第5章 chapter 5

系统思维

计算系统思维的要点是：**通过抽象，将模块组合成为系统，无缝执行计算过程**。更准确地说，计算系统思维的要点是通过巧妙地定义和使用计算抽象，将部件组合成为计算系统，该系统流畅地运行所需的计算过程，为用户提供应用价值。模块就是精心设计的部件。

计算机科学研究的计算过程需要在计算系统中运行。前面4章讲述的数字符号操作、计算模型、计算逻辑、算法、程序、网络，归根结底都需要通过计算系统实现。一个计算系统由多个部件组合而成，支持一个或多个计算过程的运行。

计算系统包括抽象的计算系统或真实的计算系统。抽象计算系统的例子包括各种计算模型，如图灵机、自动机等。真实的计算系统形成具体的产品和服务，包括计算机硬件系统（如微处理器芯片、内存芯片、硬盘、网卡等部件，以及智能手机、笔记本电脑、服务器等整机）、计算机软件系统（如操作系统、数据库管理系统、互联网浏览器、办公软件系统等）、服务系统（如互联网搜索引擎、微信社交网络、淘宝电子商务系统等）。

研究、理解和使用计算系统的最大挑战是应对系统的复杂性。破解系统复杂性挑战的主要思维方法是抽象化或抽象，其英文都是abstraction。

实例：微信系统的复杂性。当仔细考察一个计算机系统时，往往会惊叹其内部的复杂性。粗略地计算一下腾讯微信系统涉及的晶体管个数，可以稍微认识一下该系统有多么复杂。微信系统有上亿用户同时在线。假设每个用户使用一台智能手机或笔记本电脑这样的终端设备，那么至少有上亿个处理器芯片在同时工作，执行着微信的计算过程。这还不包括微信云端系统以及互联网系统。

每个处理器芯片大约包括20亿个晶体管（如苹果公司iPhone 6使用的A8处理器芯片）。如果将每个晶体管看成一个家庭住房，晶体管之间的连线看成道路（从高速公路一直到楼道），那么一个芯片的电路图比全中国的米级地图（显示了全中国每一套住房以及全部的道路）还要复杂，微信系统则比全世界的米级地图复杂得多。

显然，理解和设计微信系统不能从每一个晶体管做起，更不能靠简单地堆砌20亿亿个晶体管。在理解和设计微信系统时，需要有一套特殊的方法，称为计算**系统思维**，它包括三个利器，即**抽象化、模块化、无缝衔接**。系统抽象是最本质的考虑，模块化与无缝衔接是对抽象的补充。

5.1 从一个实例看计算系统

1. 抽象化

系统思维的第一个利器是**抽象化**(abstraction),即从多个层次(角度)理解一个系统,每个层次仅考虑该层次特有问题,忽略其他问题;并用一套抽象概念和方法统一地处理该层次所有的计算过程,解决这些特有问题。

