高等学校教学用书

数字計算机原理和元件

SHUZI JISUANJI YUANLI HE YUANJIAN

上　册

（原理部分）

祖冲霄編

人民教育出版社

高等学校教学用书

数字計算机原理和元件

SHUZI JISUANJI YUANLI HE YUANJIAN

上　册

（原理部分）

祖冲霄編

人民教育出版社

“数字計算机原理和元件”分上下两册。上册为“电子数字計算机原理”，下册为“数字計算机元件”。

上册分三部分。第一部分叙述电子数字計算机的分类、工作原理、計算机的性能和特点等基本概念。第二部分分别討論电子数字計算机的主要組成部分。第三部分深入地討論計算机的組成原則和計算机設計、定型、制造、調整和維护等問題。

下册分四部分。第一部分講电子管計算机元件。第二部分为晶体管元件。第三部分为磁元件。第四部分为其他元件。

上、下两册原来按两本书进行編写的。因此在叙述方法和使用符号上有所不同，請讀者注意。

数字計算机原理和元件

上册(原理部分)

祖　冲　霄　編

人民教育出版社出版(北京景山东街)

(北京市书刊出版业营业許可証出字第2号)

人民教育印刷厂印装

新华书店科技发行所发行

各地新华书店經售

統一书号 13010·1047　开本 787×1092 1/16　印張 18 1/4　插頁 1

字数 403,000　印数 0001—5,000　定价(6)平1.50

1961 年 9 月第 1 版　1961 年 9 月北京第 1 次印刷

目　录

第一部分

第二部分

第三部分

第一部分

第一章　电子数字計算机的基本概念

§1-1　緒言

由于科学技术的不断发展，很多科学和工程技术部門都愈来愈需要进行大量的、复杂的和准确的計算，一般机械式和电动式計算工具已經远远滿足不了这种迫切的要求。随着电子管，脉冲技术，固体物理，半导体等新技术的发展，在十几年前出現了电子数字計算机。

由于电子数字計算机的計算速度非常快，又具有特殊的"存儲"性能，使过去很多由于需要精确計算或需要很快得出答案而无法解决的問題今天能够得到解决。以天文气象学中的天气預报为例。其实，精确的預报天气的計算方法是早就有了，但是用一般手搖計算机計算"日預报"就需要一、二星期的时間，这当然是行不通的，因而只好采用大大簡化了的，凭經驗数据的方法。而用电子数字計算机就可以在一两小时，甚至于更短的时間算出精确的結果。至于原子能、人造卫星、火箭导彈等尖端科学技术的发展和电子計算机的发展就有着更为密切的联系了。

由于电子計算机可以在很短时間內求出非常复杂方程的数字解，使得許多工程技术中的实驗研究可以用它直接进行数学計算。并从許多方案中选擇最合理的方案，这就可以大大地节省物質材料和时間。在經济生活中的統計、会計工作、交通运輸等也都可广泛应用电子計算机。

电子数字計算机的出現，其意义远不止这些。由于它具有計算过程"自动控制"的特点，不仅能进行四則运算，而且还有着邏輯运算的能力，这就使它的前途远远超出了純"計算工具"的范圍。

我們知道，在人們的思維活动中，在脑力劳动的內容中，有一部分是遵循着固定的規則重复进行的，可用有限的式子描述，可以利用算术运算和邏輯运算解决。大家所知道的利用电子数字計算机进行自动翻譯就是一个很好的例子。

从一九四六年出現第一架电子数字計算机以来，这門科学得到了迅速的发展，从速度方面来講，从最初的每秒运算几百次，发展到每秒几十万次，甚至更高。尽管如此，但目前它仍然不能滿足生活实踐和科学发展的要求，无論在計算机的使用上以及組成方法上都还有不少沒有解决的問題。很多尖端技术要求把計算机的运算速度再提高几十倍，这个問題的解决除了从提高单个元件的工作速度着手外，还需从机器的結构，組成原則及使用方法上努力。

在如何更好地发揮計算机的作用上，长期以来存在着計算机的快速性，广闊的运算能力和使用上的复杂性之間的矛盾。往往一个需要化好几周才能編好的解題程序，在机器上只

花几小时就算完了，这个問題的解决需要靠机器制造者和使用者密切合作，共同努力。

关于計算机的可靠性問題，目前仍是个尖銳、突出的問題。在計算机中一个零件的变值或是从外来一个干扰信号，所影响的不是精确度的高低問題，而是正确和錯誤的問題。为了解决这个問題，不仅要提高元件的可靠性，而且也要从机器的組成原則着手，簡化机器的結构。

