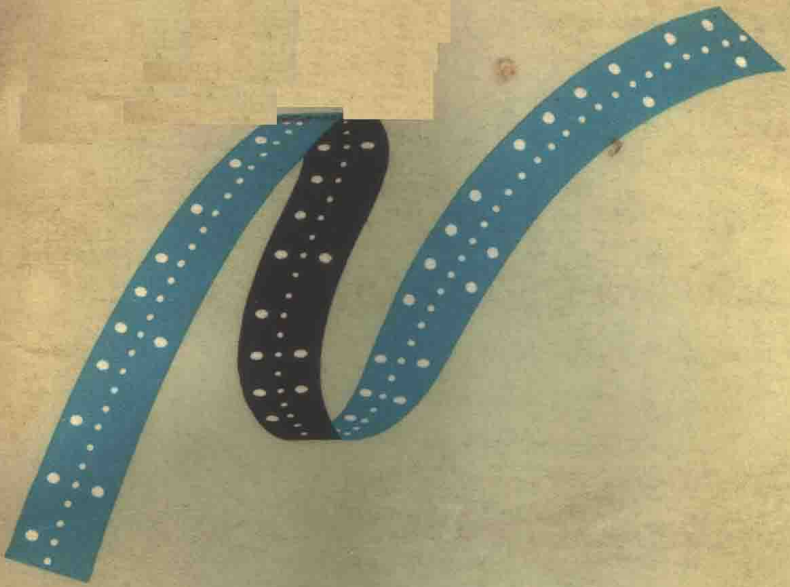

張綺霞　朱淑霞　曹东启 编

实用程序设计

科学出版社

实用程序設計

张綺霞　朱淑霞　曹东启 編

科学出版社

1963

内 容 簡 介

本书以 104 机为背景，概括叙述了有关程序設計的基本知識以及某些算題的程序設計的实际經驗.

全书共分十二章. 前二章介紹了电子数字計算机的基本知識和 104 机的結构以及指令系統；第三章至第五章叙述了程序設計的基本方法和編制程序的整个过程；第六章至第九章講述了在計算較大的、复杂的問題时，程序設計方面可能遇到的問題，并提出了处理这些問題的意見；第十、十一章講述了如何检查程序的錯誤及如何使用机器来完成算題工作；最后一章闡述了标准程序和实用問題程序标准化問題.

此书可供从事实际工作的計算工作者学习参考之用，亦可作高等学校計算专业的教学及在其他类型計算机上工作人員的参考.

实用程序設計

张綺霞　朱淑霞　曹东启 編

*

科学出版社出版 (北京朝阳門大街 117 号)

北京市书刊出版业营业許可証出字第 061 号

中国科学院印刷厂印刷　新华书店总經售

*

1963 年 12 月第 一 版　　书号：2798　字数：225,000

1963 年 12 月第一次印刷　　开本：850×1168　1/32

(京) 0001—4,000　　印张：8 5/8　插頁：1

定价：1.50 元

序

本书內容，一方面包括程序設計的基本知識，一方面包括某些算題的程序設計的实际經驗．书中叙述是以 104 机为背景，因此，对于欲在 104 机上从事实际工作的計算工作者，可作学习参考之用．此外，亦可作为高等学校計算专业及在其他类型計算机上工作人員的参考．

全书共分十二章．第一章先对电子数字計算机的基本知識作了概括叙述．第二章对 104 机的基本結构和指令系統作了具体說明．第三章到第五章講述程序設計的基本方法和編制程序的整个过程．初学的人通过这五章的学习，对于程序設計将有一个初步了解．第六章到第九章闡述了在計算比較大的、复杂的計算問題时，程序設計方面可能遇到的某些問題，并对这些問題提出一些处理意見，它将使讀者在程序規划、程序組織等方面，有所启发．第十、十一章講述如何检查程序的錯誤以及如何使用机器来完成算題工作．最后一章，叙述标准程序和实用問題程序标准化問題，可供在实际算題时的参考．

