华章教育
HZ BOOKS

全美经典
学习指导系列

计算机导论
习题与解答

INTRODUCTION TO COMPUTER SCIENCE

最佳的复习资料，实用的辅助教材

与国外高校计算机水平保持同步

为考研和出国深造奠定坚实基础

Ramon A. Mata-Toledo　Pauline K. Cushman 著
薛静锋 邹辉 等译

全球销售超过3000万册！

Mc Graw Hill
机械工业出版社 China Machine Press
中信出版社 CITIC PUBLISHING HOUSE

全美经典
学习指导系列

计算机导论习题与解答

INTRODUCTION TO COMPUTER SCIENCE

Ramon A. Mata-Toledo　Pauline K. Cushman 著
薛静锋 邹辉 等译

机械工业出版社
China Machine Press
中信出版社
CITIC PUBLISHING HOUSE

本书覆盖了基本的程序设计概念，给学生提供了相关的计算机科学的定义和规则。全书内容深入浅出，大量的例题和习题会可以增强学生的实际能力，并加强自己的理论知识，同时也重复讲解了基本规则，以便使学生能更好地掌握其所学知识。

本书版权登记号：图字：01-2002-0356

图书在版编目（CIP）数据

计算机导论习题与解答/（美）托勒多（Toledo, R. A.）等著；薛静锋等译. -北京：机械工业出版社，2002.8

（全美经典学习指导系列）

书名原文：Introduction to Computer Science

ISBN 7-111-10849-3

Ⅰ.计… Ⅱ.①托… ②R… ③薛… Ⅲ.计算机科学-基本知识-解题 Ⅳ.TP3-44

中国版本图书馆CIP数据核字（2002）第063175号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码100037）

责任编辑：华章

北京忠信诚印刷厂印刷·新华书店北京发行所发行

2002年8月第1版第1次印刷

787mm × 1092mm 1/16 · 19.5印张

印数：0 001-5 000册

定价：29.00元

RAMON A. MATA - TOLEDO 博士是 James Madison 大学的计算机科学副教授。他获得了堪萨斯州立大学的计算机科学博士学位，并获得了佛罗里达理工学院的 MS 和 MBA 学位，他的学士学位是在加拉加斯(委内瑞拉)的 Instituto Pedagogico 获得的，在那里他主攻两个专业：数学和物理学。MATA-TOLEDO 博士感兴趣的主要研究领域是数据库、自然语言处理和应用数学。他是各种专业杂志、国内外学术会议的大量论文的作者，也是《科学技术应用数学导论》的合著者。可以通过 matalra@jmu.edu 和 MATA - TOLEDO 博士联系。

PAULINE K. CUSHMAN 博士是 James Madison 大学的综合科学技术和计算机科学副教授。她获得了路易斯维尔大学计算机科学和工程博士学位，并获得了西佛吉尼亚大学教育学 MA 学位和西佛吉尼亚大学研究生院信息系统学 MS 学位。她的学士学位是在 Davis and Elkins 学院获得的，在那里她主攻基础教育学。CUSHMAN 博士感兴趣的主要研究领域是智能系统、多媒体和 Web 设计。可以通过 cushmapk@jmu.edu 和 CUSHMAN 博士联系。

作 者 序

本书面向的对象是想对计算机科学概念有综合认识并清楚理解这些概念在不同语言中的具体应用的读者。本书的目的是在不检查任何特定语言复杂性的基础上给出一些资料。书中的例子使用了 Visual Basic、C、C ++ 和 Java 结构。尽管这些语言在实现上有差异，但我们希望通过使用这些语言，使读者可以获得关于计算机科学基本概念的统一认识。

本书分为九章，覆盖了基本的程序设计概念，给学生提供了相关的应用于计算机科学的定义和规则。例题和习题使学生们可以把所学知识应用到实际中并加强自己的理论知识，同时也重复讲解了基本规则，以便使学生们更好地掌握。我们鼓励同学们在适当的环境中试着完成例题并编写程序。

