计算机与汇编语言导论

——360/370系统

〔美〕F. D. 维克斯著

科学技术文献出版社重庆分社

一九八一年十一月

出　版　说　明

为了普及计算机知识，推广应用计算机技术，更好地开展情报服务工作，我所出版了“计算机和汇编语言导论—360/370系统”一书。本书是作为计算机科学的中级教程编写的，包括了著名的ACM-68大纲规定的中级教程的大部分内容，可作为有关专业的教学参考书，对希望进一步学习计算机软件和硬件的读者也有帮助。本书由西安延河无线电厂毛成锦译，王毅校，该厂情报室承担了本书的出版编审校对，承蒙协助，谨致感谢。

计算机与汇编语言导论

——360/370系统

Introduction to Machine and
Assembly Language:System/360/370

〔美〕 F·D·VICKERS

毛成锦译　王　毅校

出　版　　科学技术文献出版社重庆分社
发　行　　重庆市科学技术情报研究所
印　刷　　科学技术文献出版社重庆分社印刷厂
书号：15176·522　　内部发行　　工本费 1.70元

目　录

前　言

本书是作为计算机科学的中级教程来写的，主要是在机器语言水平上讨论计算机的工作，包括数据和指令的各种表示形式；从程序人员角度看到的机器的基本结构；以及在程序控制下计算机的工作情况。针对某一实际的机器来进行此类问题的讨论较之于一般的论述为好。选择了360系统计算机，因为它的销售量最大，并且是作者有机会使用的一种机器。

在本书的讨论中所包含的是360汇编语言的一个不完全的叙述。本书之所以没有提供这种语言的一个完整的表示，因为好多汇编特性只有很熟练的系统程序员才能使用。更为重要的是要使学生们对计算机的工作在各方面都能有很深刻的理解，汇编语言只是为此目的服务的一个工具。同样，FORTRAN语言是用来作为输入输出的手段，当然这只有对汇编语言输入/输出的学习有了相当好的基础才行。

本书包括了ACM教程68的中级教程的大部分内容，这些是：

1.机器结构与语言；寄存器、存储器、控制以及输入/输出；指令类型、格式及其执行；运算、程序控制、输入/输出操作和中断。

2.寻址技术，包括绝对的、变址的、间接的、相对的，以及基址寄存器寻址。

3.整数、浮点数、十进制数、字符和逻辑数的表示以及转换。

4.汇编系统，包括助记码的符号编码、标号和地址、文字符号、扩展操作和伪操作，系统和用户的宏编程也包括在内。

5.在相同和不同语言之间程序的分段和连接；内存分配和可重入编程；系统组织，包括分级存储。

6.IBM370系统中的新发展。

在这里，我对国际商用机器公司表示感谢，感谢它慷慨地允准重印摘自各种360系统出版资料的一些数据和表格。这些出版资料的编号是：A22-6821-7，C20-1646-4，C28-6535-1，C28-6646-0，C28-6647-1，GA22-7000和X20-1703-6。

我也愿借此机会对那些直接地或间接地对本书的概念和构思以及论述方法给予相当影响的人表示感谢。依照时间的先后，他们中的一些人是：W·W·彼得森，C·A·泰勒，A·A·布罗伊莱斯，R·G·塞尔弗里奇，C·B·佩里曼和R·N·布拉斯韦尔。对于我的妻子在全书编写中所表现的耐心、鼓励和帮助表示特别的感谢。

F·D·维克斯

1971年6月于佛罗里达州

格因斯维尔

定 义 性 表 格

第一章　计算机的软件系统空间

本章给出了为掌握高速电子计算机的组织结构和性能所必需的基本知识。这里介绍的不是电子线路方面的知识，而是对用户和计算机科学家更为有用的机器结构和机器操作。这可以称之为计算机设计特性的学习，或者换言之，是基本编程条件下的计算机特性。我们还可以说，这是关于基本机器语言的学习。在大多数院校以及实际工作情况下，这就成为关于汇编语言的学习。

为了对计算机和信息科学的基本概念有个透彻的理解，学生们必须完全掌握有关计算机基本工作情况的知识。这样的知识，只有通过广泛的练习才能牢固地获得，这包括用机器本身的语言进行实际编程的练习。

