

5087  
08407: 1  
1

802518

# 计算机软件基础

(上)

计算机应用软件人员水平考试应试指导联合会

教材编写组

上海微电脑厂



087  
8407: 1  
1

上海微电脑厂

## 前 言

对计算机应用人员实行统一的水平考试制度，在国外已有一些国家在实行。这种考试制度的实行将有利于在统一标准下显示计算机应用人员的实际能力和水平，保证各级各类培训的教育质量，而且有利于广大的科技人员学习和掌握计算机技术，促进计算机应用的普及和发展。

上海市政府1985年将举行“计算机应用软件人员水平考试”。为了帮助有志者提高应试能力，为了使广大计算机应用人员进一步系统地掌握计算机知识，在上海计算技术研究所青年信息应用技术开发部、华东计算技术研究所青年电脑技术开发部，复旦大学计算机科学系研究生咨询中心、上海工业大学研究生学术交流协会、上海第二工业大学质量管理系(筹)、上海~~电信局~~等单位的努力下，将组织举办计算机应用人员水平考试~~应考指导~~活动，拟开设有关计算机课程的学习班，由富有教学经验的~~教师~~讲课，并聘请著名计算机专家当顾问，进行水平考试的模拟考试等。~~为此~~我们根据上述需要有针对性地编写了一套教材和习题集，供大家复习、参考。由于时间仓促和编者水平有限，本书有欠妥之处，望读者谅解。

本书由计算机应用软件人员水平考试应试指导联合会教材编写组编写。

1984.11.24

# 目 录

第一章 概述 .....	1
1.1 计算机历史简介 .....	1
1.2 计算机硬件结构简介 .....	2
1.3 软件系统 .....	4
第二章 数与指令系统 .....	6
2.1 数 .....	6
2.1.1 二进制系统 .....	6
2.1.2 基数转换 .....	8
2.1.3 浮点数 .....	11
2.1.4 奇偶校验 .....	11
2.2 指令系统 .....	12
2.2.1 指令 .....	12
2.2.2 指令系统 .....	13
2.2.3 寻址方式 .....	15
第三章 编程步骤和程序设计技巧 .....	23
3.1 编程步骤 .....	23
3.1.1 分析问题, 建立数学模型 .....	23
3.1.2 选择处理方法 .....	25
3.1.3 确定计算逻辑和编制框图 .....	27
3.1.4 编程序 .....	30
3.1.5 机器计算 .....	30
3.1.6 检查程序的正确性 .....	31
3.1.6.1 静态检查 .....	31
3.1.6.2 动态检查 .....	33
3.1.7 结果分析 .....	37
3.2 程序设计技巧 .....	38

3.2.1	递归符号 .....	38
3.2.2	程序流程图 .....	39
3.2.3	表达式的程序设计 .....	40
3.2.4	分支程序设计 .....	46
3.2.5	循环程序设计 .....	58
3.2.5.1	单重循环的程序设计 .....	58
3.2.5.2	多重循环的程序设计 .....	66
3.2.6	程序和程序、程序和子程序的数据传送 .....	71
<b>第四章</b>	<b>数据结构 .....</b>	<b>75</b>
4.1	线性表 .....	77
4.1.1	堆栈、队列、双向队列 .....	78
4.1.2	顺序存贮 .....	80
4.1.3	链式分配 .....	84
4.1.4	索引存贮和散列存贮 .....	93
4.1.5	数组及其存贮方式 .....	95
4.2	非线性表 .....	97
4.2.1	树 .....	97
4.2.2	树的搜索与遍历 .....	99
4.2.3	二叉树 .....	99
4.2.4	树与数组 .....	100
4.2.5	树的存贮 .....	102
4.2.6	已分类树、丰满树、平衡树 .....	113

# 第一章 概 述

## 1.1 计算机历史简介

计算机是一种能自动地高速度地进行大量计算工作的电子机器，它的发明和发展是二十世纪科学技术的卓越成就之一，它的出现，有力地推动了生产、科学技术和文化事业的发展。

一九四六年，世界上出现了第一台电子计算机，共有18000多个电子管，消耗近100千瓦的电力，并占用长度在30米以上的房间，不过它的计算速度有了惊人的提高，这就给计算机的发展开辟了新途径。从这台计算机至今的40年时间内，计算机技术有了飞跃的发展，已更换了五代计算机。上述的机器就是第一代电子计算机。

