个计算机系统的效率、速度、造价、资源的使用状况等等方面全面考虑。而且要软件、硬件(包括固件)综合平衡，这点一定要认识到。对计算机系统的使用者，除了关心他直接接触的那级机器(或是应用语言机器、或是高级语言机器)的状况外，当然还得关心他所选用的计算机系统是否能提供更好的性能价格比。

对于不采用微程序控制的机器，则只是没有第0级，而是直接由硬件实现机器指令系统。除此之外，上述关于计算机系统层次结构的分析对它都是适用的。

也可以考虑在第0级之下再分级。例如有的微程序机器采用二级微程序（称为微程序与毫微程序）去解释机器指令。每条微指令不是直接由硬件执行，而是由比它低一级的毫微程序解释。

上面我们分析了计算机系统可以用层次结构来认识，那么，如何从层次结构来定义“计算机系统结构”呢？

§2 计算机系统结构的定义

计算机系统结构（Computer Architecture）这个名词从七十年代以来已被广泛使用，但至今对其定义仍有不同的说法，这和前述软件和硬件的分界面在不断变化着，以及器件技术的迅速发展都是有关的。本书所用定义是从计算机系统层次结构和设计过程引出的。

2.1 计算机系统结构

Amdahl等人于1964年在介绍IBM360时提出，计算机系统结构是从程序员所看到的计算机的属性，即其概念性结构与功能特性。

这个说法至今仍被不少人所接受，然而按计算机系统的层次概念，从不同级的程序员看，计算机是具有不同的属性的。

例如，从使用高级语言（如FORTRAN）的程序员看，NOVA机和PDP-11机，DJS-1000机和DJS-2000机就几乎没有什么差别，具有相同的属性。或是说，这些机器之间本来存在的差别是高级语言程序员所“看不见”的，所不需要知道的。这种本来是存在的事物或属性，但从某种角度看又好象不存在的概念称为透明性概念。透明性概念对于理解软、硬件之间的界面有好处。当然，对同一种高级语言，在各个厂家实质上都是有些差异的，各有其“方言”及某些不同的规定。一般做不到能在某个型号的机器上运行的高级语言程序，毫不修改地能在另一型号的机器上运行，而且在编制高级语言程序时，往往还要考虑所用机器的字长及主存容量等。但这些机器的概念性结构及功能特性，却是高级语言程序员所看不见的，即对高级语言程序员是透明的，图1.7说明了这一点。

然而，从机器语言程序员或汇编语言程序员看，上述两种计算机的属性却是不同的，主要表现为指令系统（如操作类型和寻址方式）和输入/输出设备连接方式的不同。

这样，程序员从不同的层次（级）看，计算机的属性是不同的。这个属性就是计算机系统不同层次的界面。界面应是明确定义的，而且界面上、下的功能分配也应是明确的。“系统结构”指的就是每个级的界面上、下的功能分配和对界面的定义。因此，各级都有它的系统结构。

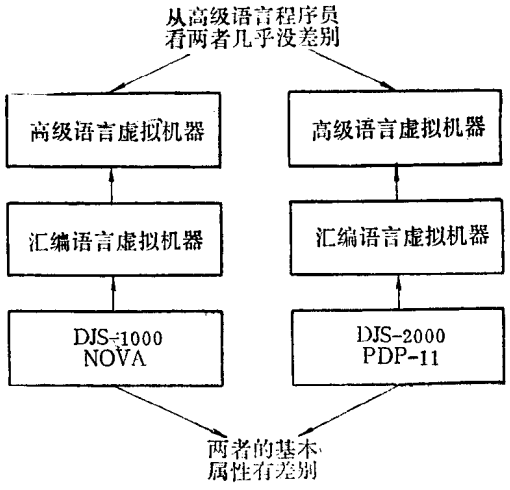

图 1.7　从高级语言程序员看的计算机属性

本书所讲的计算机系统结构，指的是图 1.6 中传统机器级（下面简称机器级）的系统结构。其界面之上的功能指的是所有软件的功能，即包括于图 1.6 中操作系统级、汇编语言级、高级语言级和应用语言级的所有功能。对于目前的通用计算机系统，其中有程序设计语言、终端命令语言、数据库描述与管理语言、作业控制语言以及这些语言的编译和解释系统；还有编辑程序、分类程序、实用程序、信息检索程序、逻辑资源管理软件（如数据库管理、文件管理、虚存管理、远程及网络管理等）和物理资源管理软件（如辅存和主存空间管理、处理机管理、其它物理设备的管理等）等。界面之下的功能指的是所有硬件和微程序的功能。这个界面是软件与硬件（包括微程序）的交界面，如图 1.8 所示。如果没有特殊说明，以后书中的“系统结构”和“计算机系统结构”是同义语。