在这里，假设读者具有某些使用计算机和计算机编程的经验，特别是有使用诸如FORTRAN或PL/I等高级语言的经验。这些高级计算机语言一般是由编译程序或解释程序翻译成基本机器语言，才能完成实际的计算。所以使用FORTRAN或PL/I语言的用户已经间接涉及到本书的内容。毫无疑问，这样的经验一定会有助于他们了解现有的其它程序语言和系统。本章的目的是以一种由表及里的方法讨论广泛的计算机系统，这些系统使用在现代化的大规模计算环境中。

1.1　系统-空间参量

本节及下一节将展示一种概念，用以尽可能精确地描述计算系统空间的通用性。这种空间是一种抽象空间，包括所有编程语言和/或系统。正像其它系统空间一样，计算系统空间具有一定的维度，但与其它空间不同，这些维度不能用通常采用的单位来度量。

计算系统空间的三个主要维度是复杂性、面向性和步骤性，步骤性是标志程序性程度的术语。所有测量单位是定性的，而不是定量的，因此不同维度用低、高或居中之类的术语来描述。

复杂性是关于使程序员的工作从对细节的冗繁而费时的注意力中解脱出来的程度的一个量度。例如，FORTRAN就是一种复杂性相当高的语言，而在另一极端，基本机器语言的复杂性就很低。用与FORTRAN编制程序的情况比较，很明显，用基本机器语言编制程序时，程序员必须对程序的步骤用多得多的细节进行描述。复杂性的一个很好的量度是源程序与相应目标程序的大小之间的转换比。假如在进行转换或编译时，某一程序可由少数几条指令扩展为大量的单个的机器命令，那么，就可以说这种语言是高度复杂性的语言。因此，当使用一种复杂性高的语言时，程序员就可以用相当少的语句来表述一种复杂的过程。

第二个维度可以测量两个极端之间的面向性程度，这两个极端是纯机器的面向性和纯问题的面向性。面向机器的语言是一种与计算机的基本功能和操作紧密相关的一种语言。另一方面，面向问题的语言与机器的基本功能性质是完全脱节的。一般说来，面向问题的语言允许用专用性术语来叙述问题，而这种术语对特定用户是熟悉的。在多数情况下，甚至用户不必确定解决问题的步骤，而只需指明是什么问题就行了。这样的面向问题的系统的几个例子是：简称GPSS的通用模拟系统，简称STRESS的结构问题解题系统，简称DYNAMO的连续系统或时间增量模拟系统和简称ECAP的电学电路解题系统。所有这些语言都允许用户用它日

常使用的专业术语来表述他手头的问题，而不需要设计解决问题的详尽步骤。衡量面向性的另一种途径是系统的通用性程度。系统越通用，在基本机器的能力范围内它就越灵活；相反，当一个系统通用性较差且应用范围较窄时，系统就很自然地丧失了属于基本机器特点的某些灵活性。因此，每当设计一种新的语言或系统时，必须在灵活性的损失与简易性的获得以及对某些所选用户的有用性之间权衡得失。

第三个参量，由于缺少一个较好的名称，作者就定其为步骤性。这个量度是用来描述一种语言的对步骤进行说明的程度。说一种语言是全步骤性的，那就是说，它要指明由计算机按步进方式完成的一系列操作。如上所述，某些语言并不产生步骤性的程序，而另一些语言几乎是纯步骤性的，如基本机器语言就是如此。自然，还有一些语言用步骤性衡量是居中的，在这些语言中要由系统本身事先确定一个过程中一定量的步骤，如 Snobol 语言就是这样。

建立这三种不同的实体作为维度可以使我们对大多数语言彼此进行有意义的对比，并且可以消除对某些语言描述时产生的混淆。如FORTRAN的特性在未将步骤性和问题的面向性分开以前仍然是不易确认的。看来很明显，FORTRAN语言是面向问题的（面向数学的），但还不像大多数面向问题的语言，FORTRAN语言是高度步骤性的。虽然在纸面上难以表示出一个三维空间，但它在语言表征中不易混淆的优点显然超过了缺点。

描述程序语言还有其它很多的方法，如学习和使用的难度，还有执行和翻译的速度或精度等，但是上面所给的三种参量似乎是描述程序语言的主要方法。从教学的观点来看，这些描述符提供了相对确切的方法，用以讨论现有成千上万种计算机系统和/或语言的特性以及它们之间的主要关系。

1.2 计算机系统空间

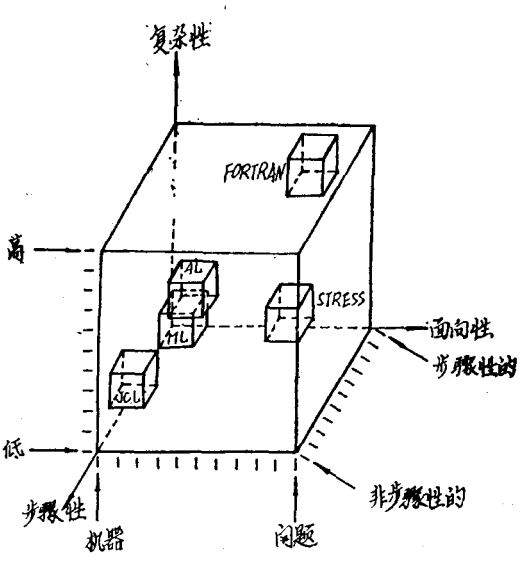

图1.1 计算机系统空间

图1.1展示了用以上讨论的三维参量表征的计算机系统的透视图。读者将会看到，这种空间由一立方体构成，此立方体置于三维空间之内，而其一角在原点。可以认为计算机处于此原点处。这种空间内的所有各种系统现在可看成是能由程序员决定的与机器发生关系的各种不同的方式方法。当某一系统离原点较远时，它就越加脱离机器本身的特性。

这个六面体左面表示完全面向机器的那些系统，右面表示完全面向问题的系统。顶面表示最高度的复杂性，而底面表示复杂性最低。正面表示非步骤性系统，背面则表示步骤性的系统。

为了避免混淆，图中只示出处于相应位置的几种语言。假如将所有系统都插进来，那么将会剩下一点点空间。假如还能剩下这点空间的话，或许这将位于立方体的边缘附近，具有非步骤性的、面向机器的特点。能够找得到的类似具有上述特点的语言结构就只有作业命令语言了。这些语言通常并不需要很多步骤来描述程序，但仍需使用计算机的物理组成来描述程序的参量，特别是输入、输出和存储设备。

在给出的计算系统空间内、在立方体原点处的最角上是机器语言。机器语言是这样一种

语言，即单独一个裸机就可以识别它。这是本文的一个主要内容。除此没有其它人们已知的方法能够给出关于计算机特性的清楚而精确的知识，而这对进一步学习计算机和信息科学来说是必需的。即使使用包含计算机电路设计在内的电学工程技术和专门方法也还是不够的，计算机设计者在作出最佳的设计之前，还必须具有从用户角度来看计算机怎样才会工作的详细而完整的知识。因此，了解计算机性能和概念的唯一合理的途径就是通过对机器语言及其应用的广泛深入的学习。

比机器语言略高一点且具有稍多一些的面向问题的语言是汇编语言。实际上很少有程序员用机器语言来写程序，通常是用汇编语言来代替。一般说来机器语言是很老的了，并且写出的形式叫人很难记住。大多数机器语言由机器命令和操作数组成，而它们又是由二进制数、字符或字符的某种组合编码的。这些编码命令并不是叫人阅读的，所以看起来是困难的。汇编语言是机器语言的符号形式，它包括助记忆操作码和操作数访问符号，这些都是为易于编程设计的。但是，这种作法的代价是必须进行翻译。这种翻译操作称为汇编阶段。对大部分程序来说，汇编语言操作与机器语言形式的命令是一一对应的，所以较之于FORTRAN语言的翻译，这种翻译是很简单的。因为汇编语言比机器语言易学易用，所以，以下章节的叙述使用汇编语言作为讲解机器语言基本内容涉及的一些概念的工具。在第一章第四节中将对汇编程序特性及其运行环境作进一步的讨论。

FORTRAN语言大概位于靠近顶部、靠近右面和十分接近背面的空间。还有比FORTRAN更复杂的语言，如PL/I。FORTRAN也并不完全是面向问题的，因为在这种语言中没有提供大量的数学操作。与其它许多类似的语言一样，FORTRAN有很多“变种”。几种相类似的语言是ALGOL、MAD、JOVIAL和BASIC，而FORTRAN的一些“变种”有：FORTRAⅡ、FORTRANⅣ（FORTRANⅢ出现的时间短暂）、FORTRANSIT、FLATRAN和QUICKTRAN。所有这些语言具有不同程度的复杂性、面向问题性和步骤性，而且要在系统空间内对它们准确定位可能带有很大程度的主观性。