我们由衷地感谢为 Schaum 系列丛书做过贡献的全体人员以及我们的朋友和同事，是他们的鼓励使我们能够完成本书。如果读者喜爱本书并发现本书对增加自己的计算机专业知识有所帮助，我们将会感到非常欣慰。

目　录

第 1 章　计算机基本概念

计算机是一个设备，在程序的指导下，它能够处理数据，改变其自身的程序指令，执行计算和逻辑运算，而这些都不需要人工干预。术语**程序**是指能够让计算机完成特定任务的特定的指令集合。**程序员**是指为计算机编写指令的一个人或一组人员。一般来说，程序就是指“**软件**”。

可以从两个不同的层次来考虑计算机：体系结构和物理构造。体系结构由用户可见的接口组成，就像程序员所看到的一样。也就是说，是从程序员的视角所看到的计算机的结构和操作。计算机的物理构造是由特定硬件（也可能是软件组件）的接口组成的，在本书中指的是计算机的各组成部分，例如：显示器、打印机、键盘以及其他一些称为“**硬件**”的电子器件。

1.1　计算机的结构

大部分计算机系统通常都由三个基本的结构或子系统组成：高速存储单元、中央处理单元和包含 I/O 子系统的外围设备（如图 1.1 所示）。

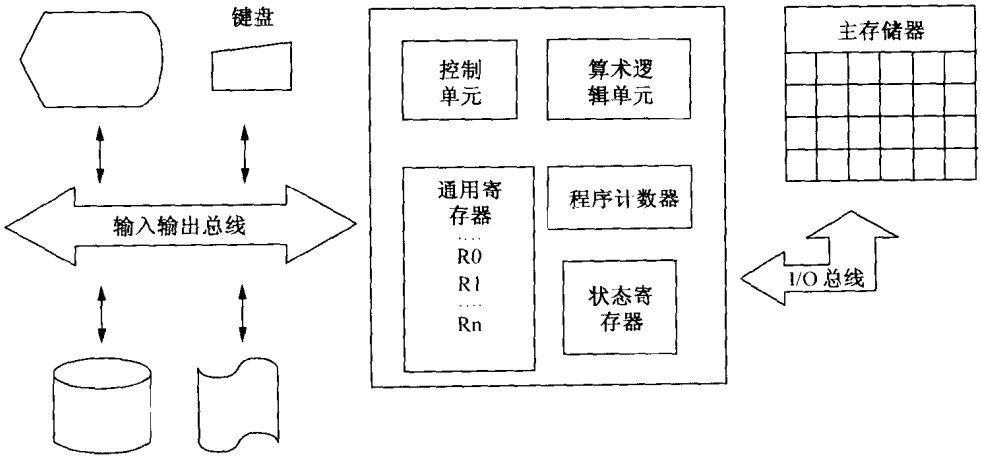

图 1.1　计算机基本结构

1.1.1　存储单元

计算机的存储单元也叫**主存储器**或**物理存储器**，存储着中央处理单元能够直接存取和执行的所有指令和数据。大多数计算机存储器都是由芯片组成的，这些芯片是由做在硅片上的

金属氧化物组成的。这种类型的存储器也叫**随机存取存储器**或 RAM。

计算机的存储器通常被分为大小相同的逻辑单元。最普通的单元叫做字节，每个字节是由 8 个连续的**位**或**二进制数字**组成的（如图 1.2 所示）。每个单独的位能够被磁化为两个不同的状态之一，因此称为二进制。一个状态表示 1；另一个状态表示 0，这两个状态分别指的是“开”和“关”。

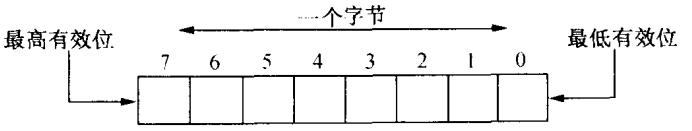

图 1.2 字节的表示

每个字节都和一个惟一的地址相关联。根据使用的约定，地址可以从右向左或从左向右增加（如图 1.3 所示）。在本书中如果没有特别指明，都假定地址是从右向左增加的。**地址空间**是一个程序能够引用的所有惟一地址的集合。用来表示一个地址的位数决定了地址空间的大小。这个空间大小是 2^N，其中 N 表示地址的位数。弄明白存储单元的地址和内容之间的不同是很重要的。一个字节的地址是确定的，但其内容可以是变化的。