本书的編写，一方面根据中国科学院計算技术研究所 1961 年以前的某些关于程序設計的工作总結，一方面根据了編者在实际工作中的浅薄体会．在編写过程中特别得到了徐献瑜先生、馮康先生、张克明先生的热情指导和帮助，謹此表示感謝．再者关姍先生、董韞美、鮑信炯、周明榘、王亨慈、卢慧琼、申宏一、李开德、张耀科、鍾明璉以及其他一些同志提供了很多宝貴意見和宝貴的实际經驗，一并表示感謝．

由于編者在机器上工作的时間不长，水平有限，缺点錯誤在所难免，希望讀者多多提出批評指正．

編　者

一九六三年三月于中国科学院計算技术研究所

第一章　通用电子数字計算机

§1.1. 基本結构和工作原理

現代的通用电子数字計算机是采用电子管、晶体管、磁元件、电阻、电容以及其他无綫电技术零件所制成的一个复杂的、能高速度完成运算的电子自动装置. 它的重要組成部分一般有存儲器、运算器、控制器和輸入輸出器(如图 1.1).

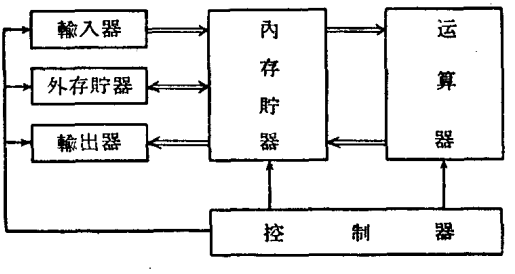

图 1.1

存儲器是存儲代碼的裝置. 我們把数、指令或一組数字編碼統称为代碼.

存儲器好比一个**旅館**，有成千上万个**房間**，每个房間有固定的編号. 我們所謂的存儲"单元"相当于"单个房間"，"单元地址"相当于"房間編号"，而"代碼"就相当于"房間"中来往的"旅客". 存儲器单元的数量(簡称存儲量)可有数百、数千或者更大.

存儲器具有接受、保存和发送代碼的功能. 存在存儲器中的代碼一般是不会改变的，从任何单元中讀出代碼后，該代碼仍旧存留在单元中，但当送入新的代碼时，就会把单元中原存的代碼冲掉而代之以新的代碼.

在較多的計算机中，存儲器分內存儲器和外存儲器. 內存儲器与运算器有直接的联系，可以发送代碼到运算器参与运算，并能从运算器接受运算結果. 外存儲与运算器一般沒有直接的联系，但有較大的存儲量，并且可以和內存儲器成批地进行代碼的轉移.

此为试读，需要完整PDF请访问：www.ertongbook.com

对于具有大量数据或指令的計算問題，在內存儲器不敷使用时，常可利用外存儲器来存放計算中暫时不用的数或指令.

运算器是对送入运算器中的代碼进行各种运算的装置，它由电子管或晶体管等元件所組成的計算綫路构成.

控制器是用来实現机器各部分間的联系，它根据計算程序指揮机器的各部分进行工作，保証了計算过程的自动进行. 控制器是由电子管或晶体管所組成的电子門綫路构成.

輸入器是用来輸入原始数据和計算程序，可用穿孔卡片、穿孔紙带或磁带进行輸入. **輸出器**是用来輸出工作結果(数或程序)，可用印刷、照相、卡片穿孔、紙带穿孔或記入磁带等方式进行輸出.

輸入器、輸出器以及外存儲器一般統称为外部設备，这些設备因与机械动作有关，因而工作速度較低. 机器的其他部件因用电子綫路构成，所以可以达到很高的速度.全机各部分用传送代碼的**路綫和控制路綫**进行互相联系.

在机器上解題时，必須把解題的算法化成机器所能完成的基本运算，并使用机器所能識别的一串数字編碼把所要进行的运算表达出来. 这些指示机器完成一定操作的一串数字編碼就称为指令. 指令应指明：进行什么性質的运算，参与运算的数从哪里选取以及运算結果送到哪里去. 完成解題所需的一系列指令称为計算程序.