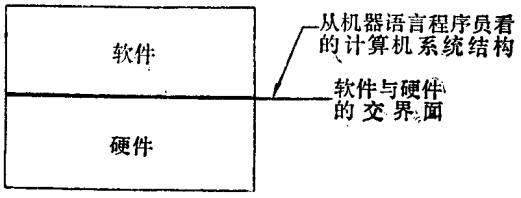

图 1.8　计算机系统结构是软/硬件交界面

这样，计算机系统结构研究的是软、硬件功能分配以及对机器级界面的确定。具体来说，此界面指的是由机器语言设计者或编译程序设计者所看到的机器物理系统的抽象或定义。它是程序设计者（或是编译程序生成系统）为使其所设计（或生成）的程序能在机器上正确运行，所需看到和遵循的计算机属性。它不包括机器内部的数据流和控制流、逻辑设计和器件设计等，这些属于我们稍后要讲的计算机组成和实现。

这些属性包括数据是如何表示（即数据表示）、如何被访问的（即寻址方式），能对这些数据进行什么样的运算和如何控制这些运算的执行等（即指令系统）。这些属性不能只是与实际执行指令有关，即不能只是与处理机有关（虽然计算机的其它部分是可经处理机看到），而应是工作于机器级的程序设计者所看到的机器的所有部分，包括处理机、存贮系统、I/O联结方式和中断机构等，即 Amdahl 等人提出的概念性结构与功能特性。对于目前的通用型机器，一般包括：

数据表示；

寻址方式（包括最小寻址单元和地址运算等）；

寄存器定义（包括操作数寄存器、变址寄存器、控制寄存器等的定义）；

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）；

中断机构；

机器工作状态（如管态和目态等）的定义和切换；

机器级的 I/O 结构（包括对 I/O 设备的访问方式，I/O数据的“源”、“目的”与所传送的数据量，I/O 操作的结束与出错指示等）；

以及对信息保护的支持；等等。

按此，DJS-1000 计算机与 DJS-2000 计算机，NOVA 机与 PDP-11 机当然有不同的机器级界面，即有不同的计算机系统结构。

应该说，计算机系统结构这三十多年来的进展不大，目前绝大多数机器的计算机系统结构仍然没能摆脱冯·诺依曼型的范畴。冯·诺依曼型计算机系统结构指的是冯·诺依曼(Von Neumann) 等人于 1946 年提出来的结构，它由运算器、控制器、存贮器和输入/输出设备组成，如图 1.9 所示。

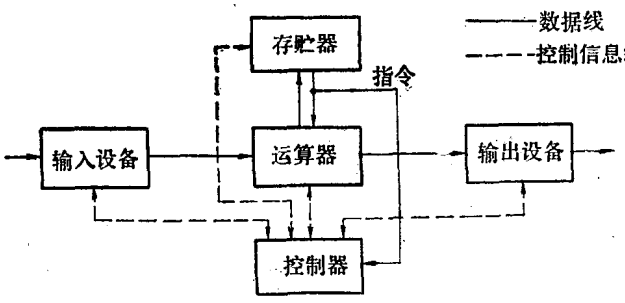

图 1.9　冯·诺依曼型机器的结构

冯·诺依曼型计算机系统结构的基本特点可归结为以下几点：

（1）存贮器是按地址访问的顺序线性编址的一维结构，每个单元的位数是固定的。机器的运算速度和访主存的次数关系很大。

（2）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，它具有的算术运算是最基本的运算。地址码指明操作数的地址。操作数的数据类型由操作码确定，操作数本身判定不了它是何种数据类型（如是定点数，还是浮点数等等）。

（3）指令在存贮器中是按其执行顺序存贮，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，指令计数器一般顺序加“1”。虽然执行顺序是可以根据运算结果改变的，但是解题算法仍然是，也只能是顺序型的。