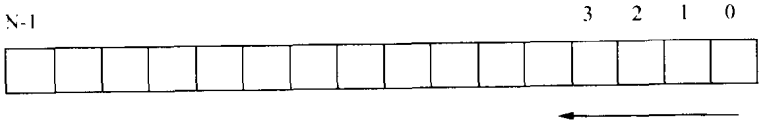

图 1.3 地址增加方向

字节也可以被组合成更大的单元。根据制造商所使用的约定，这些更大的单元可能被叫做不同的名字。表 1.1 列出了一些较小单元和更大单元的常用名字。

表 1.1 计算机中的字节表示

1 nibble（半字节）	4 个连续位
1 byte（字节）	8 个连续位
1 word（字）	2 个连续字节
1 longword	4 个连续字节
1 quadword	8 个连续字节
1 octaword	16 个连续字节

正如图 1.2 所示，一个字节的位通常被从右向左编以以 0 开始的号码。最右边的位叫**最低有效位**（lsb），同样地，最左边的位叫**最高有效位**（msb）。

由于一个字节的每一位只能存储两个值中的一个，0 或者 1，因此一个字节有 2^8 种不同的组合。每种组合表示一个惟一的值。一个字节中位的每种组合所代表的值依赖于解释这些值所使用的约定（参见例 1.5）。假定位是无符号的，那么一个字节的位所表示的数值可以通过以下方法来计算：

(1) 从最右边位的位置开始使用上标给位编号，最右边的位的上标是 0，在其左边的下一个位的上标是 1，左边再下一个位的上标是 2，依此类推。

(2) 把每一个上标作为 2 的幂指数。

(3) 以相应的 2 的幂乘以每一位的值。

(4) 将上一步得到的乘积累加起来。

例 1.1　计算无符号二进制数11001101对应的十进制数值是多少?

(1) 从最右边位的位置开始使用上标给位编号，如下所示。

$$1^7 1^6 0^5 0^4 1^3 1^2 0^1 1^0$$

(2) 把每一个上标作为 2 的幂指数。

$$(1 * 2^7) + (1 * 2^6) + (0 * 2^5) + (0 * 2^4) + (1 * 2^3) + (1 * 2^2) + (0 * 2^1) + (1 * 2^0)$$

(3) 以相应的幂乘以每一位的值并将结果相加。

$$(1 * 2^7) + (1 * 2^6) + (0 * 2^5) + (0 * 2^4) + (1 * 2^3) + (1 * 2^2) + (0 * 2^1) + (1 * 2^0) = 205$$

因此，无符号二进制数 11001101 对应的十进制数值是 205。注意根据定义，2^0 等于 1。计算无符号二进制数的值的另外一种方法在习题 1.19 中将会说明。

正如前面所指出的，一个计算机的存储器的大小是用字节来度量的。然而，存储器的大小通常用更大的单位来表示。一个最常用的单位是千字节或者叫 Kbyte，也可简称为 K。在计算机术语中，1Kbyte 等于 1024 字节。有时需要一个粗略的近似值，这时 1Kbyte 可以被认为是等于 1000 字节。表 1.2 列出了当前用来度量主存储器的一些其他单位。在编写本书时，用来表示主存储器大小的最常用的单位是兆字节或者叫 MB（$\approx 10^6$ 字节）；然而，在不久的将来可能会更频繁地使用一些更大的单位。符号≈应该读作“约等于”。