机器完成解題的过程，总的說起来大致是：首先将程序和所需数据輸入存儲器，然后根据程序的安排，控制器控制机器自动地执行程序所指定的一系列运算，輸出所要的运算結果，直到运算完毕自动停机.

机器完成一条运算指令的工作步驟，大致如下：

1. 指令由內存儲器送入控制器中存放指令的部件.

2. 控制器对指令进行分析，发出从哪里取数和作哪种运算的信号.

3. 运算器接受信号后，对数进行运算.

4. 控制器发出信号，控制机器把运算結果送到指令所指定的

地方.

5. 机器准备轉入下一条指令,并准备执行这条指令.

一架机器除包含运算指令外，为了保証机器的自动工作还需有其他类型的指令,这些指令的工作步驟与上述可能不同.

根据不同机器的結构,指令列的执行順序有二种方式:一种是在正被执行的指令中指出下一指令的地址(強制执行式);另一种，在通常情况下是按指令的存放順序执行，但当遇到某些特殊指令时,也可变更指令的执行順序(自然执行式).

§1.2. 数的表示

1. 計数系統

在日常生活中我們习慣采用十进位計数制，在这个計数系統中，有十个不同的数字：0, 1, 2, ···, 9, 任何数 A 均用一串数字来表示，

$$A = a_n a_{n-1} \cdots a_1 a_0 . a_{-1} a_{-2} \cdots a_{-m},$$

其中 $a_i (i = n, n-1, \cdots, -m)$ 为十个数字中的一个,且根据其所在位置的不同，具有不同的含意，a_0 表示个位，a_1 表示十位，$\cdots$，a_{-1} 表示十分之一位，a_{-2} 表示百分之一位，$\cdots$，即数 A 可表为

$$\begin{aligned} A = {} & a_n \cdot (10)^n + a_{n-1} \cdot (10)^{n-1} + \cdots + a_1 \cdot (10) + \\ & + a_0 + a_{-1} \cdot (10)^{-1} + a_{-2} \cdot (10)^{-2} + \cdots + \\ & + a_{-m} \cdot (10)^{-m} = \sum_{i=-m}^{n} a_i \cdot (10)^i, \end{aligned}$$

括弧中的 10 即为計数制的基数.

十进位制并不是唯一的計数制,即計数制的基数不一定是 10，而可以是其他的任意正整数 R. 用 R 作基数的进位制(簡称 R 进位制),对于任意数 A 可表为

$$A = \sum_{i=-m}^{n} a_i R^i \quad (a_i \text{ 为 } 0, 1, \cdots R-1 \text{ 中的一个数字}),$$

并亦可写成

$$A = a_n a_{n-1} \cdots a_1 a_0 . a_{-1} a_{-2} \cdots a_{-m},$$

a_0 表示个位，a_1 表示 R 位，a_2 表示 R^2 位，$\cdots$，a_{-1} 表示 R 分之一位，a_2 表示 R^2 分之一位，$\cdots$.

为区分数的不同进位制表示，我們用符号 A_R 表示数 A 的 R 进位制表示.

2. 二、四、八、十六进位制；二—十进位制

最簡单的进位制是二进位制，这时基数 $R = 2$，并且对任何数只采用二个数字 0 和 1 来表示. 例如数 41_{10} 的二进位制表示为

$$101001_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 41_{10}.$$

目前有較多的机器是采用二进制計数的，这是由于用二进制有如下的一些优点：

(1) 可以用任何只具有二个不同稳定状态的元件来記数的每一位，将一种稳定状态表示数字"0"；另一种表示数字"1". 制造具有两个稳定状态的元件要比制造多稳定状态的元件容易得多.

(2) 二进制的算术运算比較簡单，例如

加法：$0 + 0 = 0$，$1 + 0 = 0 + 1 = 1$，$1 + 1 = 10$；

乘法：$0 \times 0 = 0$　$1 \times 0 = 0 \times 1 = 0$，$1 \times 1 = 1$.

运算的簡便，可以使得机器的結构比較簡单.

