TQ-16 计算机 ALGOL程序设计

李鸣山　　胡笔蕊

测绘出版社

本书主要讲述 ALGOL60 基本内容。为兼顾上机需要，书内还介绍了上机操作的有关内容，每章后附有习题，且书末有答案，便于自学。可供有关院校教学参考，并可兼作 TQ-16 机 ALGOL 编译系统的用户手册。

TQ-16 计 算 机
ALGOL 程序设计
李鸣山　胡笔蕊

测绘出版社出版
测绘出版社印刷厂印刷
新华书店北京发行所发行·各地新华书店经售

开本 787×1092　1/16·印张·20插页 2·字数 462 千字
1982 年 7 月第一版·1982 年 7 月第一次印刷
印数 1—9,800册·定价 2.10 元
统一书号：15039·新 223

前　言

本书是根据武汉测绘学院本课程的讲义修改而成。它既可作为算法语言程序设计的教学用书，也可作为 TQ-16 计算机 ALGOL 编译系统的用户手册。多年来的使用情况说明，它能较好地适应这两个方面的需要。

ALGOL60 是我国目前使用最广泛的程序设计语言之一。TQ-16机算法语言同其他国产计算机的算法语言一样，已根据实际需要和具体机器的特点对 ALGOL60作了某些限制和增删，但未作实质性的修改，因此，本书也可供学习 ALGOL60基本知识用，而且对学习ALGOL60 来说，这可能是一种比较好的方法，因为在学习了 ALGOL60 基本内容后，便可立即在机器上实践。

本书第一章介绍电子计算机的一些基本知识，对使用计算机来说，这是必要的。第二至七章是算法语言的基本内容。第八至十章是有关上机操作的内容，这几章对于仅仅是为了学习 ALGOL60 基本知识而不准备在 TQ-16 机上实践的读者可以不学。第十一章是几个源程序例子。除第八、九、十这三章外，每章后面都有比较丰富的习题，书末附有答案。

本书对 ALGOL60 基本内容的叙述，尽量做到系统性强，深入浅出，便于自学；对有关上机操作的介绍，尽可能结合到编译系统工作情况，并叙述得比较详细。

武汉测绘学院计算站对 TQ-16 机 ALGOL 编译系统作了一些修改，主要是关于源程序的上机修改和数组存贮的动态分配，已将它们作为附录列入书中。这些修改与原编译系统是完全相容的。

由于编者水平有限，书中难免有错，诚恳希望读者批评指正。

编者　1980.8.

目　录

试读结束：需要全本请在线购买： www.ertongbook.com

第一章　电子数字计算机概述

本章将对电子数字计算机作一简略介绍，使读者对电子数字计算机，特别是TQ-16机有一个初步了解，为学习和使用TQ-16机算法语言准备必要的知识。

为了叙述方便，以后我们常常将电子数字计算机简称为电子计算机、计算机或机器。

§1-1　电子计算机的组成部分和工作特点

电子计算机是由电子元件构成的、能自动地进行高速运算的工具，它通常由存贮器、运算器、控制器、输入设备和输出设备等五个主要部分组成。输入设备用来输入算题的原始数据及关于计算步骤的指示等信息。存贮器是计算机的记忆装置，用来存放通过输入设备输入的信息以及计算的结果。运算器是用来完成算术运算和逻辑运算的装置。输出设备用来输出计算结果或其它信息。控制器统一指挥计算机的各个部分协调地、有节奏地工作，从而自动完成计算任务。以上各个组成部分的相互联系见图1-1。图中实线表示数据及其它信息传送的路径，虚线表示控制信号的通道。除了以上五个主要组成部分外，电子计算机还有电源设备、控制台等附属设备。