表 1.2　存储器的存储单位

1 Kilobyte（千字节）	= 1024 字节
1 Megabyte（兆字节）	$\approx 10^6$ 字节
1 Gigabyte（吉字节）	$\approx 10^9$ 字节
1 Terabyte	$\approx 10^{12}$ 字节
1 Petabyte	$\approx 10^{15}$ 字节
1 Exabyte	$\approx 10^{18}$ 字节

例 1.2　广告中称一台计算机主存储器的容量为32兆字节或叫“Megs”，计算用字节表示的该存储器的大小是多少?

$$32\ \text{Mbytes} = 32 * 10^3\ \text{Kbytes} = 32 * 10^3 * 1024\ \text{字节} = 32\,768\,000\ \text{字节}$$

1.1.2　中央处理单元

中央处理单元（CPU）或者叫处理器，是“计算机的大脑”。计算机内部的大部分活动

都是在 CPU 中发生的。通常可以把 CPU 进一步划分为两个基本的子单元：**算术逻辑单元**（ALU）和**控制单元**（CU）。

CPU 的主要任务是从存储器中获取指令并执行这些指令。从存储器中获取的指令要被解码，为一条指令进行解码意味着要解释这条指令是关于哪方面的以及它的操作数是什么，而执行的意思是去做这条指令所意味着的动作。

正如其名字所隐含的含义，在 CPU 的 ALU 或者叫算术逻辑单元中执行的是算术运算（加法、减法等）。类似地，其他运算，例如两个数的比较，也是在 ALU 中进行的。在 CPU 的内部有一些“通用”寄存器可以为处理器提供临时的、高速的存储。设想一个典型的情况，要对位于主存储器中的两个数进行相加运算。这个操作可以按以下方式执行：首先，位于主存储器中的数被送到 ALU 的内部寄存器，在这里完成加法运算。如果需要，运算结果可以存储到存储器的特定位置。

除了这些高速的内部寄存器，CPU 还包含着一个或多个“状态寄存器”，其提供的信息包括处理器的状态、被处理的指令、可能已经出现的任何特殊情况以及用来处理这些特殊情况所需要采取的行为。

CPU 的控制单元管理着处理器中的数据流向。例如，对于前面例子中两个数的相加，必须把操作数从存储器移动到 ALU。管理数据的流向是控制单元的职责。如果相加的结果要被存储到存储器，那么这也是控制单元的任务。控制单元内部组合了解码器，它决定了计算机需要执行的操作。

1.1.3 输入输出单元

计算机通过输入设备接收信息，用户使用输入设备向计算机发送信息，最常用的两个输入设备是键盘和鼠标。输出设备用来向用户发送信息，最常用的输出设备是显示器和打印机。其他设备，例如一些外部存储单元（硬盘、磁带、jazz 或 zip 驱动器），可能既有输入功能又有输出功能。输入和输出是 CPU 执行的最复杂的操作。关于 I/O 设备物理特性和所需数据格式的细节是由系统程序处理的，对用户来说不可见。涉及 I/O 处理过程的讨论超出了本书的范围。

1.2 总线结构

为了在不同组件之间传输数据，计算机使用了称为**总线**的导线集合。一个典型的计算机系统有三种总线：地址总线、数据总线和控制总线。数据总线用于数据的传输。数据总线中导线的根数应该不少于指定计算机中存储单元的位数。为了存取存储器中的数据，用地址来表示数据的位置很有必要。CPU 通过地址总线向存储器发送地址位。

图 1.4 显示了一个双总线结构的计算机。CPU 和存储器通过存储器总线进行交互。输入和输出功能通过 I/O 总线实现，另外也使用一些其他的结构。图 1.5 显示了用于一些小型