(3) 可以节省存儲設备. 在說明原因之前，我們先引入"数字-位"的概念. 所謂"数字-位"卽是为表示在一定范围內的任意数所需的元件数目和元件稳定状态数目的乘积. 例如用十进制表示 0 到 999 的数需要三位，每位需十个数字，这时的"数字-位"为 $3 \times 10 = 30$. 用二进制表示上面的数需要 10 位，每位需二个数字，"数字-位"为 $10 \times 2 = 20$.

我們来作一般性的研究. 設要表示从 0 到 N 的数，进位制的基为 R，需要的位数为 n，求使"数字-位" $x = Rn$ 取最小值的 R.

显然我們有：

$$N = R^n - 1, \quad R^n = N + 1 = \bar{N},$$

$$\log \bar{N} = n \log R, \quad x = R \frac{\log \bar{N}}{\log R},$$

为了求 x 的最小值，使微商 $\frac{dx}{dR}$ 为零，得

$$\frac{dx}{dR} = \frac{\log \bar{N}}{\log R} - \frac{\log \bar{N}}{\log^2 R} = 0.$$

故 $\log R = 1$ 即 $R = e(e \approx 2.72)$. 与 e 接近的整数为 2 及 3.

三进制的"数字-位"最小，但二进制的其他一些优点是三进制所不及的，故二进制还是較三进制用得普遍些.

固然采用二进制具有一系列的优点，但也有它不便之处，这主要是由于人們习慣于十进制，因此增加了将原始数据由十进制轉換成二进制；計算結果又需从二进制轉換成十进制的手續. 另外，用二进制表示数的数字較长，书写观看均有不便.

为克服书写不便的缺点，在記写时根据

$$2^2 = 4, \quad 2^3 = 8, \quad 2^4 = 16$$

的特点，把二进位数以两位、三位、四位为一小节，相应的就化为四进位、八进位、十六进位制的数. 以这几种进位制来表示数，显然数位是縮短了，且数的本身仍保有二进制的特点.

二进位制 $R = 2$, $a_i = 0$ 或 1

四进位制 $R = 4$, $a_i = 0, 1, 2, 3$. 每位四进制数以两位二进制数表示时为

0	1	2	3
↓	↓	↓	↓
00	01	10	11

八进位制 $R = 8$, $a_i = 0, 1, 2, \cdots, 7$. 每位数以三位二进位数表示时为

0	1	2	3	4	5	6	7
↓	↓	↓	↓	↓	↓	↓	↓
000	001	010	011	100	101	110	111

十六进位制 $R = 16$, $a_i = 0, 1, 2, \cdots, \bar{0}, \bar{1}, \bar{2}, \bar{3}, \bar{4}, \bar{5}$, 其中数字符号 $\bar{0}, \bar{1}, \bar{2}, \bar{3}, \bar{4}, \bar{5}$ 表示相当于十进制数的 10, 11, 12, 13, 14, 15. 这时每位数需用四位二进制数来表示：

0	1	2	3	4	5	6	7
↓	↓	↓	↓	↓	↓	↓	↓
0000	0001	0010	0011	0100	0101	0110	0111

8	9	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$	$\bar{5}$
↓	↓	↓	↓	↓	↓	↓	↓
1000	1001	1010	1011	1100	1101	1110	1111

依照上述二进制与四、八、十六进制数字間的对应关系，要把二进制数化为四、八、十六进制数是十分簡便的．例如

$\underline{11111000_2} = 3320_4$，$\underline{11111000_2} = 370_8$，$\underline{11111000_2} = \bar{5}8_{16}$.

反过来，知道了四、八、十六进制数后，要求二进制数，只要把每位四、八、十六进制数以两位、三位、四位二进制数展开即得．如

$123_4 = \underline{11011_2}$，$257_8 = \underline{10101111_2}$，$13\bar{4}_{16} = \underline{100111110_2}$.