§1-2 数字机的分类

分类的目的在于能更深入地研究每一种类型机器的特点，从而能更好地利用这一特点、或是采取更好的办法和措施，来解决每种类型机器的特殊問題。当然，各种类型数字机都是有它們的共同性。这門課程所講授的，主要是它們的共同性問題。下面是几种常用的分类方法。

(一)按用途分

1. 計算用数字机 这种机器一般是用来解数学問題，如微分方程、代数方程等。計算的結果是用数目字打印出来的。它的特点主要是独立运算，并不参加实时控制；但往往运算量很大，要求有很高的运算速度及很大的存儲量。这种机器是固定不动的，周圍工作条件較好。为了防止运算錯誤，可以采用計算两次等办法进行校对。并不要求它一刻不間断的运行。

2. 自动控制系統用数字机 不同的自动控制系統对計算机提出各种不同的要求。有的自动控制系統(如防空和控制某些有爆炸性危險的車間等)要求計算机不能出任何差錯，而且要保証一刻不間断长期的运行。而有的自动控制系統(如导彈用的)則只要求在飞行的那一段时間內絕对可靠运行。在有的自动控制系統計算机直接发命令給执行机构；而有的自动控制系統中，計算机只是操作員的一个輔助工具，它帮助操作員分析、統計由各測量机构来的信息，进行方案比較等等，但操作命令是由操作員下达的。一般来講，对自动控制系統用計算机的要求是实时控制、高可靠性、不間断运行、并具有将測量来的信息轉換成机器的輸入信息或将計算結果轉換成控制执行机构所能接受的信息的能力。

3. 数据处理用数字机 目前它主要用在經济部門，起分类、分析、統計等作用。它的发展与运筹学，信息論，統計学等有关系。輸入、輸出量很大，但运算比較簡单。要求有很大的存儲設备，但存取速度并不高。目前的方向之一是如何使所需处理的数据直接进入机器，不需經过打孔員将数据打成打孔带或打孔卡片后再送入机器。在复杂的数据处理系統中，有时还采用几台計算机并行或串級运算。

(二)按所用基本元件分

1. 眞空管机器 主要元件为眞空管，但有的还用到大量的晶体二极管及磁心等等。

2. 晶体管机器 主要元件采用的是晶体管、磁心等等。

3. 磁心机器 主要元件为磁心，一般也用了些眞空管及晶体二极管。当采用晶体三极管作为中間放大环节时，一般称为磁心——晶体管机器。

当然，根据所采用的元件还可能有其他形式，如：变参数元件机器、微波元件机器等等。

(三)按其速度及大小分

1. 特高速 一般用来解原子能、火箭等方面的极其复杂的、計算量极大的問題。运算速度要求达每秒几十万甚至一、二百万次。这种机器无論使用上、維护上都很复杂。

2. 大型的 結构不太复杂，使用比較方便，操作速度約在每秒一至二万次左右。用来解計算量很大的科学研究及复杂工程技术等問題。一般装在大型計算中心，这样可以更好地发揮它的作用。

3. 中型的 用于計算工程技术及一般科学研究問題。速度約为每秒一千次至几千次。这种机器要求比較灵活、結构簡单。

4. 小型的 主要用来解决工程設計及一般計算量小的科学研究問題。每秒速度为几百次左右。对这种类型机器的要求是簡单、便宜、可靠、使用方便。

上面四种类型都是指的計算用数字机。虽然計算机的运算速度已經如此快了，但算一个問題往往仍需几十小时甚至一周的时間。因此，在提高可靠性的前提下，千方百計提高机器的运算速度还是目前計算机主要发展方向之一。

(四)按工作特点的划分 有一种分法是：

1. 通用机 "通用"是指的能在同一架机器上解决各式各样的問題，为此只需改变解題程序的結构。目前大多数計算机都属于这一种类型。对它的要求是指令系統多样化、存儲量大、运算速度快。目前的主要問題之一是編制程序的复杂和运算速度之間很不相应，往往需好几天才能編制出的程序在机器上的运算时間却只需一、二个小时甚至更少。