第二代——晶体管计算机时代，约从1958年到1964年。这一代计算机中的逻辑电路是由晶体管组成的。其性能较第一代计算机提高了数十倍，计算机的软件也由机器语言发展为汇编语言和程序语言，计算机所配的外围设备由几种发展到十几种。因此第二代计算机开始广泛应用于工程计算、数据处理和工业控制等方面。

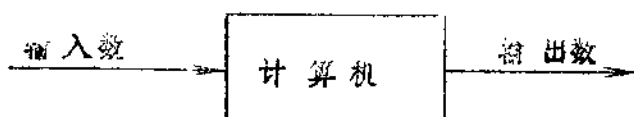
第三代——集成电路计算机时代，约从1964年到1970年，这一代计算机的逻辑电路是由中、小规模集成电路所组成。在中、小规模集成电路中，一块几平方毫米的芯片上集成了几个到几十个门电路（或者说几十到几百个元件）。这样计算机的体积和耗电有了明显减小，运算速度、存储容量有了较大的提高，计算机的系统结构有了许多改进，计算机的软件更进一步完善，出现了高级程序语言和较完善的操作系统。由于集成电路，计算机的可靠性显著提高，价格不断降低，使计算机的应用范围进一步扩大。

第四代——大规模集成电路计算机，从七十年代开始，在这一代计算机中，一块几平方毫米的芯片上已集成一百个门电路（或一千个元件）至五千个门电路（或十万个元件），从而使计算机的体积和耗电更小，可靠性进一步提高，价格进一步降低，出现了由几个片子组成的微型计算机。

目前，正由大规模集成电路向超大规模集成电路发展。在超大规模集成电路中，每个芯片上集成了五千个以上的门电路（或十万个以上的元件），当前所确定的近期研究指标是要在每个芯片上集成一亿到十亿个元件。不难想象，超大规模集成电路的发展将使整个计算机都集成在一个芯片上，出现单片微型计算机。

## 1.2 计算机硬件结构简介

计算机实质上是对数字式信息进行加工的机器。计算机的数入可以是数、数据或是从被控制对象发出的信号，经过计算机的整理和加工，又以新的数、数据或控制信号向外界输出，最简单的计算机框图如下图所示：



和笔算时一样，通常要有二个数才能进行四则运算及其他运算，这二个数叫操作数，得出的答案叫结果。一般的计算机可以进行许多不同方式的信息加工，统称为操作。在同一时间内一般只能进行一种操作，所以如要继续做各种操作，必须按排好顺序依次进行。

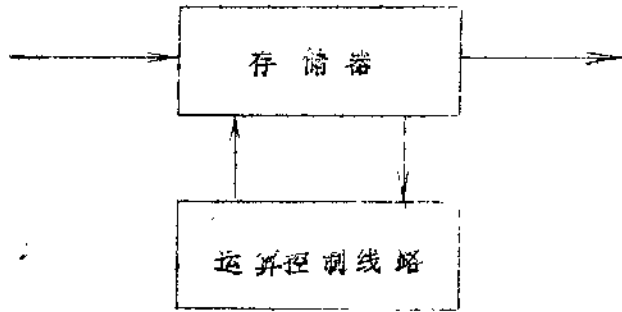
通知计算机进行什么操作的手段是指令。指令是一组数字形式的信号，主要包括二个内容：1) 告诉计算机进行什么操作；2) 指出操作数。以后还会看到，指令还包括其它内容。

要解一个数学题目，必须先把指令按解题步骤排好。如求  $a(b+c)$  的值就需进行  $(b+c)$  操作，然后进行  $a \times (b+c)$  操作，这种按顺序编排好、用指令表示出的计算步骤叫程序。

原始数据和程序都存在计算机的一个记忆装置之内，需要时可把它读出来，它可以写进去，这个记忆装置称为存储器。一台计算机可

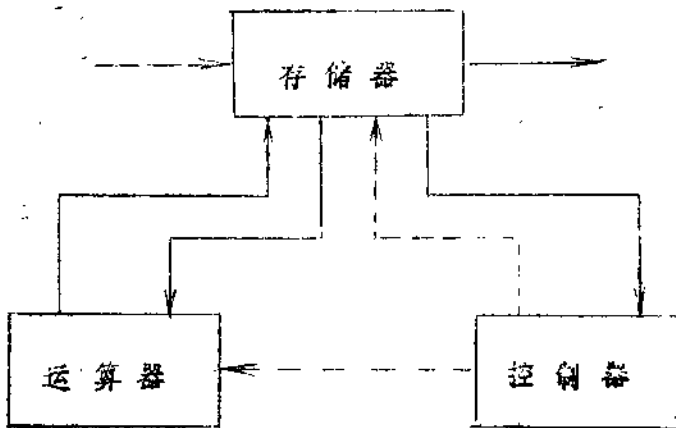
能有几个不同用途的存储器。机器本体内的存储器叫内存储器，简称内存，也叫做主存储器（主存）。