二—十进位制 用几位二进位数表示一位十进制数的計数制称为二—十进位制．如十个十进制数字采用下述表示时

0	1	2	3	4	5
↓	↓	↓	↓	↓	↓
0000	0001	0010	0011	0100	0101

6	7	8	9
↓	↓	↓	↓
0110	0111	1000	1001

数 157_{10} 的二—十进制表示为

$$0001\ 0101\ 0111_{(2-10)}$$

二—十进位制用于某些按十进制計数的机器以及用于在二进制的机器上作为二进制和十进制轉換中的中間进位制．在解題时，数制的轉換通常是利用程序或特殊設备来实现的，在进行轉換时，將十进制的原始数据表成二—十进制輸入机器；而二进制的計算結果又在机器上轉換成二—十进制表示的十进制数，然后再行輸出．

3. 数的定点表示和浮点表示；規格化的数

例如十进制的 432.56 这样一个数与下面几种写法：

10×43.256　$10^3 \times 0.43256$　$10^4 \times 0.043256$

是完全相等的.

在机器中，数一般是以上述后面两种形式来表示的，即表成 $\pm 10^p \cdot d (0 \leqslant d < 1)$. 对于 R 进制的数，其通式可表达如下：

$$A = \pm R^p \sum_{i=1}^{n} a_i R^{-i} = \pm R^p \times 0.a_1 a_2 \cdots a_n$$

$$\left(0 \leqslant \sum_{i=1}^{n} a_i R^{-i} < 1\right),$$

其中整指数 p 决定数的小数点的位置，称为数的阶，它可正可负亦可为零. $a_1 a_2 \cdots a_n$ 称为尾数，其中 $a_i (i = 1, 2, \cdots, n)$ 为 0, 1, $\cdots$, R 中的某个数字. n 是尾数的位数. 例如十进制的

$$\pm 432.56 = \pm 10^3 (4 \cdot 10^{-1} + 3 \cdot 10^{-2} + 2 \cdot 10^{-3} + $$
$$+ 5 \cdot 10^{-4} + 6 \cdot 10^{-5}) = \pm 10^3 \cdot 0.43256,$$
$$\pm 0.0045 = \pm 10^{-2} (4 \cdot 10^{-1} + 5 \cdot 10^{-2}) = \pm 10^{-2} \cdot 0.45.$$

阶 p 等于常数的計算机，称为**定点**計算机. 在这种机器中，数被表示成一串数字

$$a_0 a_1 a_2 \cdots a_{n-1} a_n,$$

其中 a_0 表示数的符号，通常正号用"0"表示，負号用"1"表示. $a_1 a_2 \cdots a_n$ 是数的尾数. 这时小数点的位置是固定的. 一般的定点計算机取阶 $p = 0$，即小数点固定在 a_1 之前. 这时机器所能表示的数在 $+1$ 和 -1 之間. 这种机器的結构比較簡单，但編制程序比較費事.

阶 p 是可变的計算机称为**浮点**計算机. 这时对于每一个数，除給出尾数外，还需給出数的阶. 一般数 A 在存儲单元中的形式为

$$p_0 p_1 p_2 \cdots p_r \; a_0 a_1 a_2 \cdots a_n,$$

p_0, a_0 各表示数的阶和数本身的符号. $p_1 p_2 \cdots p_r$ 表示数的阶. $a_1 a_2 \cdots a_n$ 是数的尾数，而小数点在 a_1 之前.

浮点計算机表示数的范围較大，程序設計較定点計算机省事，但机器結构要比定点机复杂得多. 浮点机器的四則运算按下列規則进行.

此为试读，需要完整PDF请访问：www.ertongbook.com

加法：$z = z_1 + z_2$ 数先对阶(使二数的阶相等)，然后尾数相加.

減法：$z = z_1 - z_2$ 規則同上.

乘法：$z = z_1 \times z_2$ 两数的阶相加，尾数相乘.

除法：$z = z_1 : z_2$ 两数的阶相減，尾数相除.

浮点机器中，数的尾数的最高位 $a_1 \neq 0$ 时，称为**已規格化的数**，例如 $10^3 \times 0.43256$. 否則称为未規格化的数，如 $10^4 \times 0.043256$.