2. 专用机 "专用"是指的在同一架机器上只能解固定的一种或几种問題。程序一般是固定不变的。这样在使用时只需輸入原始数据，在使用上要比通用机方便的多；同时由于它的工作的单一性，結构也簡单紧凑的多。因而設备量、体积都可較小。对于这种机器需注意如何提高它的利用率。

另一种分法是：

1. 并行机 数碼的运算及傳送是并行的。

2. 串行机 数碼的运算及傳送是串行的。

当然，按不同的特点还可有其他分法。如定点机与浮点机；同步机与异步机等等。这在以后还要詳細談到。

根据上面这种分法，苏联的 БЭСМ 机就是属于大型通用計算用真空管并行机。

关于机器的分类及每种机器的特点在本书第三部分：第七、八、九章还要詳細分析。

§1-3 碼制

各种碼制的区别在于其不同的"基"。

"基"是在数的每一位上所可能具有的不同符号的个数。例如对十进位数，在每一位上有 0、1、2、3、4、5、6、7、8、9 十种可能性。它的"基"为"十"，所以称为十进位制。而二进位制

的"基"为"二",在每一位上只有0、1两种可能性。

在机器上用得最多的是:二进位制、二——十进位制、八进位制、十六进位制等。最近还制成了三进位制的机器。

(一)二进位制 我們都知道,十进位制数是逢"十"进一,某一整数A可用:

$A=a_k\cdot 10^k+a_{k-1}\cdot 10^{k-1}+\cdots+a_1\cdot 10^1+a_0\cdot 10^0$ 表示。

如:

$$215=2\cdot 10^2+1\cdot 10^1+5\cdot 10^0=200+10+5$$

而二进位制数则是逢"二"进一。某一整数A可用:

$A=b_k\cdot 2^k+b_{k-1}\cdot 2^{k-1}+\cdots+b_1\cdot 2^1+b_0\cdot 2^0$ 表示。

而每一个 b_0、b_1、$b_2\cdots b_n$ 都只有0,1两种可能性。

如215用二进位制表示为:

$$\begin{aligned}11010111&=1\cdot 2^7+1\cdot 2^6+0.2^5+1\cdot 2^4+0.2^3+1\cdot 2^2+1\cdot 2^1+1\cdot 2^0=\\&=128+64+0+16+0+4+2+1=\\&=215\end{aligned}$$

对分数来講,也是同样的。

某一分数A用十进位制表示为:

$$A=a_1\cdot 10^{-1}+a_2\cdot 10^{-2}+\cdots$$

如:

$$0.6875=6\cdot 10^{-1}+8\cdot 10^{-2}+7\cdot 10^{-3}+5\cdot 10^{-4}$$

而用二进位制表示为:

$$A=b_1\cdot 2^{-1}+b_2\cdot 2^{-2}+\cdots$$

如0.6875用二进位制表示为:

$$\begin{aligned}0.1011&=1\cdot 2^{-1}+0.2^{-2}+1\cdot 2^{-3}+1\cdot 2^{-4}=\frac{1}{2}+0+\frac{1}{8}+\frac{1}{16}=\\&=0.5+0+0.125+0.0625=0.6875\end{aligned}$$

从"数論"的基本理論可以証明:任何的实数,不管是有理的或无理的都可以用二进位制、十进位制或其他进位制的幂級数表示出来。

在机器上采用二进位制較其他进位制的优点在于节省設备,这是因为目前計算机元件一般只具有二种稳定工作状态。另外二进位制对于加減法运算在机器上的实現也比較容易,其运算規則是很簡单的:

$$\begin{aligned}&0+0=0 \quad 1+0=1 \quad 0-0=0 \quad 1-1=0\\&0+1=1 \quad 1+1=10; \quad 1-0=1 \quad 10-1=1.\end{aligned}$$

(二)二——十进位制 由于目前最习慣用的是十进位制,因而当采用二进位制时就需要在輸入数据时进行从十进位制到二进位制的轉換。这需借助于专用設备或利用专門程序来完成,在使用时有时是不方便的。但目前还沒有具有十种稳定状态的計算机元件,要直接