先考察一个看起来很简单的微信消息通信系统实例。可以通过三个抽象层次理解微信系统,见图 5.1。

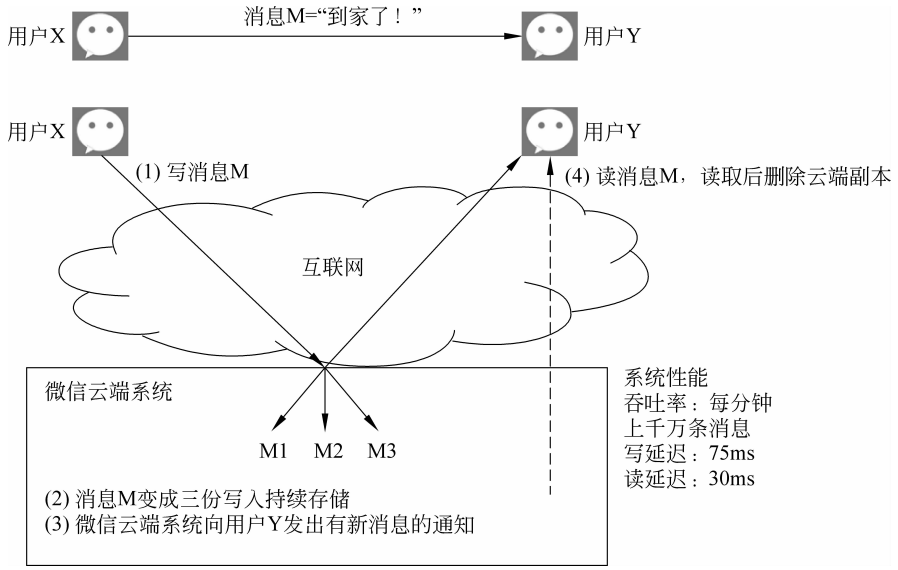

图 5.1 微信网络消息传递的三种抽象:用户层、应用层、系统层

最上一个层次可称为用户层。这里只有三个抽象概念:用户、客户端设备、消息。每个用户涉及的基本计算过程也只有三个:浏览消息、收消息、发消息。一个完整的典型计算过程是:用户 X 用他的智能手机客户端设备向用户 Y 传递一个消息"到家了!",而用户 Y 则用她的平板电脑客户端设备收到消息"到家了!"。

用户层需要考虑的特有问题比较少,包括"如何浏览消息""如何收消息""如何发消息""消息可以包括自拍的视频吗""如何加好友""发完消息后多久好友才能收到"等。更加罕见的问题也包括什么样的终端设备、什么样的网络支持微信。至于微信系统如何实现消息的处理和传递、如何保证消息不丢失、不发错,以及微信系统能够支持多少用户同时在线这类问题,用户层不用考虑。由于一个用户的朋友圈可能只有数十个或数百个其他用户,比起需要考虑 20 亿亿个晶体管的蛮力方法,理解微信系统的复杂性大幅度降低了。

第二个层次可称为应用层。这个层次的微信应用系统实现微信消息的处理和传递。

它主要涉及五个抽象概念：用户、客户端设备、客户端应用软件、消息、微信云端系统。实现用户 X 向用户 Y 传递消息 M(上面例子中 M="到家了!")的典型的计算过程如下。

步骤 0：用户 X 使用客户端设备上的客户端应用软件(俗称 App)编写并发送消息 M。

步骤 1：用户 X 的客户端通过互联网(包括移动互联网)向微信云端系统写消息 M。

步骤 2：云端系统收到消息 M 后，复制成三份写入持续存储(硬盘或固态存储)。

步骤 3：云端系统向用户 Y 的客户端发送有新消息的通知。

步骤 4：用户 Y 的客户端从微信云端系统读取消息 M，然后删除云端持续副本。

理解和设计应用层的最大挑战是：如何支持数千万个上述计算过程同时执行，并保证消息的及时传递。应用层必须关注如何实现消息的处理和传递，如何保证消息不丢失、不发错，以及微信系统能够支持多少用户同时在线这类问题。但是，应用层较少考虑这些计算过程如何由系统软件和系统硬件实现。例如，为了保证消息 M 不丢失，应用层的云端系统需要执行将 M 存储到 M1、M2、M3 的三副本持续存储操作。但是，应用层并不需要考虑如何实现这个三副本持续存储操作，也不关心 M1 存储到了哪台服务器上、在哪块硬盘中。

第三个层次可称为系统层。这个层次涵盖系统软件和系统硬件，包括客户端和云端系统所涉及的硬件、操作系统软件、数据库软件、应用框架软件等。这个层次关注如何让微信系统能够有效地及时地服务上亿用户，如何让微信应用系统能够每分钟传递上千万条消息，同时能够保证读/写一条消息的延迟分别控制在 30ms 和 75ms，如何控制系统成本等问题。它也需要解决如何有效地实现三副本持续存储的问题，包括 M1 存储到了哪台服务器上、在哪块硬盘中等系统细节。

如有必要，还可以继续考虑更具体的层次，一直到最终的电路层次甚至晶体管层次的抽象。

2. 模块化

系统思维的第二个利器是**模块化**(modularity)，即理解每一个系统是由多个模块按一定规则连接组合而成的。也可以反过来理解：一个系统如何分解成多个模块的组合。有时候模块也被称为子系统。模块化可用来应对复杂性。理解一个系统可以通过理解每一个模块，以及这些模块的组合规则，得以简化。