采用二进位制計数时，已規格化的数表为

$$A = \pm 2^p \cdot d, \quad \text{其中 } \frac{1}{2} \leqslant d < 1,$$

4. 数的原碼、补碼和反碼表示

在計算机中，数和阶都要表成一組数字

$$x = x_0 x_1 x_2 \cdots x_n,$$

其中 x_0 表示符号，而 $x_1 x_2 \cdots x_n$ 是描繪 x 的一串数字.

一般說来，数有三种表示，即原碼、补碼及反碼，分别記作 $(x)_{原}$, $(x)_{补}$, $(x)_{反}$.

我們先假定 x 是二进制小数，即小数点固定在 x_1 之前.

对于正数 $x = 0.x_1 x_2 \cdots x_n$，我們規定

$$x = (x)_{原} = (x)_{补} = (x)_{反} = 0.x_1 x_2 \cdots x_n.$$

下面我們叙述 x 为零或負数时的情况.

原碼 数本身为絕对值，其正負号由符号位表示. 若

$$x = -0.x_1 x_2 \cdots x_n = -|x|,$$

則

$$(x)_{原} = 1.x_1 x_2 \cdots x_n = 1 + |x| = 1 - x.$$

例. $x = -0.1011$，則 $(x)_{原} = 1.1011$.

用这种表示法，零有两种可能的形式，分别称为正零及負零.

$$(+0)_{原} = 0.00 \cdots 0, \quad (-0)_{原} = 1.00 \cdots 0.$$

x 用原碼表示时，其变化范围为 $[-(1 - 2^{-n}), (1 - 2^{-n})]$.

补碼 負数 $x = -0.x_1 x_2 \cdots x_n$ 的补碼表示，具有形状

$$(x)_{补} = 1.x_1' x_2' \cdots x_n',$$

式中 $0.x'_1x'_2\cdots x'_n = 1 - |x| = 1 + x$，故当 $x < 0$ 时

$$(x)_{补} = 2 - |x| = 2 + x.$$

例. $x = -0.1011$，则 $(x)_{补} = 1.0101$.

用这种表示法，零只有一种形式，即

$$(0)_{补} = 0.00\cdots 0.$$

x 用补码表示时，其变化范围为 $[-1, 1 - 2^{-n}]$.

綜合正数、零及負数的补码表示，可統一写成

$$(x)_{补} \equiv x \pmod 2,$$

由此得出

$$(x + y)_{补} \equiv x + y \equiv (x)_{补} + (y)_{补} \pmod 2.$$

反码 負数 $x = -0.x_1x_2\cdots x_n$ 的反码表示，具有形状

$$(x)_{反} = 1.\bar{x}_1\bar{x}_2\cdots\bar{x}_n,$$

式中若 $x_K = 1$，則 $\bar{x}_K = 0$；若 $x_K = 0$，則 $\bar{x}_K = 1$ $(K = 1, 2, \cdots, n)$. 故

$$(x)_{反} + |x| = 1.11\cdots 1 = 2 - 2^{-n}.$$

所以当 $x < 0$ 时

$$(x)_{反} = 2 - 2^{-n} - |x| = x + (2 - 2^{-n}).$$

例. $x = -0.1011$，則 $(x)_{反} = 1.0100$.

容易看出，当 $x < 0$ 时，$(x)_{补}$ 与 $(x)_{反}$ 仅在最低位相差 1，在 $(x)_{反}$ 的最低位加上 1 时，即为 $(x)_{补}$.

用这种表示法，零也有两种形式

$(+0)_{反} = 0.00\cdots 0$，　$(-0)_{反} = 1.11\cdots 1 = 2 - 2^{-n}$.

x 用反码表示时，其变化范围为 $[-(1 - 2^{-n}), (1 - 2^{-n})]$.