采用十进位制是不可能的。因而就引出一种二——十进位制，即用四个二进位制数表示一位十进位制数。

由于四位二进位制数可以有从 0000 到 1111 的十六种状态，而十进位数只需十种状态，因而可能有各种組合，下面所举的例子是一种最直观的：

二——十进位制	十进位制
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

如 215 可用 0010、0001、0101 表示。

显然可以比較出，从十进位制轉換成二——十进制，要比轉換成二进位制簡单得多。一般所說的十进位制机器实际上是二——十进位制机器。二——十进位还用在二进位制机器中作輸入、輸出时用。輸入数据时首先經过打孔机等設备将十进位数变成二——十进位制数，再将它們送入机器，然后再經机器（用专用設备或程序）轉換成二进位制数。在輸出結果时，是将結果由二进位制轉換成二——十进位制后，再經輸出設备用十进位制打印出来。

采用二——十进位制的机器所用設备比采用二进位制的多 20% 左右。这是很明显的，例如 12 个二进位数在二——十进位制机器中只能表示从 0 到 999 个十进位数而用二进位制时則可表示从 0 到 4095 个数，这可从下表对应关系看出：

二——十进位制	十进位制
0000 0000 0000	0 到
1001 1001 1001	999 个数
二进位制	十进位制
000000000000	0 到
111111111111	4095 个数。

因此，为了表示同样大小的数，二——十进位制的位数要比二进位制的长些，亦即要化費更多的器材。

（三）八进位制　和二进位制的差别只在于它的“基”是 8，和二——十进位制同样的道理，实际上机器用的是二——八进位制。它在二进位制机器中在輸入指令时要用到。

由于八进位制的每一位只有 0、1、2、3、4、5、6、7 八种可能性，而三位二进位数也只具有

八种状态，所以在二——八进位制中，一位八进位数只需用三位二进位数表示。

如 215 用八进位制表示为：

$$[327]_8=3\cdot8^2+2\cdot8^1+7\cdot8^0=3\cdot64+2\cdot8+7=215$$

而用二——八进制表示则为：

011 010 111

有的机器在輸入、輸出及运算时还采用二——十六进位制。

§1-4 碼制的轉換

下面我們研究二进位制和十进位制；二进位制和二——八进位制之間的轉換問題。

(一)二进位制和十进位制間的轉換 下面我們分析各种碼制轉換的一般方法。

設要将 p 进位制的数 N

$$N=a_k\cdot p^k+a_{k-1}p^{k-1}+\cdots+a_1p^1+a_0$$

轉換成 q 进位制的数 N

$$N=a_k^*\cdot q^k+a_{k-1}^*\cdot q^{k-1}+\cdots+a_1^*q^1+a_0^*$$

看出：

$$\frac{N}{q}=a_k^*q^{k-1}+a_{k-1}^*\cdot q^{k-2}+\cdots+a_1^*+\frac{a_0^*}{q}$$

我們知道 p、q 都是整数，而 $a_0^*, a_1^*, \cdots a_k^*$ 也都是整数，但都小于 q 值，所以上式的右面除了 $\frac{a_0^*}{q}$ 項外都是整数。

这样，当将 p 进位制表示的数 N，用 q 进位制的"基" q 除了以后，其除不尽的余数即为用 q 进位制表示的 a_0^*；这样不断用 q 除 K 次后，所得的 K 个余数依次即为：

$$a_0^*、a_1^*、a_2^*\cdots a_{k-1}^*$$

最后，小于 q 值的商就是 a_k^* 了。

例如：要将十进位制的 25 轉換成二进位制数，則 $p=10, q=2$ 其計算方法如下：

```
25 |2
2  12 |2
 5 12  6 |2
 4  0  6  3 |2
 1     0  2  1
          1
 a0* a1* a2* a3* a4*
```

即为

11001

若 N 为分数，則用 p 进位制表示的分数

$$N=a_1p^{-1}+a_2p^{-2}+\cdots$$

而用 q 进位制表示的数为

$$N=a_1^*\cdot q^{-1}+a_2^*q^{-2}+\cdots$$

若將N乘以q得 $$N\cdot q=a_1^*+a_2^*q^{-1}+\cdots$$

可以看出，每次用q乘了以后其所得的整数部分卽为相应的$a_1^*, a_2^*\cdots$。这是很容易理解的，因为$a_1^*, a_2^*\cdots$等对于二进位制为"0"或"1"，对于十进位制則为0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 都只能是整数，而由于它們都小于q，因此上式中除了a_1^*外都是分数。