例如，微信应用系统大致上由三部分(即三种模块)组成：①运行在数亿用户的客户端设备(桌面设备和移动设备)上的微信客户端软件系统；②运行在腾讯公司的微信云端系统；③连接这两者的互联网，包括移动互联网。在理解微信系统时，可以首先理解客户端设备、微信云端系统、互联网这三者的分工合作以及相互间的接口，然后进一步理解客户端设备、微信云端系统、互联网的应用层、系统层，从而降低理解微信系统的复杂性。在理解某个模块(如云端系统)时，可以不用考虑另一个模块(如客户端设备)的内部细节。例如，微信云端系统如何有效地实现三副本持续存储，这个问题与微信客户端的内部细节无关。

3. 无缝衔接

系统思维的第三个利器是**无缝衔接**(seamlessness),即让计算过程在全系统中流畅地运行,不出现或少出现缝隙和瓶颈。一个系统包含多个部件(又称子系统),一个计算过程在运行中可能涉及多个子系统,但子系统之间的过渡不应该出现缝隙和瓶颈。如果做不到完全的无缝,至少也要控制瓶颈,使系统的功能和性能满足用户体验需求。

例如,微信系统传递消息的计算过程涉及多个计算机硬件、多个操作系统、多个网络,这些环节或模块需要实现无缝衔接,使得整个微信系统同时流畅地执行上千万个传递消息的计算过程。腾讯公司不可能百分之百地保证所有用户的每一条消息都能够被及时送达,因为互联网和移动互联网不在它控制之中,客户端设备的硬件和操作系统也不在它控制之中。即使不能百分之百的保证,微信系统也需要精心设计其云端系统和微信客户端应用软件系统,使得这两者不成为微信服务的瓶颈。为此,微信系统设定并实现了具体的性能指标,即微信云端系统在支持每分钟处理上千万条消息的前提下,每条消息在微信云端系统中的写延迟不超过 75ms、读延迟不超过 30ms。

上述微信系统实例显示,不论是什么系统,都需要思考以下三个本质的问题。

- 系统需要什么样的抽象?
- 系统由哪些模块如何组合而成?
- 系统的部件如何无缝衔接,流畅地执行计算过程?

5.2 计算系统思维要点

计算机科学在多年的发展中,产生出了一套系统思维方法:**通过抽象将部件组合成为系统**。构造计算系统是有条理、有门道的,不是随意而为;系统思维产出一个系统整体,而不是一堆部件的罗列堆砌;计算系统往往用一套统一的方法来支持万千应用场景,而不是每个场景对应一个计算系统。在英文语境里,往往用 systematic(系统地),而不是 ad hoc(随意地),来形容这种系统思维。这些条理、门道往往体现为系统抽象。

同时,系统性不是僵化的教条,不是用于阻碍创新,更不是遏制应用的丰富性和多样性。构造计算系统是有条理、有门道的。但是具体是什么条理、门道就是创新的重要目标。这使得理解和设计计算系统富有令人激动的挑战性。系统思维不是让计算系统设计者机械地执行一些教条方法,而是要求创造性和综合性。这也是为什么在英文中,计算系统设计者被称为 **architect**(直译为建造师或建筑师,信息技术领域则称为**架构师**)。

计算机科学在过去 70 年中发展出来的系统思维方法还是很有成效的。全球只有几百万名计算机科学技术的专业人员,却支持着数十亿用户,使用近百亿台计算设备,体验万千种应用服务的价值。当然,这套系统思维方法也仍有许多不足之处。例如,与生物界相比,计算系统界的多样性欠缺,使得一个病毒可迅速影响上亿台计算设备。

本书初步介绍系统思维的抽象化、模块化、无缝衔接三个要点。它们不是三个孤立的东西,而是系统思维不同角度的体现。其中,系统抽象是最本质的考虑,模块与无缝衔接是补充。这三个概念既是动名词也是名词,前者体现方法,后者体现该方法的产物。

5.2.1 抽象化

计算抽象既是计算机科学最重要的方法，也是最重要的产物。作为动名词的抽象也被称为**抽象化**，而抽象化的产物也称**抽象**，两者对应的英文都是 abstraction。

抽象化的要点是：一个系统可从多个层次（或多个角度、多个视野）理解，每个层次仅仅考虑有限的、该层次特有的问题，并用一套精确规定的抽象概念和方法，统一处理该层次所有的计算过程，解决这些特有问题①。其他问题则留给其他层次考虑。该层次甚至看不见这些其他问题，因此也可以忽略与这些问题相关的所有细节。换句话说，抽象化和抽象具备三个性质（称为**抽象三性质**）。