计算机的单总线结构。

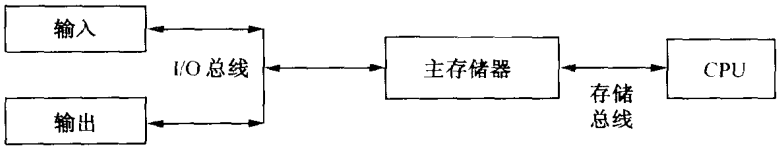

图 1.4　双总线结构

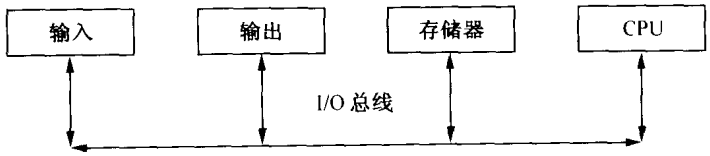

图 1.5　单总线结构

例 1.3　假定一台计算机的基本单元是字（word），在任何一个时刻通过数据总线能够传输多少位？

答案取决于一个字的位数。根据表 1.1，需要传输的位数至少是 16 位，之所以说至少，是因为一些附加的控制信息也可能需要在数据总线中传输。

1.3　计算机的基本操作

正如前面所指出，计算机的操作是由程序的指令控制的。在执行指令之前，CPU 直接从存储器中获取一条指令；然而，在执行指令之前，计算机如何知道每条指令意味着要做什么呢？这个问题的答案很简单。计算机根据预先定义的格式对每条指令进行解码。一条指令通常的格式如下所示：

operator [operand1], [operand2], [operand3]

在这里 operator 表示被执行动作的类型（ADD、SUBTRACT、MULTIPLY 等等），operand 是信息“块”，operator 所表示的操作将针对该信息块来执行。环绕 operand 的方括号表示操作数是可选的。根据这个通用的格式，一条指令可以有 0、1、2 或 3 个操作数。读者应该意识到计算机只能看懂 0 和 1 的序列，它不能看懂像“ADD”、“SUBTRACT”这样的运算符；但是它能够看懂预先定义的等价的数字代码，例如用 10000001 表示 ADD（加法）或者用 10000110 表示 MULTIPLY（乘法）。

一个典型的指令格式如下所示：

ADDW3 location1, location2, result

这个特定的指令把位于地址 location1 和 location2 的两个字的内容相加，并把和存储在称为 result 的地址单元中。“W”表示操作数为字类型，后缀“3”表示这条指令有三个操作数。

完成这条指令需要好几个步骤。首先，把指令从存储器传送到 CPU。当前将被执行的指令的地址存放在 CPU 的一个称为**指令计数器**（IC）或**程序计数器**（PC）的专用寄存器中

——术语随制造商的不同也有所不同。刚刚获取的指令被存储在另一个称为**指令寄存器**（IR）的专用寄存器中。控制单元对指令进行解码并识别，从而得知是一个带有三个字类型操作数的 ADD 操作。然后，取出第一个操作数（location1）和第二个操作数（location2），并把它们放在两个通用寄存器中。这两个寄存器中的值进行相加，得到的结果被存储到存储器中称为 result 的位置。在获取了两个操作数之后，程序计数器被更新，指向将要执行的下一条指令。在这个过程中，有两个附加的寄存器帮助完成 CPU 和主存储器之间的通信。这两个寄存器是**主存地址寄存器**（MAR）和**主存数据寄存器**（MDR）。MAR 用来保存数据被传输到的位置的地址或者数据来源位置的地址。MDR 保存要被写入地址单元或者从地址单元读出的数据。

例 1.4 假定一台计算机有两条用于加法运算的指令，其格式为：ADDW2 location1，location2 和 ADDW3 location1，location2，result。比较使用第一条和第二条指令的不同之处。

要准确地回答这个问题，需要理解这两条指令在格式上的不同。第一条 ADD 指令包含两个字类型的操作数。因此命名为 ADDW2。我们假定根据计算机制造商的约定，在这种类型的指令中运算的结果被存储在第二个操作数所在的存储单元。这种类型的加法运算指令称为破坏性加法指令，因为 location2 中的值将被加法运算的结果所替换（如图 1.6 所示）。

第二条指令包含三个字类型的操作数。因此命名为 ADDW3。在这种类型的 ADD 指令中，前两个操作数相加的结果被存储在第三个操作数所指定的位置处。这种类型的指令称为非破坏性加法指令，因为在指令执行完成后操作数的值仍被完整地保留着（如图 1.7 所示）。