綜合正数、零及負数的反码表示，可統一写为

$$(x)_{反} \equiv x \pmod{(2 - 2^{-n})},$$

由此推出

$$(x + y)_{反} \equiv x + y \equiv (x)_{反} + (y)_{反} \pmod{(2 - 2^{-n})}.$$

对于 x 非小数的情形，亦可作类似的討論. 設 x 的小数点固定在第 K 位之前，即

$$x = x_0x_1\cdots x_{K-1}.x_K\cdots x_n.$$

此时，只需在上面的叙述中，将 mod 2 和 mod $(2-2^{-n})$ 用 mod 2^k 和 mod (2^k-2^{k-1-n}) 代替即可．特别当 $k=n+1$ 时，也就是小数点固定在所有的数字之后，此时 x 为整数．当 $x<0$ 时，有

$$(x)_{原}=2^n+|x|,$$
$$(x)_{补}=2^{n+1}-|x|,$$
$$(x)_{反}=(2^{n+1}-1)-|x|.$$

在十进制中，亦可作类似的討論．例如

$$x_{10}=-0.123,$$

則

$$(x)_{原}=1.123,$$
$$(x)_{补}=1.877\ (因\ 0.877=1-0.123),$$
$$(x)_{反}=1.876,$$

其中 8，7，6 分别是 1，2，3 对 9 的补数．

数用原碼表示时，作乘除法方便，因乘除时符号与数分别运算，但作加減法不方便，因机器須事先检查两数的符号，才能决定作加法还是作減法．

数用补碼或反碼表示时，作加減法方便，但作乘除法不方便．

§1.3. 指令和指令系統

1. 指　令

当我們用計算机解題时，需要将选定的算法編成程序，程序由一系列的指令組成，每一个指令能使机器执行确定的操作．为了把指令輸入机器，須将指令表成数碼形式．每条指令包含一个操作碼和若干个地址碼．操作碼指示所規定的操作，地址碼指示操作对象及操作結果的存放位置或提供其他信息．

指令的形式一般可表达如下：

$$\theta A_1A_2\cdots A_K,$$

其中 θ 为操作碼，$A_1A_2\cdots A_K$ 为地址碼．如果一架机器的指令含有 K 个地址，則称为 K 地址計算机．常見的有以下几种．

三地址計算机，其指令形式是

$$\theta \quad A_1 \quad A_2 \quad A_3$$

通常 A_1, A_2 是二个运算对象所在的单元地址，A_3 是存放运算結果的单元地址. 三地址的指令完全符合于算术运算的邏輯构造，每个算术运算通常有三个数参加，两个作为参与运算的数，第三个作为运算結果的数.

二地址計算机. 指令中两个地址的用法可有多种形式，例如可在两个地址中指明存放运算对象的单元地址，且其中的一个地址又是存放結果的单元地址. 或者，在两个地址中，一个指明存运算对象的单元地址，一个指明存結果的单元地址.

一地址計算机的指令形式为

$$\theta \quad A$$

地址 A 和各种操作碼 θ 的使用可有各种各样的組合形式. 例如一些操作碼 θ 用地址 A 所存的数进行运算，另一些則向地址 A 送入結果等等.

在原則上，每个 k 地址运算都可分解为 k 个一地址运算. 例如对三地址运算可分解为：

$\theta_1 A_1$——把 A_1 中的数送入运算器寄存起来；

$\theta_2 A_2$——把运算器中寄存的数和地址 A_2 中的数进行操作碼为 θ_2 的运算；

$\theta_3 A_3$——将結果送入单元 A_3.

但不能由此得出結論，說一地址計算机的程序比三地址的程序长三倍，因为一般一地址計算机的运算器都有一个或几个寄存器，在連續运算时可用它来保存中間結果，所以程序只比三地址計算机的程序稍长一些.

四地址計算机，指令形式为

$$\theta \quad A_1 \quad A_2 \quad A_3 \quad A_4$$

一般是強制执行式的計算机. $\theta\ A_1\ A_2\ A_3$ 的作用与三地址指令类似，而第四地址是指明下一次所要执行的指令的地址.

这种強制执行式的指令也有用三地址或二地址的，此时除用

一个地址說明下一次执行的指令地址外，其他部分分别与二地址或一地址指令相类似.

由于指令的操作碼和地址碼都是用数碼来表示，因此在存儲单元中的指令也具有数的形式，可以参与运算，改变指令的形式.