- **有限性**：抽象化意味着从多个层次（或多个角度、多个视野）理解一个计算系统，每个抽象仅仅考虑一个层次的有限的特有问题，忽略其他层次，忽略同一层次的其他问题。
- **精确性**：抽象化的产物是一个计算抽象，它是一个语义精确、格式规范的计算概念。
- **通用性**：计算抽象强调用一个通用抽象代表多个具体需求。它意味着用统一的一套方法处理该层次所有的计算过程，解决该层次的特有问题。它不是只对特定的具体问题实例有效，而是可以触类旁通、用于其他实例。这也被称为抽象的**泛化**（generalization）能力。只对某个实例有效的抽象，不是好的抽象。

抽象化是所有科学技术学科共有的方法。那么，什么是计算机科学的抽象化特色呢？就是可自动执行的、比特精准的信息抽象，即数据以及对数据的操作，包括存储操作、运算操作、通信操作。典型的**存储操作**包括将数据存放在某个地方（例如内存），或者从某个地方取出来放在寄存器中。该“地方”的名字称为存放该数据的**地址**。基本的**运算操作**包括加、减、乘、除、与、或、非等算术逻辑运算。典型的**通信操作**包括将数据从一个地方（例如硬盘）传递到另一个地方（例如内存）。数据传递到显示器时，相应的信息就被显示出来了。

将某一类数据及其操作（存储操作、运算操作、通信操作）合起来称为**数据抽象**（data abstraction），也称数据类型（data type）或数据结构（data structure）。通常，针对这些数据抽象的多个操作步骤组合起来才能解决一个问题。控制多个步骤如何组合起来实现计算过程的操作抽象称为**控制抽象**（control abstraction），它确定某个步骤何时激活。数据抽象和控制抽象是计算机科学的抽象特色的重要体现。

1. 数据抽象

下面简要介绍七种常见的数据抽象，包括比特、4 比特、字节、字、指针、文件、数据。它们也体现了基本上从小到大的七个抽象层次。

---

① 由于抽象化和抽象在计算机科学中使用普遍，来源众多，含义丰富，反而没有一个众所周知的、达成广泛共识的统一定义。例如，关于“系统可从多个层次理解”这个思想，即抽象化层次（level of abstraction）概念，牛津大学的 Luciano Floridi 教授将它归功于图灵在 1950 年发表的图灵测试论文。本书对抽象化和抽象的定义综合了多个来源。

(1) **比特**(bit)是最基本二进制数位。单个比特取值可为 0 或 1,其涉及的运算操作主要是布尔逻辑运算。

(2) **4 比特**(hexadecimal,简写为 hex,也称十六进制数,或**半字节**)是用四个比特合起来形成的一个十六进制数位。它可以取 16 个值之一,从 0 到 F。由于十进制不够用,人们增加了从 A 到 F 六个字符来表示从 10 到 15 的六个额外数值。这 16 个值的二进制表示是 0:0000,1:0001,2:0010…9:1001,A:1010…F:1111。

(3) **字节**(byte)是用 8 个二进制数位组合而成的。

(4) **字**(word)可由 8、16、32、64 或 128 个二进制数位组合而成。字往往是算术逻辑运算的基本单位。一次运算通常会产生一个结果字,而不仅仅是一个结果字节或结果比特。人们一般所说的"32 位计算机"或"64 位计算机",通常指的是该计算机采用了 32 位(32bit)或 64 位(64bit)的字,32bit 或 64bit 是计算机的**字长**。早期的计算机也采用 40bit、48bit 等字长。

假如要将 30 亿个数求和,该如何办呢?往往是先将每个数对应到一个字,将这 30 亿个数存放在内存里相邻的地址,形成一个**数组**(array)。执行求和计算时,先将结果"和"置零,然后再将数组里的这 30 亿个字与"和"逐一相加,最终得到结果。