例如，要将十进位制数0.6875轉換成二进位制則：

$q=2$ 0.6875
$a_1^*=1$ 1.3750
$a_2^*=0$ 0.7500
$a_3^*=1$ 1.5000
$a_4^*=1$ 1.0000

則得用二进位制表示的0.1011。

每次所得的整数部分在下一次乘法中是不应再参加相乘的。

当然，不論是$N\times q$或$\frac{N}{q}$；在运算时N，q都应为用同样位制表示的数（在所举例子中，$N\times q$的25及2；$\frac{N}{q}$中的0.6875及2都是十进位制的数）。

（二）二—八进位制和二进位制間的轉換　这二种碼制之間的轉換是比較簡单的，这可以由下面的方法来証明。数N用八进位制表示为：

$$N=a_n8^n+a_{n-1}8^{n-1}+\cdots+a_18^1+a_08^0$$

用二进位制表示为：

$$N=b_k2^k+b_{k-1}2^{k-1}+\cdots+b_22^2+b_12^1+b_02^0$$

二式都用8除：

前式除不尽的余数为$\frac{a_0}{8}$

后式除不尽的余数为$\frac{b_2\cdot 2^2+b_12^1+b_02^0}{8}$

这两个余数是必需相同的，卽 $a_0=b_2\cdot 2^2+b_1\cdot 2^1+b_0\cdot 2^0$

同理得 $a_1=b_52^5+b_4\cdot 2^4+b_3\cdot 2^3$。

仍以215为例，前已說明215用

二——八进位制表示为：327亦为：011　010　111卽：

$a_0=111$　　$a_1=010$　　$a_2=011$

二进位制表示为：　　011010111

二种表示比較即可看出，只需将二——八进位制的各位合起来即得二进位制数；而二进位的所有位，从低位开始，按三位一組合即得二——八进位制表示的数。

(三)二——十进位制和二进位制間的轉換 首先分析一般机器采用的从二——十进位制到二进位制的轉換方法。我們知道，十进位制的小数：

$$0.\alpha_1\alpha_2\cdots\alpha_n(\alpha_i \text{ 有 } 0,1,2,\cdots 9 \text{ 十种可能性})$$

实际上为

$$\frac{\alpha_1}{10}+\frac{\alpha_2}{10^2}+\cdots+\frac{\alpha_n}{10^n}$$

例如 0.382 是

$$\frac{3}{10}+\frac{8}{10^2}+\frac{2}{10^3}$$

显然，如果把每一項都化成二进位数并求其和，即可得用二进位表示的該数。

在二——十进位制中：

$$0,\alpha_1'\alpha_2'\cdots\alpha_n',$$

每一个 α_i' 由四个二进位数表示，如上例为：

$$0,0011\quad 1000\quad 0010$$

但它的每一項实际值是：

$$\frac{3}{16},\ \frac{8}{16^2},\ \frac{2}{16^3}$$

所以在轉換时，需将用 4 个二进位数表示的 α_1'，$\alpha_2'\cdots$逐項用二进位数表示的$\frac{16}{10}$、$\frac{16^2}{10^2}\cdots$相乘并求其和即可得用二进位表示的該数，如上例用下式

$$A=(0011)\left(\frac{10000}{1010}\right)+(1000)\left(\frac{10000}{1010}\right)^2+(0010)\left(\frac{10000}{1010}\right)^3$$

即：

$$A=\alpha_1'\left(\frac{16}{10}\right)+\alpha_2'\left(\frac{16}{10}\right)^2+\cdots+\alpha_n^1\left(\frac{16}{10}\right)^n$$

即可将采用二——十进位制送入机器的数轉換成二进位制的数。这个輸入、乘、加过程是可以由机器自动完成的。

至于从二进位制到二——十进位制的轉換原理一般利用在(一)，所講述的方法；即将二进制表示的小数接連乘以用二进位制表示的"十"(即 1010)，其整数部分順序为二——十进位制中的 $\alpha_1', \alpha_2', \cdots\alpha_n'$。当然，在用机器实現时，还需要考虑到机器能不能表示大于"1"的整数，是否要作相应的变化。