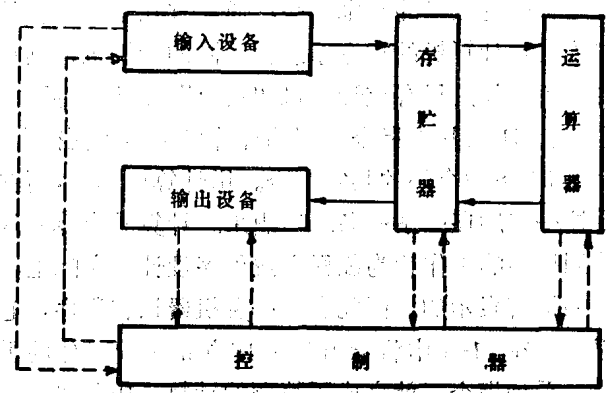


图 1-1 电子计算机各组成部分的相互联系

下面对存贮器作一些介绍，这些内容与程序设计直接有关。

存贮器由一系列存贮单元（简称单元）组成，被存贮的信息就存放在这些单元中。存贮单元总数称为该存贮器的存贮量或容量。存贮单元是顺序编了号的，各单元的编号，称为该单元的地址。存入存贮器的信息都以二进制编码，称为字，字的二进制位数称为字长，对一台具体的计算机来说，字长通常是固定的。一般地说，存贮器的一个单元存放一个字。例如，TQ-16机的存贮器，存贮量为32768，也称为32k❶，地址编号为0至32767，字长为48位，一个单元存放一个字。

我们可以将信息按指定的地址存入存贮单元，称为写入；也可以按需要根据指定的地址将单元中的信息取出，称为读出。应该注意，对任何一个存贮单元来说，在写入新的信息以前，它始终保留有原来所存的信息，也就是说，当从某个单元读出信息后，单元中仍保留有该信息，但若将新的信息写入该单元，则该单元原来所存的信息就消失了。

一般通用计算机的存贮器分为内存贮器和外存贮器，分别简称为内存和外存。内存与运算器、控制器直接联系，其中所存的数可以直接进入运算器参加运算。内存的存取速度

❶存贮量若为$2^{10}=1024$，则称为1k。$32768=2^{15}=32\times2^{10}$，即32k。

快，但容量一般不很大。目前国产计算机一般用磁心存贮器作为内存贮器。外存贮器用来存放计算过程中暂时不用的大量信息，它是内存贮器的后备。外存的存取速度比较慢，但容量很大；它所存的信息不能直接进入运算器或控制器，但它可以和内存成批地交换信息。目前常用磁鼓、磁带、磁盘作为外存贮器。

现在我们来看看电子计算机工作的主要特点。

电子计算机一般只能做几十种简单的操作，如算术运算、布尔逻辑运算、移位、两个数比较大小等等。对于一个具体的数值计算问题，如开方，求定积分的值，求微分方程的数值解等，在电子计算机上进行计算之前，要通过一定的计算方法转化为一系列加、减、乘、除等简单的运算，这些简单的运算都是计算机能够完成的。但这还不够，因为计算机并不“懂得”人们习惯的数学语言，所以还必须把计算的具体步骤用计算机能接受的语言表示，“指示”计算机某一步应该做什么运算，参加运算的数存放在存贮器的什么位置等，这种以计算机能接受的形式给出的有关计算的“指示”，称为指令。每一台计算机都规定有一定数量的基本指令，通常有几十种，有的有上百种。一台计算机的基本指令的全体称为该计算机的指令系统。不同的计算机，其指令系统一般是不相同的。因此，计算机的指令系统是只有该计算机“自己”才“懂得”的一种特殊语言。例如 TQ-16 机的指令系统有 51 条基本指令，别的计算机是不“懂得”这 51 条指令的，TQ-16 机也不“懂得”除这 51 条指令外的其它指令。用计算机算题，首先要将计算过程用机器指令系统中的指令按一定顺序一条一条地写出来。计算某个算题的所有指令的集合，称为该算题的程序，编制程序的工作称为编程序或程序设计。实际上，这里讲的程序是用机器指令这种机器特有的语言表示的，因此它是一种机器语言程序，它能够直接在机器上执行，通常称为手编程序。在 §1-4 中将给出 TQ-16 机的一个具体的手编程序，计算机自动地执行算题程序中的一系列指令，就完成了算题的计算。由此可见，电子计算机所以能自动地完成计算，完全是因为人们预先编好了程序的缘故。在程序控制下自动地进行工作，这是电子计算机工作的主要特点。