2. 指令系統

机器所能执行的基本操作的总和称为机器的指令系統. 各种机器的指令系統很不一致，但为保証机器自动完成計算工作，对于通用的計算机，它們都具有以下三类指令：

(1) 算术运算指令，如加、減、乘、除等.

(2) 邏輯运算指令，如邏輯乘法、邏輯加法等.

(3) 控制指令，如改变指令执行順序、改变机器的工作状态、停机等.

显然，机器的指令系統愈完备，把解題算法写成計算程序就愈容易实現，但如果系統中包含的操作过多，則会使机器的結构复杂化，因此在設計机器的指令系統时，总是兼顧到便利程序設計和簡化机器結构这两方面的.

附录. 人工換算数制的方法

在二进制机器上解題时，虽然数制的轉換可由机器来完成，但在实际工作中亦常有需要由人工換算少量的数据，因此熟悉一种轉換数制的方法是很必要的.

前面我們已介紹过二进制数与四、八、十六进制数之間的換算方法. 在机器上的二进位数，人們常用八进制或十六进制数来記录. 因此解决了十进制与八进制，或十进制与十六进制之間的換算，也就解决了十进制数与二进制数間的換算. 由于本书以后各章的介紹是以 104 机器为背景的，而 104 机器的二进制数是以十六进制数来記录的，因此下面我們就只討論十进制数与十六进制数之間的換算. 至于其他不同进位制間的轉換，讀者可用类似的法則得到.

1. 直接計算的方法

$x_{10} \Longrightarrow x_{16}$

若 x_{10} 为整数，因

$$x_{16} = x_{10} = a_r 16^r + a_{r-1}16^{r-1} + \cdots + a_1 16^1 + a_0 16^0 =$$
$$= (a_r 16^{r-1} + a_{r-1}16^{r-2} + \cdots + a_1)16 + a_0,$$

故 x_{16} 的个位数 a_0 为 x_{10} 除以 16 后的整商余数. 再繼續用 16 除每次所得的商，我們就將得到 16^1 位数 a_1，16^2 位数 a_2 等等. 我們可写出換算規則如下：

以 16 除 x_{10}，設商为 q_1，余数为 $a_0 = x_{16}$ 的个位数；
以 16 除 q_1，設商为 q_2，余数为 $a_1 = x_{16}$ 的 16^1 位数；
以 16 除 q_2，設商为 q_3，余数为 $a_2 = x_{16}$ 的 16^2 位数，
.....

当商为零时，余数 a_r 即为最高位的十六进位数.

例 1. $1024_{10} = (?)_{16}$.

$$\begin{array}{r|l|l} 16 & 1024 & 0 \\ 16 & 64 & 0 \\ 16 & 4 & 4 \\ & 0 & \end{array}$$

以 16 除 1024，商为 64，余数为 0；
以 16 除 64，商为 4，余数为 0；
以 16 除 4，商为 0，余数为 4.

故

$$1024_{10} = 400_{16}.$$

例 2. $2047_{10} = (?)_{16}$.

$$\begin{array}{r|l|l} 16 & 2047 & 15 \\ 16 & 127 & 15 \\ 16 & 7 & 7 \\ & 0 & \end{array}$$

以 16 除 2047，商为 127，余数为 $\bar{5}$；
以 16 除 127，商为 7，余数为 $\bar{5}$；
以 16 除 7，商为 0，余数为 7.

故

$$2047_{10} = 7\bar{5}\bar{5}_{16}.$$

若 x_{10} 为小数，因

$$x_{16} = x_{10} = a_1 16^{-1} + a_2 16^{-2} + \cdots + a_r 16^{-r}$$
$$16x_{10} = a_1 + (a_2 + a_3 16^{-1} + \cdots + a_r 16^{-r+2})16^{-1},$$

故 a_1 为 $16x_{10}$ 的整数部分，即 a_1 为 x_{16} 的第一位小数. 再繼續用 16 乘各次剩下的小数部分，就將陆續求得 x_{16} 的第二位小数、第三位小数等. 因此得

此为试读，需要完整PDF请访问：www.ertongbook.com