（4）在存贮器中指令和数据同等对待，从它们本身是区别不了的。指令同数据一样可以送到运算器进行运算，即由指令组成的程序是可以修改的。

（5）机器以运算器为中心，输入／输出设备与存贮器之间的数据传送都途经运算器；运算器、存贮器、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。

（6）数据以二进制编码表示，采用二进制运算。

冯·诺依曼等人提出的这种结构在当时是很了不起的，它奠定了计算机发展的基础。虽然当时的计算机是为了解非线性微分方程而设计的，但基本上采用这种结构的计算机，却成功地应用于其它各种数值解题以及诸如信息处理系统、事务数据处理和工业控制等的广泛领域，这是冯·诺依曼等人当时所未想到的。当然，当初设计时既没考虑到要采用高级语言，也没顾及会有操作系统以及很多应用领域的各种要求，由此引起的矛盾到三十多年后的今天就显得日益突出了。

2.2　计算机组成与实现

2.2-1 计算机组成

对计算机组成（Computer Organization）的定义，同样有不同的说法。本书讲的计算

机组成指的是计算机系统结构的逻辑实现，包括机器级内的数据流和控制流的组成以及逻辑设计等。它着眼于机器级内各事件的排序方式与控制机构、各部件的功能以及各部件间的联系。

我们可以从计算机系统结构与计算机组成的对比来认识它。例如，指令系统属计算机系统结构，而指令的实现，如取指令、取操作数、运算、送结果等事件的操作及其排序则属计算机组成。这样，具有相同结构（如指令系统）的计算机，其组成却可由于诸如速度要求不同等各种因素而不同，且可能差别很大。例如，实现相同的系统结构，组成中的取指令、取操作数、运算、送结果等可以是顺序执行的，也可以使这四个事件重迭地进行，以提高机器的速度。我们再举一些例子，例如，决定是否有乘法指令属计算机系统结构，但乘法指令是用专用乘法器实现，还是经加法器用重复的相加和右移操作实现，则属计算机组成。这主要取决于计算机所要达到的速度以及在程序中乘法指令的执行频度和所采用的乘法运算方法等。又如，对主存系统，主存空间大小（地址码位数）与编址方式（如按位、字节、字访问等）的确定属计算机系统结构，而为要达到所定的性能价格比，主存速度应多快，在逻辑结构上需采用什么措施（如多体交叉存贮等），则属计算机组成。

计算机组成的设计内容是按所希望达到的性能价格比，如何最佳、最合理地把设备和部件组成计算机，以实现所定的计算机系统结构。计算机组成这三十多年来的主要进展是在如何提高速度方面，着眼点在于提高操作的并行度、重迭度以及功能的分散和专用功能部件的设置。

在计算机组成的设计中，所要确定的方面包括：

数据通路宽度。(这点我们稍后再讲)。

专用部件的设置。设置什么样的专用部件（如乘除法专用部件、浮点运算部件、字符行运算部件、地址运算部件直至I/O处理机等），设置多少个专用部件，这都取决于所要达到的机器速度、专用部件的使用频度和所需造价等。

各事件操作对部件的共享程度以及各功能部件的并行执行。它们决定了完成一段程序或一条指令的各个事件、操作是串行执行，还是并行执行。(这些，我们在第三、七章还要详述)。

缓冲、排队技术和预估、预判技术的应用。(这些，我们在后几章中再讲)。

可靠性技术。(在第六章详述)。

所用器件的集成度和速度。

所用的组成方式。组成方式指的是事件、操作的排序机构是采用硬联技术，还是微程序技术，以及机器内是采用单个处理机，还是采用多个功能分布的处理机等等。

微程序技术的出现和被广泛应用是计算机组成技术的重大进展，下面再稍加叙述。

2.2-2 微程序技术

我们在前面（参看图1.4）已经讲过，采用微程序控制的机器是在微程序机器M0上，用一串M0级机器指令（即微指令）的执行来仿真传统机器级M1的一条机器指令。我们称M0级机器为微程序宿主机。采用微程序组成方式的机器实质上是在某个宿主机上，用仿真方法实现计算机系统结构的指令系统。

可以说，微程序技术的出现和采用，最初是为了使控制器规整，以便于设计和生产。它之所以能够使控制器（参看图1.10）规整化是由于把存贮器技术引入到控制器内（有人因此称之为存贮逻辑），从而把硬联控制器网络的复杂逻辑联结，转变为控制存贮器中存贮的