由本节可以看出，数字机所采用的各种碼制和目前一般数字机中所用元件的"双稳定状态"性有很大的联系。

§1-5 定点制和浮点制

下面所要研究的是数在机器中是如何表示的。在分析时要考虑到二点：一是机器的位

数是有限的，这就需要考虑到机器所能表示的最大的数及最小的数；如何在位数已定的情况下扩大所能表示数的范围。其二是使用方便，如减少在确定比例系数时的困难等等。

在目前机器中一般采用二种方法：定点制和浮点制。

定点制 定点制指的是小数点的位置是固定不变的；虽然定点制能表示大于"1"的数，但通常只是用来表示分数；这在运算、使用上都要方便得多。

为了表示数的符号，通常用"0"表示〈+〉号，"1"表示〈−〉号。

如：

$$+0.6875 \text{ 为 } \boxed{0}\,1011$$
$$-0.6875 \text{ 为 } \boxed{1}\,\underbrace{1011}_{n\text{位}}$$

可以看出，若数碼的位数为 n，则所能表示的最小的数为 2^{-n}（如 $n=4$ 则为，$0{,}0001=2^{-4}=\frac{1}{16}$）；所能表示的最大的数为将所有数位都放上"1"，以 $n=4$ 为例：

$$.1111=1-0.0001=1-2^{-4}=1-\frac{1}{16}$$

即在机器中所能表示的数 N，其范围为：

$$2^{-n}\leqslant|N|\leqslant 1-2^{-n}$$

对于小于 2^{-n} 的数机器只能以"零"来处理，我們称之为"机器零"，显然他是随机器的位数长短而改变的。

从机器所能表示的最大的数也可看出，在使用、运算时必须使操作結果小于或等于所能表示的最大的数，否則就会出錯。例如：0.6875 加 0.6875 本应得 1.3750 但

$$\begin{array}{r} 0.1011 \\ +\ 0.1011 \\ \hline 1.0110 \end{array}$$

机器就会将它誤認为：−0.375。

因此在运算前要适当选擇比例系数，它的确定是一件相当复杂的工作。选擇大了，可能使运算結果大于机器所能表示的数；而选擇小了，又可能使运算結果或中間結果小于机器所能表示的最小的数，从而出現很大的誤差。

为了大大減輕选擇比例系数的工作，就引出了浮点制。

浮点制 从一般的十进位数的概念可知：

$$98.2 \text{ 可写成 } 10^{2}\cdot 0.982$$
$$0.00982 \text{ 可写成 } 10^{-2}\cdot 0.982$$

这样，任何的十进位有理数可用 $10^{p}\cdot m$ 来表示，我們称 p 为阶碼，它是整数。称 m 为数碼，它是小数。

轉換規律为：数碼右移 1 位和阶碼 +1 相当

数碼左移 1 位和阶碼 −1 相当

例如：98.2 既可用，$10^2\cdot0.982$ 表示，也可用 $10^4\cdot0.00982$ 表示（即阶碼加 2 和数碼右移二位相当）；

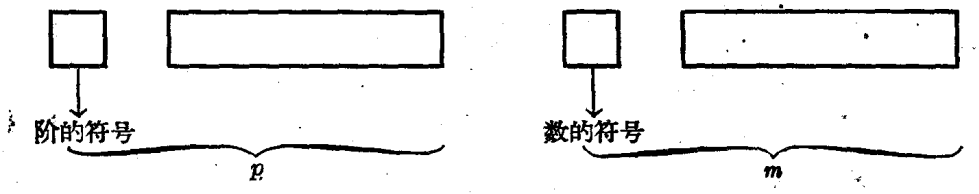
0.00982 既可用 $10^0\cdot0.00982$ 表示又可用 $10^{-2}\cdot0.982$ 表示（即数碼左移二位和阶碼减 2 相当）。

当然，用二进位制表示时其規律也是一样的，只需将"基"由 10 变为 2：

$$N=2^p\cdot m$$

$|m|<1$，m 可能为正数或負数，p 为整数，它也可能为正数或負数。

这样，在机器中的表示方法如下：



如：$-25=-2^{101}\cdot0.11001=32\times\left[-\left(\frac{1}{2}+\frac{1}{4}+0+0+\frac{1}{32}\right)\right]$ 在机器上表示为