(5) 很多时候,多个数据不能放置在相邻的地址,这时需要**指针**(pointer):一个数据对应的地址之后,存放的不是下一个数据,而是下一个数据的地址。

数组与指针示例如图 5.2 所示。

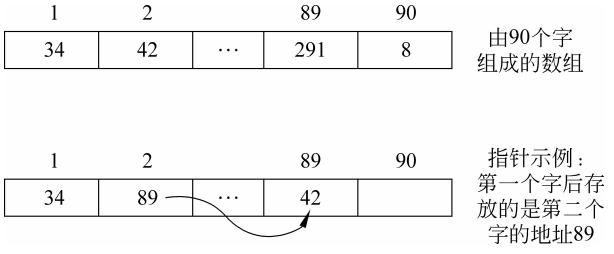

图 5.2 数组与指针示例

(6) 文件(file)是一个常用的更大粒度的数据结构。它可能采用了上述所有的数据类型。例如,一本书可以是一个以 pdf 格式存储的文件。它包括中文字符(每个中文字符需要两个字节表示),还包括了需要的各种格式排版与图形图像和公式信息。这也是为什么一本书的 pdf 格式文件比 txt 格式文件大得多的原因。

(7) 数据(data)是一个范围较广的术语,是指一个或多个数字符号。一个比特、4 比特、字节、字、指针、文件都是数据。数据不只是数值,还有对应的含义,称为数据的**语义**。

实例:字符数据的语义。计算机如何表示人类语言的各种字符呢?表示英文字符比较简单。英文只有 26 个字母,区分大小写也只有 52 个字母。加上 10 个十进制数字字符、各种标点符号等,一个字节足以表示全部英文字符。1967 年,美国国家标准学会正式发布了美国信息交换标准代码(简称 ASCII 码),用一个字节的 7 位表示英文字符,剩余的 1 位用于纠错。7 位字长有 128 个组合,可表示 128 个符号。其中,32 个组合用于表示回车、换行等控制符,另外 96 个组合用于表示 52 个字母(A～Z,a～z)、10 个数字(0～9)、

各种标点符号(！ @ # $ %，. 等)。(用户可通过键盘输入“Alt＋小键盘数字键”看到对应的 ASCII 码对应的字符显示。例如，输入 Alt＋64 将看到@，输入 Alt＋65 将看到 A。)

中文字符表示也可采用类似的方法。但是，康熙字典中就有数万个中文字符，单字节表示显然不够。其他非拉丁语言也可能有同样的问题。一个国际性的非营利组织 Unicode Consortium 应运而生，推出了双字节 Unicode 码，即采用两个字节(16 比特)表示语言字符。后来又推出了四字节(32bit)Unicode 码。这样，一共有 2^{32} 个组合，可表示 400 万个字符，基本能够覆盖世界上所有语言。这也是为什么 Unicode 被翻译成“万国码”“国际码”“统一码”“国际标准字符集”的原因。采用双字节 Unicode 码，每个中文字符对应于一个由四个十六进制数字符号组成的数值。例如，“中”“国”两个中文字符分别对应两个 Unicode 码 4E2D 和 56FD。

小结一下，在理解一个特定的数据抽象(例如 Unicode 码)时，需要从四个角度考虑才能完整理解。

- 该数据的数字表示。由四个十六进制数字符号表示的数值，如 Unicode 码 2632。
- 该数据的存储格式。例如，2632 存放在相邻两个字节中。
- 该数据的语义。例如，2632 对应的语义是中文八卦中的离卦☲字符。
- 该数据的操作(存储操作、运算操作、通信操作)。大致上讲，从键盘输入该字符到计算机中，可以看成一种通信操作。计算机将 2632 存放在内存，再读出来进行后续处理，这些是存储操作。计算机将 2632 根据 Unicode 规则变成☲以便于显示，这是运算操作。

一个文件数据(如一本书的 pdf 文件)也可以从这四个角度理解。

- 该数据的数字表示。例如，一本书的 pdf 文件的数字表示是由接近一亿个二进制数字符号(0 或 1)组成的数值。
- 该数据的存储格式。一本书的 pdf 文件按照 pdf 格式存放在连续的大约 1000 万个字节中。
- 该数据的语义。就是打开该书 pdf 文件后看见的内容。
- 该数据的操作(存储操作、运算操作、通信操作)。打开该书 pdf 文件，意味着从硬盘将文件“读”到内存和处理器中，是一种存储操作。将文件加密或压缩，是运算操作。将该文件上传到某个网站，是通信操作。

上述字符数据例子中，“为世界上的所有语言的所有字符规定二进制数字符号”是抽象化的过程，而 Unicode 码是抽象化的产物。Unicode 码已经体现了抽象三性质。

- 有限性。Unicode 这个抽象解决了如何用数字符号表示世界上各种语言字符的问题，但忽略了字体(是宋体、隶书还是黑体)、大小(是小五还是四号字体)、如何对齐、如何具体显示打印等问题。
- 精确性：Unicode 这个抽象是一个语义精确、格式规范的计算概念。它没有歧义。它的格式也是规范的，每个字符存放在相邻两个字节中。
- 通用性：Unicode 这个抽象并不是针对某个计算机、某个软件或某个应用场景设计的。它用一个通用抽象支持多个具体应用需求。不论何时何地，不管是什么计算机、使用什么操作系统和应用软件、处于何种应用场景，Unicode 用统一的一套

方法解决了“用数字符号表示世界上各种语言字符”这个特有问题。

2. 控制抽象

下面简要讨论三种常见的控制抽象，包括**顺序**、**条件跳转**、**调用**。顺序(sequential order，或 sequence)是最基本、最常用的控制抽象，在很多情况下是默认的控制抽象。例如，如果没有特别说明，在描述一个计算过程时，列出的步骤都是一步一步顺序执行的。

步骤 0：开始计算过程。

步骤 1：Sum＝0。

步骤 2：i＝0。

步骤 3：Sum＝A[i]＋B[i]＋Sum。

步骤 4：i＝i＋1。

步骤 5：如果 i＜30 亿，跳转到步骤 3。

步骤 6：打印出结果 Sum。

步骤 7：终止计算过程。

上述计算过程中，步骤 5 通过条件跳转抽象来确定下一步骤是步骤 6 还是步骤 3。除此之外，所有步骤都采用默认的顺序下一步骤。

步骤 6 值得特别关注。在一般的计算机中，打印并不是一条基本的指令，而是通过一段子程序来实现的。步骤 6 事实上应该是一个子程序调用，是一种特殊的控制抽象。

这些抽象可被组合，产生更多的抽象。例如，顺序、条件跳转、调用可组合成循环抽象。

步骤 0：开始计算过程。

步骤 1：Sum＝0。

步骤 2：for(i＝0；i＜30 亿；i＋＋)Sum＝A[i]＋B[i]＋Sum。

步骤 3：打印出结果 Sum。

步骤 4：终止计算过程。

在这个过程中，步骤 2 变成了一个更加抽象的循环语句。它的含义是调用下列子程序。

步骤 A：i＝0。

步骤 B：Sum＝A[i]＋B[i]＋Sum。

步骤 C：i＝i＋1。

步骤 D：如果 i＜30 亿，跳转到步骤 B。

显然，该 for 循环语句抽象比相应的子程序更加简洁易懂。

3. 硬件抽象

抽象化涉及的数据抽象和控制抽象不只局限于看起来偏“软”的数据格式与算法步骤规定，它们也可以体现为硬件抽象。

实例：运算器抽象化。在计算机发展历史中，一个关键的抽象化工作是确定运算器

抽象。具体的问题是：一个处理器中的运算器应该如何设计，才能产生一个通用处理器？早期的计算机采用了比较随意的(ad hoc)方式，而非系统的(systematic)方式，因此，不能回答这个问题。例如，ENIAC 的运算器包括了计数器、乘法器、除法器、平方根器等硬件单元，能够支持一些数值计算(科学计算)。当代计算机则需要支持科学计算、企业工作流、事务处理、文字处理、数据分析、游戏、上网、社交网络等多种应用。1964 年的 IBM 360 系统地回答了这个问题，提出了**定点部件**和**浮点部件**硬件抽象，使得 IBM 360 成为了通用计算机。通俗地讲，定点部件处理整数的加、减、乘、除、与、或、非运算，而浮点部件处理有限精度的实数加、减、乘、除运算。

用定点部件和浮点部件硬件抽象支持所有的应用所需的运算，而不是为科学计算设计几种运算器，又为企业计算设计几种运算器，再为文字处理设计几种运算器，这就是“用一个通用抽象支持多个具体应用需求”的体现。x86、ARM、龙芯、RISC-V 处理器指令系统体系结构等，都继承了这种硬件抽象。

抽象化需要人们付出辛勤努力和聪明才智。即使浮点部件这样粗看起来较小的硬件，也是汗水和灵感浇灌而成的。加拿大计算机科学家威廉·卡汗(William Kahan)从 20 世纪 70 年代开始长期致力于计算机浮点部件硬件抽象的研究，并促成了 IEEE 754 浮点算术标准的建立。人们今天使用的计算机，包括超级计算机、微机、平板电脑、智能手机，都用到了他的研究成果。卡汗教授也因此在 1989 年获得了图灵奖。