§1-2 计算机中数的表示

电子数字计算机是一种计算工具，计算的对象主要是数。那末在计算机中数是用什么方式表示的呢？数又是怎样进行运算的呢？对了解计算机的工作原理来说，这两个问题都很重要，不过对使用电子计算机来说，前一个问题更重要，因为它直接关系到原始数据的准备，关系到计算结果的输出形式。因此在这一节里，主要是讲第一个问题，其中涉及一些更具体的问题则留到 §1-3 介绍 TQ-16 机时再介绍。至于第二个问题，我们只在这一节里作简单的介绍。

一、进位计数制

（一）十进位计数制

在生产活动和日常生活中，用得最多的也是人们最熟悉的是十进位计数制的数，简称十进制数。十进制中采用 0，1，2，3，4，5，6，7，8，9 十个数字符号，它们

依次代表零至九，一个十进制数的任意一位必定是这十个数字符号中的某一个。对一个十进制数来说，其中的数字符号的意义除了取决于该数字符号本身外，还取决于它在数中所处的位置（或数位）。例如在 $32.395_{(十)}$❶中，自左至右，3 表示 3×10^{1}，2 在个位，它表示的数值就是它本身，即 2×10^{0}，小数点后的 3 表示 3×10^{-1}，9 表示 9×10^{-2}，5 表示 5×10^{-3}，因此，$32.395_{(十)}$ 还可以写成多项式的形式：

$$32.395_{(十)}=3\times10^{1}+2\times10^{0}+3\times10^{-1}+9\times10^{-2}+5\times10^{-3}$$

用十进制计数，当某一位满十就要向高位进一，因此，十进位计数制就是“逢十进一”的计数制。在十进制中，“10”代表“十”，这个“十”叫做十进位计数制的基数。

十进位计数制并不是唯一的计数制，在生产和生活中还用到别的非十进制的计数制。比如计时，六十秒为一分，六十分为一小时，这就是一种六十进位的计数制。我们还可以举出其他的计数制。在电子计算机中，用得最多的是二进位计数制。

（二）二进位计数制

二进位计数制简称二进制。二进制中采用 0 和 1 两个数字符号，它们代表零和一，二进制数的任意一位必定是 0，1 这两个数字符号中的某一个。与十进制数类似，二进制数中某个数字符号的意义由该数字符号本身及它在数中所处的位置确定。例如在二进制数 10.101 中，自左至右各个数字符号分别表示 1×2^{1}，0×2^{0}，1×2^{-1}，0×2^{-2}，1×2^{-3}，因此，$10.101_{(二)}$ 也可以写成多项式的形式：

$$10.101_{(二)}=1\times2^{1}+0\times2^{0}+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$

计算上式右端，得 $2.625_{(十)}$，就是说，二进制数 10.101 与十进制数 2.625 数值是相同的，即十进制数 2.625 用二进制表示是 10.101，二进制数 10.101 用十进制表示是 2.625。

用二进制计数，当某一位满二就要向高位进一，因此，二进位计数制就是“逢二进一”的计数制。在二进制中，“10”代表“二”，这个“二”就是二进位计数制的基数。

十进制数中 0 至 9 十个数字的二进制表示见表 1-1。

表 1-1 十进制数字与二进制数对照表

十进制数字	对应二进制数	十进制数字	对应二进制数
0	0	5	101
1	1	6	110
2	10	7	111
3	11	8	1000
4	100	9	1001

表 1-1 中十进制数字与二进制数的对应关系的正确性可以通过计算对应二进制数的十进制数值（如前面对二进制数 10.101 所做的那样）来验证，也可以通过对 0 逐次加 1 得到该对应关系，加的时候注意“逢二进一”。

电子计算机中之所以广泛采用二进制，是因为二进制有如下一些特点：

❶以后常常在一个数的右下角标以（十）、（二）等，表示这是十进制数，二进制数等。

（1）二进制中只有0和1两个数字符号，因此二进制数的每一位只需用一个具有两种不同稳定状态的元件就可以表示，而这种元件是比较容易获得的。例如氖灯的亮和灭、继电器的闭合和断开、晶体管的通导与截止、双稳态电路的高电位与低电位、门电路的正脉冲与负脉冲、磁心的正剩磁与负剩磁，等等，只要规定其中的一种状态表示1，另一种状态表示0，就都可以用来表示二进制数字。要制造多于两种不同稳定状态的物理元件，则困难得多。