系统空间中另一种语言的例子是STRESS。这种语言是与结构问题打交道的结构工程师们使用的。这种语言有很大程度的面向问题的能力，并且完全是非步骤性的。相对而言，正如图1.1所示，STRESS语言大约具有70%的复杂性。为了解决涉及结构的问题，程序员先要确定结构连接点的座标、连接点的型式、组成结构的梁的型式，和加到结构上负载的位置和数量。然后用STRESS系统解决这个问题，给出关于结构的必要信息，如连接点的反作用力、梁内的张力和压力，以及可表示该结构物理性能的其它压力或应力关系。这样，工程师可以用他熟悉的术语去解决问题，从而避免了为解决此类问题所需要确定的复杂的计算机步骤。STRESS系统中装有为分析结构所需要的这些计算机步骤，它们是由设计和完成该系统的系统程序员装入的。类似STRESS的大多数系统，最初是用汇编语言写成的，所以大多数系统程序员在他们的工作中必须知道和会用汇编语言。

作为讨论系统空间及其组成部分的最后一个例子是简称JCL的作业控制语言。多数计算系统都有某种形式的控制语言，用户用这种语言可以指挥他的作业通过该系统。虽然JCL命令的次序并不是完全无关紧要，但是这个次序又不像在FORTRAN程序或其它高度步骤性语言中对句子顺序要求得那样严格。同时，大多数JCL命令与物理设备或机器/系统的功能直接关系。因此，JCL是一种稍具特色的语言，有某种面向机器性、但又不是完全步骤性的。由于这种语言在学习和使用汇编语言时是需要的，所以在本书以后的某些章节里将给予一定的篇幅来讨论。IBM360系统计算机用的JCL有好几种形式，在大系统中应用的一种具有上述的特性。

1.3 供进一步学习用的一种特定汇编语言

作者认为，应该选择一种有实际用途的特定机器或机型，根据它的机器语言和汇编语言作进一步的学习。只有对一种实际运转的计算机进行广泛的应用实践和练习才能对这个主题有一个详尽的了解。虚构的计算机，即使它们具有在现有机器上工作的模拟器及其与之相关的系统，也永远达不到一种日常都在进行正常计算应用的现实系统所具有的完善性和复杂性。因此，作者选择了一种特定的系统，同时作者充分估计到本书的读者只是较少的一部分人，但是，它仍将像那些用虚构机器来说明问题的书籍一样有用。被选为特定机器的是IBM360系统，并且被选作特定管理系统的是一般称为OS/360的360操作系统。之所以选择这种特定机器，有两个很明显的原因：一是360系统是销售量最大的机器；二是作者碰巧有机会使用360系统的65型机。之所以在其他许多管理系统中选择OS系统，简单的原因是大多数360系统机器，如65型机，是在这个环境中运行的。对读者而言，OS的选择并不成其为一种限制因素，因为OS是从IBM公司得到的所有用于IBM360系统计算机上的各种管理系统中最完善和最通用的一种。

1.4 汇编语言的运行环境

在多数计算机系统中，汇编语言可用在程序的很多上下文中。本节将描述汇编语言在360系统中的一种典型用法。在很大程度上，这种特定的用法用在头九章里的编程举例和学生的练习中。正像任何一种高级语言一样，例如FORTRAN或PL/I，在可执行前要编译或翻译成目标程序，汇编语言也必须这样做。但是，汇编语言的翻译称为汇编，而不是编译。此外，在编译或汇编阶段与最终目标、执行阶段之间存在一种第二变换。在360系统中，这个新的操作称为连接编辑阶段。除此以外，360系统提供了一种组合程序段的能力，就是说，可将一些用几种不同语言写成的称为子程序的程序片段组合成一个较大的程序，作为一个单一的任务去执行。这样，做成一个总作业的各个阶段或作业步的数量可能就变得比较多了。

图1.2以非常一般的形式用图解法说明一个作业是如何被分割成许多作业步的。它进一步说明了每一作业步的输入和输出是如何与其它所有作业步骤关联的，它还示出了对各种输入都可使用的源和为输出提供的目的去处。这种特定作业由一个FORTRAN编译步、后面紧跟着的一个汇编步、连结编辑步和最后的执行步组成。其它的方案可包括把头两步倒过来，用PL/I步代替FORTRAN步，再加一个单独的PL/I步，或者涉及一种或更多种语言的许多其它的可能情况。

所以选择图1.2的特定方案是基于下述原因：本书所涉及的主要是汇编语言步以及所产生的程序对该计算机的影响。但是为了简便起见，在这本书的前九章里也用了FORTRAN，在这里，它是被当作进入汇编语言程序以及为它提供数据的手段。这就允许我们把关于汇编语言输入输出的讨论推迟到本书稍后的部分。在正规的OS/360条件下，关于输入输出的问题是非常复杂的，并且要求首先要有相当多的基础知识。

图1.2所示的第一个作业步是FORTRAN编译。可以将任何合理数量的、称为FORTRAN源模块的FORTRAN子程序提供给FORTRAN编译程序。可从系统库内获得一定数目的这类源模块，当然它们是由一先前的作业步存贮的。编译程序的输出是一系列目标模块，它们之中的每一个对应于一个源模块。这些模块保存于辅助存储器中，以备向连接编辑阶段输入。

第二个作业步是汇编阶段。可将任何合理数量的、称为汇编程序源模块的汇编语言子程序提供给汇编程序。另外，还可从系统库向该作业步提供若干源模块。汇编语言的输出是一

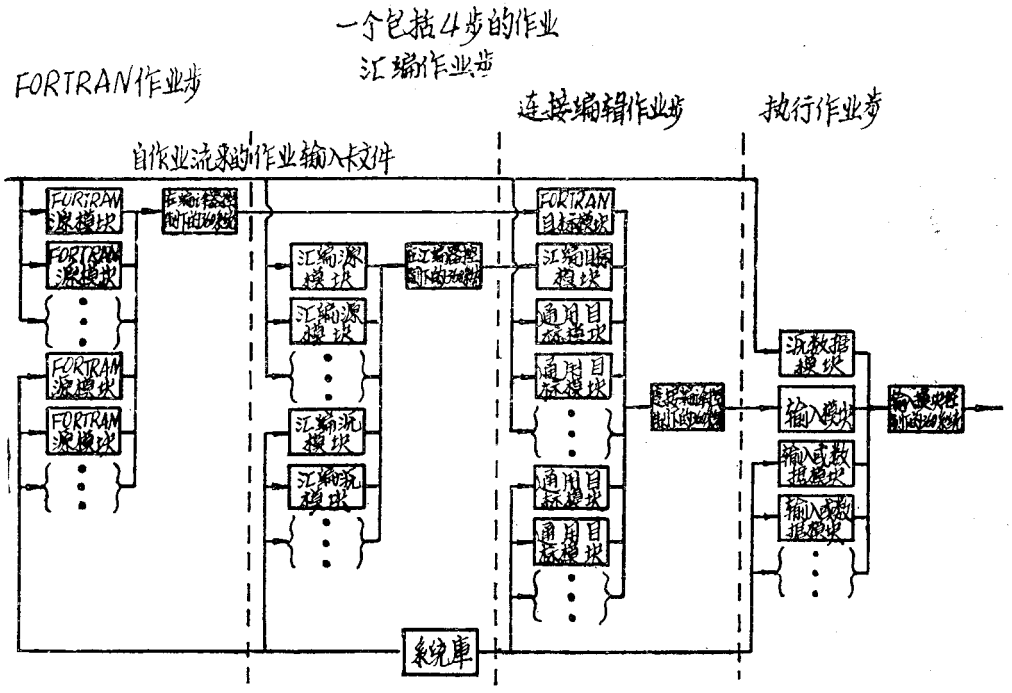

图1.2　包括FORTRAN和汇编语言在内的作业步

系列的目的模块，每一个与一种源模块相对应。此外，这些目的模块保存在辅助存储器中，以备向连接编辑阶段输入。

第三个作业步称为连接编辑，它把由第一和第二作业步产生的各种目标模块与任何以前被编译或被汇编的目标模块（这后者或是由户在此次运行时以卡片形式提供，或是由系统库提供）置于一起，对于作为操作系统的一个组成部分的连接编辑程序来说这是一个总的整体。连接编辑程序解决和完成各个目标模块之间的各种访问，并产生一个可执行的输入模块（可加载模块）。然后，将此输入模块与其它执行时所必需的输入模块加载到计算机的主存中。

最后的作业步是由计算机来实际执行被组合在一起的输入模块。如果各个子程序的书写是正确的，执行时提供的数据是正确的，并且计算系统的操作也是正确的，那么，将会输出所要求的结果。关于各个作业步和作业步控制的细节将在本书其余部分的适当地方介绍。当然，一个程序的执行成功与否决定于用户提供的作业控制信息，但是对于本文的主要意图来说，这个问题是次要的。

1.5　本书内容的组织

第二章对数制范围各个方面的问题进行了一定程度的讨论，这是全面理解现代计算机的操作和性能所要求的，这包括二进制数和十六进制数制。计算机的体系结构在第三章介绍。这是一种抽象的讨论，牵涉不到电子或逻辑线路的考虑。作者试图从第一代、第二代和第三代计算机之间在设计上取得的进展来描述这种结构。一般认为，如果与旧一些的概念进行比较，就会对较新的概念有更深刻的理解。第四章是就计算机的组织结构及其与汇编语言的关系给出的一个有关机器语言的导引。给出了一个解A＝B＋C的简单例子，并论证了包括整型数算术运算在内的实际计算机的操作。第五章介绍包括整型数算术运算在内的所有各种机器指令以及地址逻辑的各种基本概念。第六章是专门叙述转移操作的，这包括条件转移、循环

指令和变址控制。第七章介绍包括字符处理和码转换在内的各种操作。第八章和第九章介绍逻辑操作和十进制算术运算。第十章讨论实型数或浮点数算术运算的内容。程序中断、管理程序的服务与系统宏操作的概念是第十一章的内容。第十二章导入了在机器语言级上的输入和输出的广义化的概念。这里也给出了作为操作系统一部分的数据管理的导论。第十三章和第十四章是讨论在OS/360环境中完成输入和输出操作的两种特殊的方法。第十五章叙述了用户定义的宏指令的概念。最后一章，即第十六章，阐述了IBM360系统和370系统的主要区别。

作者认为，较之于许多简单的练习，少量设计周到、难度不同的问题更能增加学生的知识。因此，从第四章开始，在每一章的最后提供了一个单独的编程问题。当然，对于那些想测试一下自己学到的知识的学生，在每章的结尾也提供了一些练习。但是，必须强调指出，只有通过在由一个实际系统控制下的实际机器上的实际编程和运行相当数量的程序，才能对计算机的性能有一个真正的理解。因此，在一些叙述编程问题本身的章节里讨论了某些概念。所以，学生们至少必须读一读有关这些问题的叙述，以便从中得到附加的知识。

练　习

1. 试论系统空间的三维参量。
2. 以第一章介绍的概念试论机器语言和汇编语言的主要特点。
3. 试用计算系统空间的定性量度来比较示于图1.1的五种语言。
4. 用你知道的一种或多种语言与示于图1.1的五种语言进行对比。
5. 试为下述作业画一类似于图1.2所示的框图，此作业要包含一个FORTRAN源程序、一个汇编源程序和一个数据卡片组。它们均以卡片形式提供，要求有四个作业步。
6. 去掉汇编语言程序，重画练习5的框图，它能包括多少步？
7. 去掉FORTRAN语言程序，重画练习5的框图。它能包括多少步？
8. 假定在以前的作业中两个源程序已经编译和汇编，并且目标卡片组已穿孔，重画练习5的框图，仅此当前作业包括多少步？

第二章　数　　制

多数大型高速计算机并不使用十进制数作为它们的基本计算系统。它们使用的是二进制数制，这主要是经济上的原因。做成一个保存和运行二进制数的电子线路较之于十进制数的线路要容易得多。两个二进制数的值0和1可很方便地与诸如开关的打开和关闭或者灯的打开和关闭这样的普通电学现象相对应。所以，为了正确估价所完成的操作，这种机器的用户应该熟悉二进制数制。

为了方便的原因，还必须学习第二种数制。这第二种数制与二进制数数制有一个特别的关系，并被用于以较简短的形式来书写二进制数。这第二种数制的数基为16，并称之为十六进制数数制。

本章将通过对上述两种新的数制与十进制数制进行比较使读者了解所有数制的基本共同点。也将讨论从一种数制向另一种数制变换的方法，同时也要叙述在每一特定数制内的加法和减法这些基本数学操作。很明显，一种数制较之另一数制的优劣只能视其结果而论。例如，人类如果一开始就有十二个手指的话，那么今天应用最广泛的数制可能就是十二进制数了。

为了避免可能的误解，这里要指出的是，无需像在数论的准确论述中可能要求的那样去区别数制与记数制 (NUMERATION SYSTEMS)。在本章的各处，一个具体数的符号表示与其抽象值是可以交互使用的。严格地讲，只有一种自然数制，所以本章将要讨论的实际上是记数制。

2.1　关于十进制数制

对每种数制来说，最根本的是称为数基的值。十进制数制的数基为十。数基的值是以下列方式进入一个数的，如十进制数375实际上是下面这个表示式的缩写，即

$$3\times10^2+7\times10^1+5\times10^0$$

同时，有一点是很清楚的，那就是在表示式中每个数位的值是小于数基的，所以在十进制数制中每个数位的值必须在数字 0 与 9 之间，其中也包括 9，或者说，每个数位的值必须小于10。

十进制分数同样是包含有10的负方次的类似于上述表示式的记法，如十进制数 3.842 是下述表示式的记法：

$$3\times10^0+8\times10^{-1}+4\times10^{-2}+2\times10^{-3}$$

所有上述这些特定的情况可用一种一般的形式来表示。让我们来讨论一个十进制数，它的每个数位是以带下标的变量X来表示的：

$$X_m\cdots X_2X_1X_0.\quad X_{-1}X_{-2}\cdots X_{-n}$$

这个是以下表示式的记法：

$$X_m\times10^m+\cdots+X_2\times10^2+X_1\times10^1+X_0\times10^0+X_{-1}\times10^{-1}+X_{-2}\times10^{-2}+\cdots+X_{-n}\times$$
$$\times10^{-n}$$

上式的附加限制是$0\leqslant X_i<10$

现在可以确定一个对任何具有数基R的数制的定义。

2.1 具有数基R的数制

具有数基R的数的记法是

$$X_m\cdots X_2X_1X_0.X_{-1}X_{-2}\cdots X_{-n}$$

它代表的一个表示式是

$$X_m\times R^m+\cdots+X_2\times R^2+X_1\times R^1+X_0\times R^0+X_{-1}\times R^{-1}+X_{-2}\times R^{-2}+\cdots+X_{-n}\times R^{-n}$$

式中$0\leq X_i<R$

十进小数点（·）在十进制数中是被用来区分整数部分和小数部分的，这个小数点实际上在一般的定义中应该称为数基小数点。

2.2 二进制数制

二进制数是一种具有数基为2的表示式的记法。由定义2.1可知，在二进制数中每个数位（一般称为比特）必须是0或1（等于0或大于0，但小于2），并且在记法中每一处表示一个2的方次。所以，下述二进制数

1 0 1 1 0 0 1 (1)

是下式的记法：

$$1\times2^6+0\times2^5+1\times2^4+1\times2^3+0\times2^2+0\times2^1+1\times2^0 \quad (2)$$

对于一个以前从来没有用过二进制数的人来说，下述的表2.1包含了二进制数最初的十七个数，其名称是按照Bowers（鲍尔斯）和Bowers*建议的原则来起的，对读者来说很清楚起这些名称的原则和在十进制数中是一样的，只是把它应用于新的数基而已。当牵涉到要记忆最初十六个二进制数结构时，读者将会发现它是很有用的。

由表示式（2）的形式很容易得到一种将二进制数转换为与其等效的十进制数的方法。执行上式指明的数学运算，我们就可得到下面的结果：

$$1\times64+0\times32+1\times16+1\times8+0\times4+0\times2+1\times1=$$

$$64+0+16+8+0+0+1=89$$

*亨利·鲍尔斯和乔德·E·鲍尔斯著 Arithmetical Excursions: An Enrichment of Elenentary Mathematics（纽约道佛出版公司1961年版）。