为了便于实现各种算术运算及为了防止在运算过程中产生不应有的"机器零"的中間結果，一般称数碼的头一位有"1"的数为規格化了的数，每次参加算术运算的数应是規格化了的。而对运算結果也应将之規格化。規格化的方法可以利用上面所講的轉換規律。

如不規格化的数 $2^3\cdot0.00101$ 其轉換的方法为数碼左移二位，阶碼减 2，成为：

$$2^1\cdot0.10100$$

关于为什么对規格化了的数进行运算就可以大大避免"机器零"的出現，这可以从下面十进位数运算的例子看出：

如数碼部分的位数为 4，則不規格化的数 $10^0\cdot0.0004$ 和 $10^0\cdot0.2$ 相乘，本应得 $10^0\cdot0.00008$ 但机器的数碼部分只有四位，則其結果只能是"零"了；但如事先将所需运算的数轉換成規格化的数，則为 $10^{-3}\cdot0.4\times10^0\cdot0.2$ 得 $10^{-3}\cdot0.08$ 这样，就避免了出現"机器零"的中間結果，由此也可看出浮点制的优点。

設机器中数碼的位数为 μ（不包括符号位）阶碼的位数为 ρ（不包括阶的符号位）則机器所能表示的最大的数为：

$$2^{\overbrace{111\cdots1}^{\rho\text{位}}}\cdot\underbrace{0.111\cdots1}_{\mu\text{位}}=2^{2^\rho-1}\cdot(1-2^{-\mu})$$

所能表示的最小的規格化的数为：

$$2^{\overbrace{-111\cdots1}^{\rho\text{位}}}\cdot0.100\cdots0=2^{-(2^\rho-1)}\cdot2^{-1}=2^{-2^\rho}$$

即:

$$2^{-2^{\rho}} \leqslant |N| \leqslant 2^{2^{\rho}-1}(1-2^{-\mu})$$

下面我們来比較一下对同样位数的机器，采用定点制比浮点制在所能表示的数上差别多大。

設机器的总位数为 43 位，对定点机去掉一位符号位則 $n=42$ 位，数的范圍为:

$$2^{-42} \leqslant |N| \leqslant 1-2^{-42}$$

而浮点机去掉一位数碼符号位及一位阶碼符号位。設取 $\rho=6$ $\mu=35$。則数的范圍为:

$$2^{-2^6} \leqslant |N| \leqslant 2^{2^6-1}(1-2^{-35})$$

即为:

$$2^{-64} \leqslant |N| \leqslant 2^{-63}(1-2^{-35})$$

可見浮点制要比用定点制所能表示的数的范圍大得多。

虽然浮点制机器有数的表示范圍大，比例系数較易选择等优点，但浮点制机器要比定点制机器复杂，运算所需的操作多，因而运算时間就长。目前，比較多的机器还是定点制的。

不論是定点制还是浮点制都还是有不少缺点的；应該进一步探索新的数的表示方法，以簡化运算方法，机器結构，使机器的使用更方便。

§1-6 对数碼的运算

目前通常采用的数碼制有原碼，补碼，反碼三种。

原碼 原碼表示即为前面所講的方法，正数时符号位是"0"，負数时符号位为"1"。大部分机器在存儲时都是采用原碼存儲的。

若有一分数 x，則:

$$[x]_{原}=\begin{cases} x & \text{如果 } x \geqslant 0 \\ 1-x & \text{如果 } x<0 \end{cases}$$

如:

$$[0.10011]_{原}=0.10011$$

$$[-0.10011]_{原}=1-(-0.10011)=1.10011$$

可以看出，"0"有二种表示形式:

$$[+0]_{原}=[+0,00\cdots0]=0,000\cdots0$$

$$[-0]_{原}=[1-0,00\cdots0]=1,000\cdots0$$

大部分机器乘除时是用原碼操作的。

二进位制的乘法規則是很簡单的:

$$\begin{cases} 0\times0=0 \\ 0\times1=0 \\ 1\times0=0 \\ 1\times1=1 \end{cases}$$

$$0.1011\times0.1001=0.01100011$$

$$\begin{array}{r} 0.1011 \\ \times\ 0.1001 \\ \hline 1011 \\ 0\,000 \\ 0000 \\ +1011 \\ \hline 0.01100011 \end{array}$$