（2）因为二进制数中只有0和1两个数字，所以运算简单。例如，它的加法表为

$$0+0=0$$
$$1+0=0+1=1$$
$$1+1=10$$

乘积表为

$$0\times 0=0$$
$$1\times 0=0\times 1=0$$
$$1\times 1=1$$

（3）由于采用二进制，就可以使用逻辑代数（或称布尔代数）这一数学工具来分析和综合计算机的逻辑线路，这就为计算机的逻辑设计提供了方便的工具。

但是，二进制也有它的缺点，比如，书写同样一个数，用二进制比用十进制一般来说要长得多，也不容易看懂。另外，由于人们习惯用的是十进制，因此原始数据一般都是用十进制数表示，计算结果也要求用十进制数表示，这就需要做十进制数与二进制数之间的大量转换工作，这一工作虽然可以用专门的转换程序由机器自动进行，但要占用不少机器时间。

（三）八进位计数制

八进位计数制简称八进制。八进制中采用0，1，2，3，4，5，6，7八个数字符号，它们代表零至七，八进制数的任意一位必是这八个数字符号中的某一个。八进制的基数是“八”，计数时“逢八进一”，“10”表示“八”。同十进制数和二进制数类似，一个八进制数也可以表示成多项式的形式，例如八进制数27.5可以表示成

$$27.5_{(八)}=2\times 8^1+7\times 8^0+5\times 8^{-1}$$

计算上式右端得$23.625_{(十)}$，即八进制数27.5与十进制数23.625数值是相同的。

由于$8=2^3$，而且我们知道八进制数字0至7与十进制数字0至7相同，它们用二进制表示是0至111，因此前式右端还可以改写成如下形式：

$$\begin{aligned}2\times 8^1+7\times 8^0+5\times 8^{-1}&=10\times(2^3)^1+111\times(2^3)^0+101\times(2^3)^{-1}\\&=10\times 2^3+111\times 2^0+101\times 2^{-3}\\&=(1\times 2^1+0\times 2^0)\times 2^3+(1\times 2^2+1\times 2^1\times 1\times 2^0)\times 2^0\\&\quad+(1\times 2^2+0\times 2^1+1\times 2^0)\times 2^{-3}\\&=1\times 2^4+0\times 2^3+1\times 2^2+1\times 2^1+1\times 2^0+1\times 2^{-1}\\&\quad+0\times 2^{-2}+1\times 2^{-3}\end{aligned}$$

上式右端用二进制表示就是10111.101，即八进制数27.5与二进制数10111.101值相同。

试读结束：需要全本请在线购买： www.ertongbook.com

实际上，将八进制数 27.5 的每一位按表 1-2 写成对应的三位二进制数，立即可以得到它的二进制表示 10111.101，这就是八进制数转换为二进制数的“一位拉三位”的方法（二进制数整数部分最高位和小数部分最低位出现的 0 都可以丢弃）。

表 1-2　八进制数字与二进制数对照表

八进制数字	二进制数	八进制数字	二进制数
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

反过来，将二进制数每三位写成对应的八进制数字，就立即可以得到该二进制数的八进制表示。例如二进制数 11110000.00111 的八进制表示是 360.16。

$$\underbrace{011}_{3}\ \underbrace{110}_{6}\ \underbrace{000}_{0}\ .\ \underbrace{001}_{1}\ \underbrace{110}_{6}$$

这就是二进制数转换为八进制数的“三位并一位”的方法。具体做法是：二进制数的整数部分从右到左三位合成一位，最后不足三位的左边以 0 补足成三位；小数部分从左到右三位合成一位，最后不足三位的右边以 0 补足成三位。

由此可见，八进制与二进制之间互相转换是很方便的。前面介绍二进制时已经提到，一个用二进制表示的数写起来很长，这很不方便，若用八进制书写，长度可缩短三倍，而且不需要数制转换的计算，因此计算机工作者普遍用八进制书写或从控制台上直读二进制数或其他二进制信息(如指令)。另外，前面也已经提到，人们习惯采用的十进制与机器通常采用的二进制之间的转换工作主要是由机器自动完成的，但有时也有少量数据的这种转换工作要用人工完成，这种转换是一位一位进行的，工作量大。我们可用八进制作为中间过渡的数制，方法是：将十进制数转换成二进制数时，先转换成八进制数，再由八进制转换成二进制。这样做，比直接转换方便，且工作量小（具体转换方法下面介绍）。将二进制数转换成十进制数，也是先转换成八进制数，再由八进制转换成十进制。

除上述几种计数制外，有时候还要用到四进位计数制、十六进位计数制等，它们与二进制、八进制的意义类似，这里就不再一一介绍了。

二、计数制之间的转换

为了书写方便，以下对十进制和二进制数间的转换，简称为十翻二、二翻十，记为10→2、2→10；同样，用 10→8、8→10、8→2、2→8 分别表示八进制与十进制、二进制与八进制数间的转换。前面已经介绍了 8→2 和 2→8 的方法；在计算 $10.101_{(二)}=2.625_{(十)}$ 和 $27.5_{(八)}=23.625_{(十)}$ 两例时指出 2→10 和 8→10 可用多项式计算进行；还指出了做 10→2 时，可先做 10→8，再做 8→2。因此，下面只介绍 10→8 的方法。

（一）整数10→8 ——“除八取余”法则

设有十进制正整数 $x_{(十)}$，求它的八进制表示 $x_{(八)}$。

所求的 $x_{(八)}$ 一定是八进制正整数，设它从最高位起依次是 α_n，α_{n-1}，…，α_1，α_0，即 $x_{(八)}=\alpha_n\alpha_{n-1}\cdots\alpha_1\alpha_0$，它还可以表示成下面的形式：

$$x_{(八)}=\alpha_n\times 8^n+\alpha_{n-1}\times 8^{n-1}+\cdots+\alpha_1\times 8^1+\alpha_0\times 8^0$$

因为 $x_{(八)}$ 与 $x_{(十)}$ 在数值上是相等的，所以下面的等式也是成立的：

$$x_{(十)}=\alpha_n\times 8^n+\alpha_{n-1}\times 8^{n-1}+\cdots+\alpha_1\times 8^1+\alpha_0\times 8^0 \qquad (1\text{-}1)$$

$x_{(十)}$ 已知，我们的目的就是将上式中的 α_0，α_1，…，α_{n-1}，α_n 确定出来。

式 (1-1) 两端除以 8，得

$$\frac{x_{(十)}}{8}=(\alpha_n\times 8^{n-1}+\alpha_{n-1}\times 8^{n-2}+\cdots+\alpha_1)+\alpha_0\times 8^{-1}$$

上式右端括号内是整数，即 $x_{(十)}$ 除以 8 的商数，α_0 就是 $x_{(十)}$ 除以 8 所得的余数，这样，就把对应八进制数的最低位 α_0 确定了。将商数再除以 8 得

$$\frac{\alpha_n\times 8^{n-1}+\alpha_{n-1}\times 8^{n-2}+\cdots+\alpha_2\times 8^1+\alpha_1}{8}$$

$$=(\alpha_n\times 8^{n-2}+\alpha_{n-1}\times 8^{n-3}+\cdots+\alpha_2)+\alpha_1\times 8^{-1}$$

上式右端括号内的整数是商数，α_1 是余数，这样就把 α_1 也确定了。

用同样的方法继续下去，就可以把 α_2，α_3，…确定下来，最后商为 0 对应的余数就是八进制数最高位 α_n。这就是整数 10→8 的“除八取余”法则。

〔例 1-1〕 将 $262_{(十)}$ 转换成八进制。

解：

	余数	
8 | 262	6…α_0	↑ 读出方向
8 | 32	0…α_1	
8 | 4	4…α_2	
0		

因此，$262_{(十)}=406_{(八)}$。

若进一步转换成二进制，则

$$262_{(十)}=406_{(八)}=100000110_{(二)}$$

对于负数，先对其绝对值做转换工作，最后在所得的八进制数前加上负号就是了。

（二）纯小数 10→8 —— “乘八取整”法则

设有十进制正的纯小数 $x_{(十)}$，求它的八进制表示 $x_{(八)}$。

所求的 $x_{(八)}$ 一定是八进制正的纯小数，设它从最高位起依次是 α_{-1}，α_{-2}，…α_{-m}，…，

即 $x_{(八)}=0.\alpha_{-1}\alpha_{-2}\cdots\alpha_{-m}\cdots$，它还可以表示成

$$x_{(八)}=\alpha_{-1}\times 8^{-1}+\alpha_{-2}\times 8^{-2}+\cdots+\alpha_{-m}\times 8^{-m}+\cdots$$

因为 $x_{(八)}$ 与 $x_{(十)}$ 在数值上相等，所以有下面等式：

$$x_{(十)}=\alpha_{-1}\times 8^{-1}+\alpha_{-2}\times 8^{-2}+\cdots+\alpha_{-m}\times 8^{-m}+\cdots \qquad (1\text{-}2)$$

下面我们由上式将 α_{-1}，α_{-2}，…，α_{-m}，…确定出来。

式（1-2）两端乘以 8，得

$$8\times x_{(+)}=\alpha_{-1}+(\alpha_{-2}\times 8^{-1}+\alpha_{-3}\times 8^{-2}+\cdots+\alpha_{-m}\times 8^{-m+1}+\cdots)$$

上式右端括号内是纯小数，α_{-1}是整数，即 α_{-1} 是 $x_{(+)}$ 乘以 8 的乘积的整数部分，这样就把对应八进制小数的最高位 α_{-1} 确定了。

将乘积的小数部分再乘以 8，得

$$8\times(\alpha_{-2}\times 8^{-1}+\alpha_{-3}\times 8^{-2}+\cdots+\alpha_{-m}\times 8^{-m+1}+\cdots)$$
$$=\alpha_{-2}+(\alpha_{-3}\times 8^{-1}+\cdots+\alpha_{-m}\times 8^{-m+2}+\cdots)$$

上式右端括号内是纯小数，α_{-2} 是整数，即 α_{-2} 是前一乘积的小数部分再乘以 8 所得乘积的整数部分，于是 α_{-2} 也确定了。

用同样的方法继续进行下去，可以继续确定出 α_{-3}，α_{-4}，…。这就是纯小数 10→8 的“乘八取整”法则。一般说来，一个十进制纯小数转换为八进制，是一个不尽小数，截取多少位根据需要确定，按“三舍四入”法则进行。对于负的纯小数，先对其绝对值做 10→8，然后再加上负号。

〔例 1-2〕 将 $0.634_{(+)}$ 转换成八进制。

解：

		整数部分				
		0	. 6	3	4	× 8
读出方向↓	α_{-1}…	5	0	7	2	
	α_{-2}…	0	5	7	6	
	α_{-3}…	4	6	0	8	
	α_{-4}…	4	8	6	4	
	⋮	⋮		⋮		

因此，$0.634_{(+)}=0.5044\cdots_{(八)}$。若舍入成三位小数，按“三舍四入”法则，得 $0.505_{(八)}$；若进一步转换成二进制，得 $0.101000101_{(二)}$。

对于既有整数部分又有小数部分的十进制数的 10→8 方法是：分别对其整数部分和小数部分 10→8，然后将八进制的整数和小数合在一起。

〔例 1-3〕 将 $-262.634_{(+)}$ 转换成二进制。

解：先将 $-262.634_{(+)}$ 转换成八进制。

由例 1-1，$262_{(+)}=406_{(八)}$；由例 1-2，$0.634_{(+)}\approx 0.505_{(八)}$，于是

$$-262.634_{(+)}\approx -406.505_{(八)}$$

而

$$-406.505_{(八)}=-100000110.101000101_{(二)}$$

所以

$$-262.634_{(+)}\approx -100000110.101000101_{(二)}$$

在用计算机算题时，有时需要用人工方法做十进制正整数与八进制正整数之间的数制转换工作。例如 TQ-16 机的单元地址是用八进制书写的，如果有 $8038_{(+)}$ 个数依次存放在地址为 070001 开始的单元中，要问最后一个数的存贮地址是多少，这就要将 $8038_{(+)}$

转换为八进制；又如有一个程序，它占用内存 7051 $_{(八)}$ 个单元，问相当于十进制数多少个单元，这就要将 7051 $_{(八)}$ 转化为十进制。为了便于做这种数制转化工作，我们给出了表 1-3 和表 1-4 两个表。

表 1-3 是整数 10→8 用表。从表中可查得十进制数 $n\times10^k$ ($n=1, 2, \cdots, 9$; $k=0, 1, 2, \cdots, 5$)的八进制表示。对于不是 $n\times10^k$ 这种形式的十进制数，10→8 的方法以下例说明之：

设有 8038 $_{(十)}$，从表 1-3 查得 8000 $_{(十)}$ = 17500 $_{(八)}$，30 $_{(十)}$ = 36 $_{(八)}$，8 $_{(十)}$ = 10 $_{(八)}$，将这三个八进制数相加得 8038 $_{(十)}$ = 17546 $_{(八)}$。

表 1-3　整数 10→8 用表

$n\times10^k$ 10^k \ n	1	2	3	4	5	6	7	8	9
$10^0=1$	1	2	3	4	5	6	7	10	11
$10^1=10$	12	24	36	50	62	74	106	120	132
$10^2=100$	144	310	454	620	764	1130	1274	1440	1604
$10^3=1000$	1750	3720	5670	7640	11610	13560	15530	17500	21450
$10^4=10000$	23420	47040	72460	116100	141520	165140	210560	234200	257620
$10^5=100000$	303240	606500	1111740	1415200	1720440	2223700	2527140	3032400	3335640

表 1-4　整数 8→10 用表

$n\times8^k$ 8^k \ n	1	2	3	4	5	6	7
$8^0=1$	1	2	3	4	5	6	7
$8^1=10$	8	16	24	32	40	48	56
$8^2=100$	64	128	192	256	320	384	448
$8^3=1000$	512	1024	1536	2048	2560	3072	3584
$8^4=10000$	4096	8192	12288	16384	20480	24576	28672
$8^5=100000$	32768	65536	98304	131072	163840	196608	229376
$8^6=1000000$	262144	524288	786432	1048576	1310720	1572864	1835008

表 1-4 是整数 8→10 用表。从表中可查得八进制数 $n\times10^k$ ($n=1, 2, \cdots, 7$; $k=0, 1, 2, \cdots, 6$)的十进制表示。对形式不是 $n\times10^k$ 的八进制数，8→10 的方法以例说明如下：

设有 7051 $_{(八)}$，由表 1-4 查得，7000 $_{(八)}$ = 3584 $_{(十)}$，50 $_{(八)}$ = 40 $_{(十)}$，1 $_{(八)}$ = 1 $_{(十)}$，将这三个十进制数相加，得 7051 $_{(八)}$ = 3625 $_{(十)}$。

三、数的定点与浮点表示

我们知道对一个数来说，小数点的位置是很重要的，因此电子计算机中表示二进制数时对小数点位置的处理是一个很重要的问题。最早的计算机中所表示的二进制数的小数点位置是固定的，通常是固定在最高数位之前，即所表示的数的绝对值小于1，这样表示数的方式叫定点方式，相应的计算机称为定点机。定点机实现运算的逻辑线路比较简单，但对使用电子计算机来说是不方便的。另一种表示数的方式是所谓浮点方式，即小数点位置不是固定的，相应的计算机称为浮点机。还有的计算机可同时采用定点和浮点两种表示数的方式，这种计算机称为定-浮两用机。

下面具体地介绍什么是定点数和浮点数以及它们在计算机中的表示方式。

一个十进制数 43.525 可以表示成

$$43.525 = +10^{+2} \times 0.43525$$

类似地，一个二进制数 100.011 可以表示成

$$100.011 = +2^{+11} \times 0.100011$$

右端这种形式显然不是唯一的，例如 $43.525_{(十)}$ 还可以表示成 $+10^{+3} \times 0.043525$，$100.011_{(二)}$ 还可以表示成 $+2^{100} \times 0.0100011$，等等。

一般地，任意一个 N 进制数 x 可以表示成

$$x = \pm N^{\pm j} \times w \tag{1-3}$$

其中：

N——基数。对十进制数，$N=10$；对二进制数，$N=2$；等等。N前面的正负号是 x 的正负符号，称为数符。

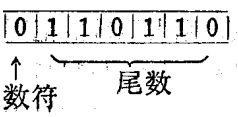
$\pm j$——阶，是 N 进制整数。j 前面的正负符号称为阶符，j 称为阶数。

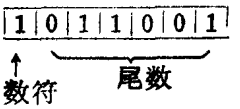
w——尾数，是 N 进制正纯小数，即 $0 \leqslant w < 1$。

当把数表示成 (1-3) 的形式时，如果对任何数，阶是固定不变的，这就是数的定点表示方式，这样的数称为定点数。例如，设基数 $N=10$，阶固定为 0，尾数 w 是 6 位，则十进制数 $x = \pm 10^{0} \times .w_1 w_2 \cdots w_6 = \pm w_1 w_2 \cdots w_6$，即小数点总是在最高数位前面；若固定阶为 1，则 $x = \pm 10^{1} \times .w_1 w_2 \cdots w_6 = \pm w_1 . w_2 \cdots w_6$，即小数点总是在最高位与第二位之间。可见定点数的小数点位置总是固定的。

对定点机来说，$N=2$，通常将阶固定为 0，这样的定点机只能表示绝对值小于 1 的数，如 +.110110，-.011001，等等。

为了进一步说明问题，我们假定某定点机用二进制 7 位表示一个定点数，其中最高位用来表示数符，并用 0 表示正、1 表示负，后 6 位表示尾数，则 +.110110 和 -.011001 在计算机中可分别表示如下：

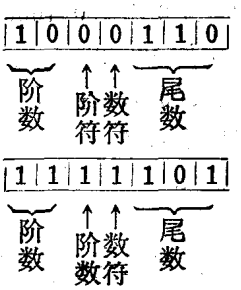




小数点在尾数最高位和符号位之间。

若把数表示成（1-3）的形式时，阶可以取不同的值，这就是数的浮点表示方式，这样的数称为浮点数。例如，设基数$N=10$，阶数是2位，尾数是3位，则十进制数$+10^{+01}\times.927=9.27$，$-10^{+03}\times.927=-927$，$10^{-02}\times.927=.00927$，可见，尽管它们的尾数相同，但因阶不同，小数点位置也随着不同，或者说是“浮动”的。TQ-16机中的数是用浮点方式表示的。

现在我们来进一步看看电子计算机中是怎样表示一个浮点数的。我们假定某浮点机用二进制7位表示一个浮点数，其中阶占3位，阶符占1位，阶数占2位；尾数占3位，数符占1位。阶符和数符皆以0表示正、1表示负。则二进制浮点数$+2^{+10}\times.110(=+11.0)$和$-2^{-11}\times.101(=-.000101)$在计算机中可分别表示如下：



阶符、阶数、数符、尾数所占的位置随计算机不同而可能不同。

一个数的浮点表示形式不是唯一的。例如二进制数0.00101可以表示成$+2^{-10}\times.101$，也可以表示成$+2^{-01}\times.0101$。如果限定尾数是3位，则后一种表示形式成为$+2^{-01}\times.010$。由此可见，前一种表示形式有三位有效数字，而后一种表示形式只有两位有效数字，损失了一位。前一种表示形式的特点是尾数最高位不为0（对二进制数来说就是1），即尾数w满足$1/N\leqslant w<1$（对二进制是$1/2\leqslant w<1$），称为数的规格化浮点表示，这样的数也称为规格化浮点数。因为浮点机中用来表示尾数的位数是一定的，为了在运算过程中尽可能不丢失有效数字，提高运算的精确度，一般都采用规格化浮点表示。一个非规格化浮点数通常可以通过调整阶并将尾数移位变成规格化浮点数。例如$+2^{+11}\times.011$，将它的阶减1（相当于乘2^{-1}），尾数左移1位（相当于乘2^{+1}），就变成规格化数$+2^{+10}\times.110$。再如对$+2^{+00}\times.001$，可以将它的阶减2，尾数左移2位，变成规格化数$+2^{-10}\times.100$。

最后我们对数的定点和浮点两种表示方式作一比较。

在计算机数位一定的条件下，采用浮点表示的数的范围比采用定点表示的范围要大。比如，我们假定计算机是用二进制7位表示数。若采用定点表示，且$j=0$，并规定用其中1位表示数符，其余6位表示尾数，则表示数的范围是

$$.000001\leqslant|x|\leqslant.111111$$

这相当于十进制数范围$2^{-6}\leqslant|x|\leqslant1-2^{-6}$，即$1/64\leqslant|x|\leqslant63/64$，绝对值小于.000001的

试读结束：需要全本请在线购买： www.ertongbook.com