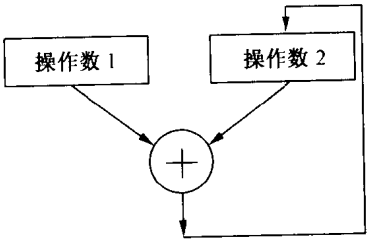

图 1.6 破坏性加法

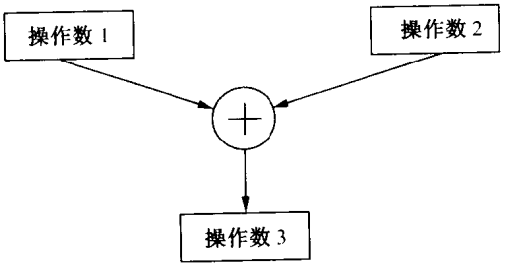

图 1.7 非破坏性加法

1.4 存储器中数据的表示

如果我们能够看一看计算机存储器中的内容，就会发现所有类型的数据和指令都是仅仅用 0 和 1 来表示的。因此，一个很显然的问题就是计算机如何能够得知存储在特定位置的信息的类型？答案很简单：计算机是从数据被使用的上下文环境中得知存储在特定位置的数据的类型的，这就允许计算机正确地去解释它。例 1.5 显示了一个字符串数据可以有不同的解释。

例 1.5　下面给定4个字节的序列，如果分别用这样的格式进行考虑（a）四个单独的字节；（b）两个字类型的整数，每个字有两个字节；（c）一个双字类型的整数，每个双字有四个字节，那么这个序列所表示的值是什么？

01100011	01100101	01000100	01000000

←——地址增加方向

每个单独字节的值可以使用第 1.1.1 节中指出的方法来计算。数字右下角带圆括号的下标表示数字是二进制（2 或十进制（10。正如在后面将要看到的，这个数字叫做数字的基数或根。

（a）从右向左考虑各个字节，得到：

$$01000000_{(2} = 0^7 1^6 0^5 0^4 0^3 0^2 0^1 0^0 = (1 * 2^6) = 64_{(10}$$

$$01000100_{(2} = 0^7 1^6 0^5 0^4 0^3 1^2 0^1 0^0 = (1 * 2^6) + (1 * 2^2) = 64 + 4 = 68_{(10}$$

$$01100101_{(2} = 0^7 1^6 1^5 0^4 0^3 1^2 0^1 1^0 = (1 * 2^6) + (1 * 2^5) + (1 * 2^2) + (1 * 2^0) = 64 + 32 + 4 + 1 = 101_{(10}$$

$$01100011_{(2} = 0^7 1^6 1^5 0^4 0^3 0^2 1^1 1^0 = (1 * 2^6) + (1 * 2^5) + (1 * 2^1) + (1 * 2^0) = 64 + 32 + 2 + 1 = 99_{(10}$$

（b）第一个双字节字由右边的两个字节组成；使用第 1.1.1 节中的方法，得到：

$$0100010001000000_{(2} = 0^{15} 1^{14} 0^{13} 0^{12} 0^{11} 1^{10} 0^9 0^8 0^7 1^6 0^5 0^4 0^3 0^2 0^1 0^0$$

$$= (1 * 2^{14}) + (1 * 2^{10}) + (1 * 2^6)$$

$$= 17\,472_{(10}$$

下一个双字节的字是 0110001101100101，使用第 1.1.1 节中的方法，得到：

$$0110001101100101_{(2} = 0^{15} 1^{14} 1^{13} 0^{12} 0^{11} 0^{10} 1^9 1^8 0^7 1^6 1^5 0^4 0^3 1^2 0^1 1^0$$

$$= (1 * 2^{14}) + (1 * 2^{13}) + (1 * 2^9) + (1 * 2^8) + (1 * 2^6) + (1 * 2^5) + (1 * 2^2) + (1 * 2^0)$$

$$= 24\,445_{(10}$$

（c）使用第 1.1.1 节中的方法，计算包含四个字节的双字的值，得到：

$$0^{31} 1^{30} 1^{29} 0^{28} 0^{27} 0^{26} 1^{25} 1^{24} 0^{23} 1^{22} 1^{21} 0^{20} 0^{19} 1^{18} 0^{17} 1^{16} 0^{15} 1^{14} 0^{13} 0^{12} 0^{11} 1^{10} 0^9 0^8 0^7 1^6 0^5 0^4 0^3 0^2 0^1 0^0{}_{(2}$$

$$= (1 * 2^{30}) + (1 * 2^{29}) + (1 * 2^{25}) + (1 * 2^{24}) + (1 * 2^{22}) + (1 * 2^{21}) + (1 * 2^{18}) + (1 * 2^{16})$$

$$+ (1 * 2^{14}) + (1 * 2^{10}) + (1 * 2^6) = 1\,667\,580\,992_{(10}$$

1.4.1　数制系统和代码

最常用的数制系统是十进制系统。但是，由于电子设备的二进制特性，在计算机中使用十进制系统受到了很大的限制。大部分的数据表示和算术运算是在二进制系统中完成的，或者是在一些“速记（shorthand）”表示法例如八进制或十六进制系统中完成的。然而，在考虑二进制、八进制或十六进制系统之前，我们先回顾一下十进制系统的一些特性。

正如其名字所表明的一样，十进制系统的基数是数字 10。数字 10 称为系统的**基数**（basis）或**根**（radix）。在任何数制系统中，基数表明在系统中有多少个不同的符号。在十进制系统中的这些符号叫做数字（digit），就是常见的 0，1，2，3，4，5，6，7，8 和 9。注意它

们的数值范围是从 0 到 9。一个比较常用的、精确的规则被描述如下：

“给定任何一个基数或根为 N 的正整数，在系统中有 N 个各不相同的符号可以用来书写这个数字。这些符号的值的范围是从 0 到 N - 1。”

除了二进制系统之外，计算机和通信领域也广泛地使用一些其他的数制系统，包括八进制（其基数是 8）和十六进制（其基数是 16）系统。例 1.6 说明了这些系统的基本组成。

例 1.6 八进制和十六进制系统中分别有多少个不同的符号？这些符号的范围是什么？

八进制系统的基数是 8，即 N = 8，所以有 8 个不同的符号。这些符号的范围是从 0 到 (8 - 1)，即从 0 到 7，分别是：0，1，2，3，4，5，6 和 7。我们用十进制的名字来称呼这些符号：一、二、三等等。

十六进制系统的基数是 16，即 N = 16，所以有 16 个不同的符号。这些符号的范围是从 0 到 (16 - 1)，即从 0 到 15，分别是：0，1，2，3，4，5，6，7，8，9，A，B，C，D，E 和 F。注意在 9 之后，符号是从 A 到 F 的字母。在该系统中，字母 A 表示 10，B 表示 11，C 表示 12，D 表示 13，E 表示 14，F 表示 15。选择字母代替数字的组合来表示大于 9 的符号的原因是为了使所有的符号都是单个字符。

为了表示一个给定数字的基数，可以在数字的右下角使用一个下标。正如在第 1.4 节中指出的符号 $0101_{(2}$表示该数字是二进制数字。同样地，符号 $01256_{(8}$表示该数字的基数是 8。对于十进制系统的数字，下标通常省略不写。然而，用文字进行明确说明还是很有必要的。尽管数字这个词通常指的是十进制系统中的单个符号，但是其他数制系统中的单个符号也经常用这个名称。

1.4.2 位置系统

迄今为止我们考虑过的所有数制系统，包括十进制系统，都是**位置系统**。也就是说，在一个数字表达式中，一个符号所代表的值取决于其在数字中的位置。例如，在十进制系统中，数字 478 中的 4 代表 400；数字 547 中的 4 代表 40。如果我们对数字进行分解，就可以更清楚地看到这一点，如下所示：

$$\mathbf{478} = \mathbf{400} + 70 + 8$$

$$547 = 500 + \mathbf{40} + 7$$

在任一位置系统中，数字表达式中任一数字所代表的值都可以通过以下步骤进行计算：(1) 使用上标从右向左为数字编号，从 0 开始，最右边数字的上标为 0，然后按从右向左移动的方向依次将上标递增 1；(2) 将每个上标作为基数的幂指数；(3) 按十进制运算法，用数字本身的值乘以相应基数的幂。

要计算与整个数字等价的十进制数，将步骤 3 中得到的所有乘积累加起来。

例 1.7 在数字$1228_{(10}$中每个数字的值是多少？

因为是十进制数字，所以基数是 10。按照下面的步骤计算每个数字的值。

(1) 使用上标为数字编号。从 0 开始，最右边位置的数字编号为 0，从右向左移动，上标依次加 1。这样处理后该数字如下所示：

$$1^3 2^2 2^1 8^0$$

(2 - 3) 将上标作为基数的幂指数，按十进制运算法用数字本身的值乘以相应基数的幂：

在**1**228中 1 的值等于 $1 * 10^3 = 1 * 1000 = 1000$。

在1**2**28中第一个 2（从左到右）的值等于 $2 * 10^2 = 2 * 100 = 200$。

在12**2**8中第二个 2（从左到右）的值等于 $2 * 10^1 = 2 * 10 = 20$。

在122**8**中8的值等于 $8 * 10^0 = 8 * 1 = 8$。

例 1.8　在数字$1253_{(8}$中每个数字的值是多少？与其等价的十进制数是多少？

因为该数字是八进制系统中的数字，所以基数是 8。使用与上例中类似的方法，给数字写上上标，得到 $1^3 2^2 5^1 3^0$。将这些上标作为基数的幂指数，得到：

在 1253 中 1 的值等于 $1 * 8^3 = 1 * 512 = 512$。

在 1253 中 2 的值等于 $2 * 8^2 = 2 * 64 = 128$。

在 1253 中 5 的值等于 $5 * 8^1 = 5 * 8 = 40$。

在 1228 中 3 的值等于 $3 * 8^0 = 3 * 1 = 3$。

通过累加以上所有的乘积，可以得到与该数字等价的十进制数。

$$\begin{aligned} 1253_{(8} &= (1 * 8^3) + (2 * 8^2) + (5 * 8^1) + (3 * 8^0) \\ &= (1 * 512) + (2 * 64) + (5 * 8) + (3 * 1) \\ &= 512 + 128 + 40 + 3 \\ &= 683 \end{aligned}$$

注意，无论什么时候只要我们得到了等价的十进制数，也就确定了每个单个数字的值。从现在开始我们将使用这种组合方法。

例 1.9　在数字$1A5F_{(16}$中每个数字的值是多少？与其等价的十进制数是多少？

因为该数字是十六进制系统中的数字，所以基数是 16。给数字写上上标，得到 $1^3 A^2 5^1 F^0$。要计算与该十六进制数等价的十进制数，分别用等价的十进制数 10 和 15 代替符号 A 和 F。通过将以上求得的所有幂累加起来，可以得到与该数字等价的十进制数。

$$\begin{aligned} 1A5F_{(16} &= (1 * 16^3) + (10 * 16^2) + (5 * 16^1) + (15 * 16^0) \\ &= (1 * 4096) + (10 * 256) + (5 * 16) + (15 * 1) \\ &= 4096 + 2560 + 80 + 15 \\ &= 6751 \end{aligned}$$

在 1A5F 中 1 的值等于 $1 * 16^3 = 1 * 4096 = 4096$。

在 1A5F 中 A 的值等于 $10 * 16^2 = 10 * 256 = 2560$。

在 1A5F 中 5 的值等于 $5 * 16^1 = 5 * 16 = 80$。

在 1A5F 中 F 的值等于 $15 * 16^0 = 15 * 1 = 15$。