小结一下计算抽象的要点。

计算过程和计算思维理解 8：抽象化

少数精心构造的计算抽象可产生万千应用系统。

计算过程刻画：

(1) 一个计算过程是解决某个问题的有限个计算步骤的执行序列。

(2) 每个计算步骤都是对数据抽象的操作，并受控制抽象约束。

(3) 计算步骤有大有小。大的计算步骤由小的计算步骤组合而成，是一组小的计算步骤的组合抽象。

计算思维要点：计算系统思维方法的要点是**通过抽象将部件组合成为系统，执行计算过程**。

计算抽象既是计算机科学最重要的方法，也是最重要的产物。与其他学科的抽象不同，计算机科学的抽象强调可自动执行的、比特精准的信息变换抽象。

抽象化和抽象具备**抽象三性质**。

- 有限性：每个抽象仅仅考虑一个层次的有限的特有问题，忽略其他层次和其他问题。
- 精确性：每个抽象是一个语义精确、格式规范的计算概念。
- 普遍性：每个抽象都具有泛化能力。

5.2.2 模块化

有一类特殊的抽象化方法在设计和理解计算系统时得到广泛应用，这就是模块化方法。模块化方法的要点是理解如何从部件（即**模块**）组合系统。有时候，模块也被称为**子系统**。模块化方法需要回答三个问题，它们合起来称为**系统架构三问题**。

- 系统是由哪些模块组成的？也可以反过来问：一个系统如何分解成多个模块？
- 系统是由这些模块如何组成的（模块之间如何连接、有什么接口）？
- 计算过程在系统中如何执行？

系统由多个模块组合而成，这句话表面上看来是显而易见的。例如，一个计算机系统可以由下列模块（子系统）组合构成：

计算机 = 硬件 + 系统软件 + 应用软件
计算机硬件 = 处理器 + 存储器 + 输入 / 输出(I/O) 设备
处理器 = 运算器 + 控制器 + 寄存器 + 数据通路 + …
……

最终到达计算机的基本操作，例如逻辑门电路。

事实上，“系统由多个模块组合而成”大有讲究，是计算机科学领域的重要创新环节。人们将“系统由多个模块组合而成”的成功的特定方法称为**系统架构模型**，并赋予特定的名称，例如，布尔函数的组合电路模型、时序电路的自动机模型、计算机硬件系统的存储程序计算机模型（也称冯·诺依曼模型）、数据管理系统的关系数据库模型，等等。

下面通过几个从小到大逐级的抽象，看看如何使用模块化方法构成系统。每个抽象本身也是一个系统，但在更大的系统中扮演了子系统（模块）的角色。这些抽象是逻辑门、组合电路、时序电路、自动机、存储程序计算机、应用程序执行模型。组合电路和时序电路合起来称为**数字电路**。我们还阐述模块化的一个核心原理，即信息隐藏原理。

1. 逻辑门

从逻辑设计角度看，计算机系统的最基本的抽象是布尔逻辑门电路，简称**门**（gate）。图 5.3 和图 5.4 列出了五种常见门的符号与真值表，即与门、或门、非门、异或门、与非门。

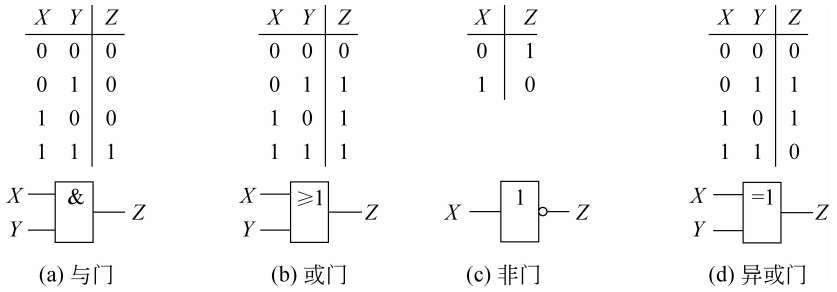

图 5.3 四种逻辑门电路及其真值表

这些逻辑门之所以是计算抽象，是因为它们集中考虑计算逻辑设计层次，忽略了具