对于在数字机上的运算，减法操作一般都是以加"負数"来代替的。

如: $9-5=9+(-5)$

这样在机器上一般就只需有加法器，而不必有減法器。問題是采用什么办法来表示負数更便于执行有正、負数的加法运算(使符号位也参加运算)，并且能直接得出运算結果。用原碼操作是做不到这点的。

如: 0.1000 減 0.1011 本应得 1.0011

而用原碼操作为: $0.1000+1.1011$

$$\begin{array}{r} 0.1000 \\ +\ 1.1011 \\ \hline 0.0011 \end{array}$$

这样本应得 1.0011(−0.0011)而現在却得了錯誤的結果 0.0011(+0.0011)。

因此，根据机器运算的特点及要求，引出了补碼操作及反碼操作的方法，其目的在于在加法器上进行減法时能直接得到正确的运算結果。

补碼 若有一分数 x，則:

$$[x]_{\text{补}}=\begin{cases} x & \text{如} x\geqslant 0 \\ 2+x & \text{如} x<0 \end{cases}$$

$$[+0.10011]_{\text{补}}=0.10011$$

$$[-0.10011]_{\text{补}}=2+(-0.10011)=1.01101$$

可以看出，对于 $x\geqslant 0$ 用原碼或用补碼表示都是一样的。

"变补"的方法在机器上是这样实现的;

如: $x=-0.x_1x_2\cdots x_n$,

$x_1x_2\cdots x_n$ 为二进位数，可能是 0 或 1; 則:

$$[x]_{\text{补}}=1,\bar{x}_1\bar{x}_2\cdots\bar{x}_n+0,\underbrace{00\cdots01}_{n-1\text{個}0}$$

$$=1,\bar{x}_1\bar{x}_2\cdots\bar{x}_n+2^{-n}$$

式中 $\bar{x}_i=0$ 当 $x_i=1$; $\bar{x}_i=1$ 当 $x_i=0$。

現将 −0.10011 变补为例

$$[x]_{\text{补}}=1.01100+0.00001=1,01101$$

对于用补碼表示的"0"只有一种表示形式:

因为 $[+0]_{\text{补}}=0.000\cdots0$

$$[-0]_{补}=1.11\cdots1+0.00\cdots1$$
$$=0.000\cdots0 \text{ 是相同的。}$$

可以証明出;

$$[x]_{补}+[y]_{补}=[x+y]_{补}$$

对于 x, y 为正、負数都是正确的。

当 x, y 都是正数时是很明显的;

如: 0.1001+0.0010 应得 0.1011

用补碼运算:

$$[0.1001]_{补}=0.1001$$
$$[0.0010]_{补}=0.0010$$
$$\text{相加得} \quad 0.1011=[0.1011]_{补}$$

当 x, y 其中之一是負数时, 还以 0.1000 減 0.1011 为例, 本应得 −0,0011, 但用补碼运算:

$$[0.1000]_{补}= \quad 0.1000$$
$$[-0.1011]_{补}= \quad 1.0101$$
$$\text{相加得} \quad 1.1101$$

而应得結果 −0.0011 的补碼表示为:

$$[-0.0011]_{补}=1.1101$$

亦卽:

$$[0.1000]_{补}+[-0.1011]_{补}=[-0.0011]_{补}$$

也就是:

$$[x]_{补}+[y]_{补}=[x+y]_{补}$$

这样, 当負数用补碼表示并使符号位参加运算, 則能直接在加法器上得到应有的正确結果。

由于乘、除法一般是原碼操作的, 数碼在存儲器一般也是以原碼存儲, 所以有时需将用补碼表示的負数重新变回原碼表示。变回原碼的方法和从原碼变到补碼是相同的, 卽把数碼部分每一位由"0"变"1", 由"1"变"0"且在最末位 +1(卽 $+2^{-n}$)。

如上例, 要将 1.1101 变成原碼則为:

$$1.0010 \quad [1.\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4]$$
$$+) \quad 0.0001 \quad [2^{-n}]$$
$$1.0011=[-0.0011]_{原}$$

如果是两个負数相加, 例如: (−0.1001)+(−0.0100)应得 −0.1101

而
$$[-0.1001]_{补}=1.0111$$
$$[-0.0100]_{补}=1.1100$$
$$\text{相加得} \quad 1.0011$$

而应得結果 −0.1101 的补碼表示为: