8086/8088

汇编语言程序设计

Huibian Yuyan Chengxu Sheji

主编　唐宁九

四川大学出版社

8086/8088 汇编语言程序设计

Huibian Yuyan Chengxu Sheji

主　　编　唐宁九
副 主 编　李征
编委会成员　唐宁九　李征　张刚　郑成明　周群彪

四川大学出版社

责任编辑:曾　鑫　廖庆扬
责任校对:朱辅华
封面设计:翼虎书装
责任印制:李　平

图书在版编目(CIP)数据

8086/8088 汇编语言程序设计 / 唐宁九等编著. —成都:四川大学出版社, 2008.1
ISBN 978-7-5614-3953-1

Ⅰ.8… Ⅱ.唐… Ⅲ.汇编语言-程序设计 Ⅳ.TP313

中国版本图书馆 CIP 数据核字 (2008) 第 011305 号

书名　8086/8088 汇编语言程序设计

主　　编　唐宁九
出　　版　四川大学出版社
地　　址　成都市一环路南一段 24 号 (610065)
发　　行　四川大学出版社
书　　号　ISBN 978-7-5614-3953-1/TP·162
印　　刷　华西医科大学印刷厂
成品尺寸　185 mm×260 mm
印　　张　12.5
字　　数　306 千字
版　　次　2008 年 1 月第 1 版
印　　次　2008 年 1 月第 1 次印刷
印　　数　0 001～2 000 册
定　　价　22.00 元

◆读者邮购本书,请与本社发行科联系。电 话:85408408/85401670/85408023　邮政编码:610065
◆本社图书如有印装质量问题,请寄回出版社调换。
◆网址:www.scupress.com.cn

前　言

随着计算机科学的发展，各类程序设计语言也得到不断的充实和更新；随着计算机应用范围的拓展，各类程序设计语言的应用领域和前景也得到相应深化和发展。汇编语言，作为最接近计算机底层的程序设计语言，在计算机理论与应用不断更新和提升时，并没有被发展的浪潮所淹没，而且其理论与应用也在不断得到升华。汇编语言作为深入理解计算机系统原理的一种原理性程序设计语言，在计算机学科内仍然有其存在的必要性。并且，因为它是衔接硬件与软件的中介，其存在的必要性将长期持续下去；在各类对速度要求较苛刻的计算机应用系统中，核心代码仍然大多采用汇编语言编写，以提高其执行效率，部分针对硬件采取特殊操作的应用程序也是采用汇编语言编写的，这些都充分说明了在实际应用中汇编语言仍然有存在的必要性。汇编语言在计算机科学的理论与应用中都有存在的必要性，同时需要强调，汇编语言存在还具有相当的重要性。汇编语言是深入理解计算机底层结构的基础理论，如果对它感兴趣，那么计算机底层世界的大门将会敞开，如果惧怕、反感它，那么它将成为初学者深入计算机底层世界永远的绊脚石。

现代计算机所使用的信号都是数字信号，都是“数字计算机”。通常情况下，数字计算机能直接识别、执行或处理的指令或数据，是0和1的二进制编码。查阅Intel、AMD或任意一款处理器手册，我们会看到在描述每一条指令时，必然会介绍该指令的二进制代码。换言之，如果期望计算机执行某条指令的功能，唯一的方法就是将该指令对应的二进制代码送入处理器让它被解释、被执行。这种处理器指令的二进制代码就是机器指令，是计算机唯一能直接解释、执行的指令，机器指令的全集即是机器语言，而汇编语言正是它的助记版本。用机器指令编写的程序是计算机唯一能够直接识别并执行的程序，而其他语言编写的程序必须转换成机器语言程序才能被执行。因此，机器语言程序又被称为目标程序；更重要的推论是，计算机的一切命令和高级语言的函数、过程等，都是由机器语言组成的。任何语言都是交流的工具；计算机编程语言，无论是C/C++，Delphi，JAVA，还是汇编语言，都是人和计算机交流的工具，即人向计算机传达意图的工具，用于描述我们期望计算机做什么、怎么做，预置应对变化的对策等。我们会看到，计算机能够解释、执行的指令数量是非常有限的，似乎很难和现代电脑的强大和精彩有什么联系。但正如无数乐曲都是由7个音符组成的一样，所有的程序都是机器指令的不同组合。

理论上，我们完全可以使用机器指令编程，早在20世纪40年代计算机问世时人们正是这样做的。然而二进制代码编写的程序尽管是计算机可直接识别和执行的，但人使用起来却非常吃力：每条0和1的代码是那么相似，几乎每读一条指令都需要查指令对照表，或者将对照表全部背诵下来。在现代程序设计中，这种担心已经没有必要了，因为人们很容易就会想到，应该用符号来代替不易识别的二进制代码，最自然的符号就是人类语言中的字或词，8086/8088汇编语言采用的是英文单词缩写。这些代替机器指令的“符号”被称为“助记符”。以助记符替代机器指令为主体的编程语言，就是汇编语言。由于是用助记符逐条替代机器指令，即指令助记符和机器指令是简单的一一对应的关系，所以汇编语言又被称为“符号化的机器语言”。因为直接使用机器指令编程是极不实用的，所以汇编语言才是具有实用价值的最古老的计算机编程语言。设计程序时，先使用助记符来编写程序，编写完成后，再使用对应的二进制代码替换助记符。显然与直接用二进制代码相比，用这种方式编程的效率和程序的可读性会提高很多。

从计算机诞生到20世纪80年代初期，由于汇编语言是面向机器的语言，其应用几乎是所向披靡的，特别是其不可比拟的时空效率，使汇编语言在编程工具中有着不可撼动的王者地位。高级语言是面向问题的语言，方便人的使用，但编译必须考虑问题的一般性导致代码相对庞大。然而随着计算机硬件平台性能的不断提升，汇编语言时空效率的优势变得越来越不重要了。随着计算机应用的飞速发展，面向不同应用的编程语言具有更高的编程效率和更好的可读性，到今天，汇编语言仿佛已经不是一种主流的编程工具了。那么，这里存在一个不可回避的问题是：为什么还要深入理解汇编语言呢?

第一，汇编语言是现代计算机原理的重要组成部分，是计算机硬件和软件的衔接点，是深入计算机底层世界的基础。学习汇编语言，能使我们对计算机组成原理、编译原理等计算机的基础理论有准确的认识；第二，对涉及系统和底层的开发和编程有极其重要和有益的帮助；第三，尽管汇编语言已经不是主流的编程工具，但在一些关键环节，汇编语言的优势还是非常重要的，例如在重要软件的核心环节，由于使用频度高，如果代码是用汇编语言编写的，会有效地提高运行效率；第四，使用汇编语言进行程序设计，会使系统对硬件平台性能的要求降低，即在有效节约成本方面也具有独到的作用。

和其他编程语言不同，汇编语言（或机器语言）和某种或某系列的CPU是密切相关的，不同CPU的汇编语言程序是不能直接换用的，这也是汇编语言和其他编程语言不能相比的重要缺点。但是，对于汇编语言的学习过程，我们只能选学某种汇编语言，在其中去了解和体验汇编语言中最普遍的东西。相对于现代处理器而言，8086/8088 CPU结构简单，指令系统功能也相对简单，但是麻雀虽小，五脏俱全。例如，8086/8088 CPU中具有存储器的分段管理模式，能够体现逻辑地址与物理地址的关系；具有标志寄存器，能够体现分支或循环结构程序设计的硬件基础；具有堆栈管理，能够体现子程序调用、返回或中断调用、返回的硬件机制基础；具有能够同时完成无符号数、补码运算的加减指令，能够体现一条指令具有多种含义的特点；具有BCD码运算调整指令，能够体现底层数据的不同编码机制。这些特点正是各种计算机系统、包括现代的CPU中具有的最普遍、最本质的原理，并且在

学习和掌握8086/8088汇编语言的过程中所提炼的学习和思维方法，同样适用于其他CPU及其汇编语言。掌握了这样一些基本原理和基本学习方法后，自学其他更为复杂的汇编语言，将会成为不困难的事情。并且，由于桌面PC非常普及，Windows操作系统通常自带DEBUG等8086/8088指令系统的模拟调试程序，各种版本的8086/8088汇编器、连接器通常都能在PC上运行，使用方便，所以介绍8086/8088汇编语言也是非常有利于上机实践的。

本教材的内容主要基于8086/8088指令系统，试图以全新的诠释方式来讲解这种汇编语言。就本教材涉及的教学内容而言，可作为计算机专业本科生的汇编语言程序设计教材。本教材的特点在于：第一，由于初学者普遍感觉指令记忆很困难，而且枯燥，为了避免初学阶段记忆太多指令而抹杀了学习的乐趣，将指令系统中的指令分散到各个章节中，使初学者在学完指令系统的基础部分后即可进行编程实践，能够循序渐进，保持知识积累与学习兴趣的并行提升；第二，增强有关CPU结构及工作原理的说明，但同时避免讨论其细节，使初学者能够尽快地为自己搭建一个基本合理的模型计算机，并借助它加深对汇编语言的理解，并且，在掌握汇编语言之后，借助此模型机深化对计算机系统原理的理解；第三，结合实际的教学经验，在总结初学者容易产生的疑问与容易出现的错误理解后，举例加以说明；第四，突出硬件与软件的衔接，充分说明CPU中关键器件与指令系统中核心指令与程序设计的关系，使初学者能够真正深入理解软件结构是如何由底层硬件结构支撑起来的；第五，教材与上机实践密切结合，在详细说明编写、汇编、调试程序方法的基础上，教材中所列举的例子都可以直接上机实现调试。

本教材由唐宁九主编、统稿，其中第1章至第5章由李征编写，第6章由张刚编写，第7章、第8章由唐宁九编写，第9章、第10章由周群彪编写，第11章由郑成明编写。

在这里，我们期望这本以新方式诠释的教材能够为汇编语言初学者提供一点微薄的帮助。由于编者水平有限，时间比较仓促，疏漏之处在所难免，欢迎广大读者提出宝贵意见！

唐宁九

2007年9月20日

目　录

第1章 引 言

1.1 二进制编码与计算机系统

在种类繁多的计算机应用中，我们能够见到丰富多彩的信息表达形式，也能够使用功能各异的应用程序。例如，我们可以使用各种媒体播放软件观赏众多的视频和音频文件，能够欣赏到五颜六色的画面和悦耳的音乐；可以使用各种文字处理软件编辑文档或源程序，从而制作由不同文字组成的文本，这些软件提供的编辑、查找功能使我们能够方便地对文本进行修改、浏览；可以使用数据库管理软件定义各种数据库，在数据库中存放种类丰富的数据。但是，计算机应用领域中这些形式各异的软件和数据形式在计算机底层的描述却具有共通性。这种共通性则是指，无论两个应用程序的功能或性质有着多么大的不同，无论两种数据形式的性质有着多么大的不同，它们都是使用由“0”和“1”两种数字信号的排列组合来表达的。一般而言，现代计算机硬件系统只能鉴别“0”和“1”这两种信号，看起来十分单一，但是通过不同的排列组合，这两种信号组成的序列可以表达丰富多彩的信息，而我们平时在计算机应用中见到的不同信息表达在底层正是这样表述的。这种在计算机系统中，由“0”、“1”两种信号排列组合而成的序列，我们称之为编码。

在计算机系统中的二进制编码是需要人来解释的，只有当人们按照某种既定的编码规则来解释其含义时，二进制编码才是有意义的。例如，在任意一个存储单元中，保存了“00110101”这样一个二进制编码，对它的解释可以有很多种，它可以是一帧图像中某个像素位置的亮度表示；可以是音频数据某个采样点上的音量幅度；可以是坐标系中的一个坐标值；可以是一个字母的编码；可以是另一数据的地址；也可以是CPU（中央处理器）执行的一条机器指令。那么，初学者通常会问，它究竟具有什么含义呢？这个问题恰好忽略了提问者本身，软件的设计一定是由人来完成的，因此，系统中信息的解释也就是由人来定义的。如果我们把上述那个编码解释为像素的亮度值，会将它送到显示卡的显示存储器中，使它能够在屏幕上显示出来；如果我们把它解释为音频数据中的音量幅度，会将它传送到音频卡，使它能够在音箱中播放出来；如果我们认为它是一个坐标值，或许会将它用于坐标变

换；如果把它看作是一个字符编码，我们会将它用于字符模板的查找，并将字符模板送到显示卡，最终将字体形状显示在屏幕上；如果我们认为它是程序中的一条指令，会将它送到CPU去解释并执行。

经过上面的阐述，我们认识到计算机系统底层中的信息是需要人来解释的。事实上，如果站在较高层面上，操作系统、应用程序这类软件也会帮助人们来解释这些信息，但归根结底，信息的解释是由人来完成的。这个问题看起来似乎很容易理解，或许我们以前早就理解到了。但是，当我们面对加法指令同时完成无符号数和补码两类加法运算、同时影响两类标志位时，我们可能就会忘记这样的问题，因而不能解释指令使用的究竟是什么编码，从而陷入困惑。在正式进入汇编语言的学习殿堂之前，应当建立起人解释底层编码含义的观念，为解决将来学习过程中的困惑做好准备。

1.2 汇编语言基本概念与术语

CPU能够直接执行的程序，是由机器指令序列组成的程序，CPU从内部存储器中逐条读取机器指令，并加以解释、执行，并且CPU会将这个过程不断持续下去。上一节已经阐述过，使用二进制编码表示的指令即是机器指令。例如，我们可以使用代码“001”表示加法操作，使用代码“010”表示减法操作，使用代码“011”表示逻辑与操作等等。机器指令是可执行程序的最小组成单位，它具有原子性，CPU要么完整地执行一条机器指令，要么完全不执行它。

首先，机器指令实际上是最底层的软件元素，那么它与汇编语言有什么关联呢？汇编语言中的汇编指令即是机器指令的助记版本，在汇编语言中使用英文单词缩写来表达指令功能，而不是使用二进制代码。例如，我们可以使用“add”表示加法操作，使用“sub”表示减法操作，使用“and”表示逻辑与操作等等。正因为如此，人们记忆汇编指令会比记忆机器指令容易得多。对于一种特定的CPU，其机器指令的全集构成机器指令系统，其对应的汇编指令全集也就构成了汇编指令系统。原则上来说，机器指令与汇编指令是一一对应的关系，但是也不排除个别例外的情况。这种例外的情况在8086/8088指令系统中也是存在的，在后面的章节中会给出相应的说明。另外，应当注意，不同型号的CPU具有不同的指令系统。其次，汇编指令（或机器指令）和高级语言中的语句有什么区别和联系呢？我们可以通过如下的例子来感受一下。

例1.2.1 下面的C语言语句与汇编语言程序片段具有相同的功能。

C语言语句：

```
w = x+y-z;    /*x变量加y变量，再减z变量，运算结果保存到w变量中*/
```

相应的汇编语言程序片段：

```
mov al, x        ; 将变量x的内容传送到al寄存器
```

```
add al, y       ; 将变量 y 的内容与 al 寄存器的内容相加，结果保存到 al 寄存器
sub al, z       ; al 寄存器的内容与变量 z 的内容相减，结果保存到 al 寄存器
mov w, al       ; al 寄存器的内容传送到变量 w 保存
```

从上面的例子可以看出，在C语言中的一个赋值语句，如果使用8086/8088 CPU的汇编指令系统表述，需要至少4条指令。可以得出这样的结论：高级语言中的语句是面向问题描述的，其表述形式更接近于人类描述问题的方式，语句虽相对复杂，但可读性、功能性均较强；汇编指令则不同，它们是面向机器操作的，其表述形式更接近于人类操作机器的方式，每条指令相对简单，功能性较弱，但对于不熟悉指令系统的初学者，程序片段的可读性并不高。如果程序较长，那么阅读和理解汇编语言程序比阅读和理解高级语言程序要困难得多。

无论是使用高级语言还是汇编语言所编写的程序，最初都是一种文本格式的程序，概念上称为源程序，而源程序是不能被执行的。使用汇编语言书写的源程序通常称为汇编语言程序。对应的，由机器指令组成的，能够被CPU直接解释、执行的程序称为目标程序（或目标代码）。源程序必须先经过翻译过程，转换为目标程序后，才能被CPU执行。对于高级语言，其语句与机器指令并不是一一对应的，因此这样的翻译过程比较复杂，称为编译过程，完成编译过程的应用软件称为编译器。对于汇编语言，其指令语句与机器指令基本上是一一对应的，因此翻译过程相对简单，称为汇编过程，完成汇编过程的应用软件称为汇编器（或汇编程序）。如果是将内部存储器中的机器指令序列翻译为汇编指令，以达到分析程序的目的，那么这样的过程称为反汇编过程。一个完整的汇编语言程序除了包括汇编指令系统中的指令外，还包括其他一些语法要素，而这些语法要素与具体的机器指令系统无关。例如，汇编语言中的伪指令，它们能够完成变量定义、空间分配、定义标号、定义常量、条件汇编、重复汇编等功能，但它们的解释与执行是由汇编器来完成的，而不是由CPU直接完成，并且，伪指令的执行阶段是在源程序的汇编阶段，而不是目标代码的执行阶段。因此，汇编语言是一种低级语言，它由某种特定CPU型号对应的汇编指令系统与其他一些与指令系统无关的语法要素、语法规则构成。

1.3 汇编语言在计算机科学中的位置

从1.2节的阐述中，我们明白了汇编语言与高级语言的区别，高级语言是面向问题描述的，其源程序容易被人们读懂，程序编写也相对容易；而汇编语言则是面向机器操作的，其源程序阅读和理解相对困难，程序设计也相对繁琐。那么，我们为什么还要花费那么多精力学习它？花费那么多时间来做一件貌似事倍功半的事情呢？这与汇编语言在计算机科学中的特殊位置是紧密联系在一起的。下面将从理论和应用两个方面来阐述汇编语言在计算机科学中的位置。

首先，CPU从产生至今，其核心原理中始终保持了周而复始地读取机器指令、解释机

器指令、执行机器指令这样一个基本机制。只要这种机制继续存在，机器指令系统就会始终存在，因此，汇编语言也会随之存在下去。照目前的发展趋势来看，CPU 的这种基本机制在相当长的时间内不会出现本质的改变。由于汇编指令实际上是机器指令的助记版本，因此学习汇编语言就是学习机器指令系统。那么学习指令系统的必要性和重要性是什么呢?

总体上，计算机科学可以分为硬件和软件两个大的分支，而汇编语言正是衔接这两个分支的重要衔接点。可以这样说，汇编语言既是深入学习计算机硬件工作原理的必要基石，也是理解硬件结构对各类软件实现支撑的关键原理所在。例如，计算机组成原理课程是一门深入剖析 CPU 内部结构和原理、深入阐述指令系统设计原理、深入分析指令微操作的课程。那么，如果不理解指令系统以及相关的计算机系统基本原理，则学习这门课程的基础将不充分。对于微机原理与接口技术这样的课程，指令系统已被作为基本工具使用。换言之，如果没有掌握汇编语言，那么深入学习计算机系统的硬件工作原理几乎是不可能的。对于软件相关的课程，汇编语言与它们的学习没有直接的关联，但是如果没有掌握汇编语言，将影响对底层软件工作原理的理解，影响对软件优化的理解。

在现代计算机的应用领域，汇编语言的应用仍然是比较广泛的。首先，它可用于跟踪或监控正在运行的应用程序，例如针对某些病毒程序的跟踪或分析，由于该病毒程序不是自己编写的，没有源代码，那么通常所使用的程序跟踪工具都是将机器指令反汇编为汇编指令，通过阅读汇编语言程序来分析程序的功能。因此，汇编语言是调试、分析系统底层程序的基本工具。其次，目前一些对响应速度要求较高的系统中，其软件的核心部分都是使用汇编语言来编写的，因为相对于高级语言，汇编语言程序产生的机器指令序列要短小得多，执行效率也高得多。因此，由于汇编语言程序的时间、空间效率优势，在计算机应用中，它仍然是对高级语言程序的一种必要补充。

习题 1

1. 试简述汇编语言与高级语言的区别与联系。
2. 为什么说汇编语言是最接近计算机系统底层的语言？请结合自己的理解加以叙述。
3. 编译器与汇编器的概念是什么？它们有什么区别?
4. 简述机器指令、汇编指令的概念，并说明它们的区别与联系。

第 2 章　数与编码

2.1　进位记数制

记数的概念是伴随着人们对数量概念的认识而产生的。人们最早对物品记数时，为了区分物品数量的具体差异，可能会使用小石子的数量来代表体积较大物品的数量，这就是最原始的记数概念。但是数量太大以后，小石子可能越来越多，也就无法清晰地展示物品的数量了。那么，人们可能很快想到用较大的石子来代替一组小石子，这样记数又重新清晰起来，这种代替操作就是原始的进位概念。使用进位来帮助记数，会使记数结果非常清晰，进位记数制的概念由此产生。在小学阶段，我们学习基本四则运算时，做加法总是逢十进一，此时我们实际上使用了进位记数制中最普遍的一种，即十进制，基于十进制的数字符号我们称为十进制数。我们在学习经典数学理论时，在日常生活中，所说、所见、所写的数字通常都是十进制数。那么，有没有其他的进位记数制呢？回答是肯定的。计算机系统内部使用的就是二进制编码，而二进制编码是建立在二进制数基础之上的。

数与编码是两个不同的概念，数是指理论上的数值，进位记数制正是数在理论上的符号表达形式，而编码则是数在数字系统中的表达形式。二者的区别将在 2.3 节讨论编码时给出详细的阐述。这里讨论进位记数制，是因为计算机系统所使用的进位记数制与人们通常使用的十进制不同，是为了清晰地展现二者间的区别与联系。使已经牢牢接受了十进制数的人们能够站在更高的层面上，从整体理论角度来重新认识进位记数制，使人们能够接受二进制数，从而深刻理解系统底层的编码理论、指令功能。在此基础上，才能进一步接受在程序设计中常用的八进制、十六进制数。在讨论进位记数制的本质前，我们先来观察一下十进制数的多项式形式。

例 2.1.1　十进制数 5769.32 的多项式表达如下式所示：

$$5769.32 = 5 \times 10^3 + 7 \times 10^2 + 6 \times 10^1 + 9 \times 10^0 + 3 \times 10^{-1} + 2 \times 10^{-2}$$

从上例的多项式表达中，我们能够看出什么规律？抽象出怎样的一般概念呢？首先，按

照多项式表达形式，我们可以抽象出“系数”、“数位”、“权值”、“基数”的概念。例中的5，7，6，9，3，2都是多项式中的系数，它们各自所在的“千位”、“百位”、“十位”、“个位”、“十分位”、“百分位”即是各系数所在的数位。例中的10^3，10^2，10^1，10^0，10^{-1}，10^{-2}则是数位对应的权值，它们是与数位对应的，可以观察到，权值表示为指数形式。其中“10”为“基数”，此10与“十进制”中的“十”是对应的。我们可以容易地得出一般性的结论，十进制数表达形式其实是系数按照所在数位对应的权值大小，依降序排列而成的系数序列。系数同时也解释为表达数字的基本符号，那么在十进制中，这样的系数总共有多少呢？我们都很清楚，在十进制中，系数从0到9，总共有10个系数符号，系数符号的数量等于基数。那么，我们会问，这些规律能够一般化吗？如果是R进制数，它的系数符号有多少个呢？它可以表达为这种多项式形式吗？我们先来看一个例子。

例 2.1.2 几种进位记数制的系数列举如下：

十进制数的系数：0，1，2，3，4，5，6，7，8，9
二进制数的系数：0，1
三进制数的系数：0，1，2
四进制数的系数：0，1，2，3
八进制数的系数：0，1，2，3，4，5，6，7
十六进制的系数：0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F

我们可以得出一般性的推论，R进制数的系数共有R个，从0到$R-1$，但是要注意，单个系数必须是单个符号，例如十进制的“12”，在十进制中它不是单个系数，但在十六进制中它却是单个系数，用“C”表示。而数值R即为R进制数的基数。R进制数的标准书写方法为：$(\text{系数序列})_R$。任意的R进制数都可以采用多项式来表出，如下例所示。

例 2.1.3 整数部分n位、小数部分m位的R进制数，可使用多项式表达形式如下：

$$(K_{n-1}K_{n-2}\cdots K_1K_0.K_{-1}K_{-2}\cdots K_{-m})_R = K_{n-1}R^{n-1}+K_{n-2}R^{n-2}+\cdots+K_1R^1+K_0R^0 + K_{-1}R^{-1}+K_{-2}R^{-2}+\cdots+K_{-m}R^{-m}$$

其中K_i为R进制系数，变化范围从0到$R-1$。

2.2 不同进位记数制间的转换

在2.1节中，我们明确了进位记数制的概念，接下来，我们要讨论不同进位记数制间的转换问题，这对于初学者理解二进制数是有益处的，并且，这样的数制转换在程序设计中也经常会使用到。首先，我们会提出这样的问题，理论上同一个数值是否可以用不同的进位记数制来表示？不同的进位记数制表示相同数值时得到的不同表达形式，其本质含义是什么？下面我们先看一个例子。

例 2.2.1 考虑十进制数“5”在数轴上的位置，以及在其他进位记数制中，它的表达