这样计算机本体就可分为二大部分，一部分是运算和控制线路，称中央处理机（CPU），另一部分是存储器（内存）。



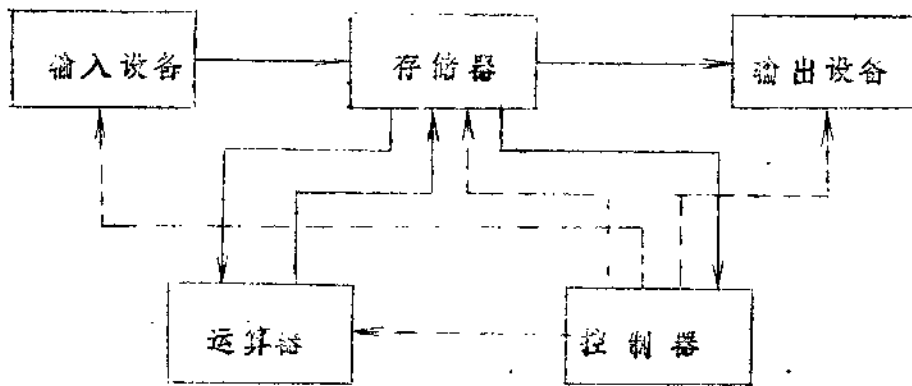
计算机是根据指令的指示进行各项操作的，所以首先要读出指令，然后判别进行何种操作。在大多数情况下，还要根据指令的指示要读出一个操作数。故运算控制线路和内存的交往是很频繁的。把一个数从内存中取出，叫做读，读出后内存的这个数并不消失。把一个数存入内存叫做写。

CPU又分为运算器和控制器二大部分，如下图所示。运算器是直接负责数据加工的部件，控制器则对运算器和存储器发出控制信号，命令它们完成各种指令的操作。图中实线表示数的传送，虚线表示控制信号。



最后计算机还必须具备把原始数据和指令输入内存的半段和计算结果输出的半段，还要靠输入输出设备，简称I/O。输入设备把输入的信息变成电信号按序送入计算机并写入内存，而输出设备则把计算结果以文字形式打印出来或显示出来，或者用其他方式记录在某些记录装置上。输入和输出设备也由控制器控制。这些设备是装在计算机本体之外的，统称为外部设备。

于是我们得到如下的图。在这图中，计算机分为四大部分：运算器、控制器、存储器（内存）和外围设备，这是计算机的基本框图。



### 1.3 软件系统

人们编写的一个基本功能序列称为一段程序。人们编排基本功能序列的工作称为编制（编写）程序，也称为程序设计。为履行某个（某些）特定功能而编制的，行之有效的程序，称为软件（SOFTWARE）。而计算机本身及与其配套的设备统称硬件（HARDWARE）。没有硬件，那末软件失去了效用，但光有硬件，没有配上相应的软件（即所谓裸机），计算机也不能发挥它的潜在能力。

我们设想计算机加电开机后处于初始状态，这时，为让机器干活，总要首先向内存储器送进少量指令称作引导指令序列。启动这个



程序，可以从外存介质引入新的程序。此新程序执行起来，又可引入另一新的程序。有时要分几级引导，从外存介质将操作系统这组程序输入内存存储器，并把控制权转给操作系统，即让操作系统中某个程序执行起来。这时，人们可以通过键盘控制台（用按钮方式）发出各种指示或命令，让计算机转出执行指示中提出的有关程序。故计算机软件中，除了几级引导程序之外，我们可以引出：

- 操作系统。这是一组“管家”程序，是计算机和用户之间的接口。

- 诊断程序。是一组检查计算机本身有否毛病的程序，供硬件维护人员检查和考验机器之用。

操作系统中包含着自己的程序库，供用户调用，称为系统程序库。而用户自己编制的程序称为用户程序，用户程序也可入库保存。系统程序库一般包括：

- 汇编语言编译程序。功能是将输入的符号程序（又称汇编语言程序）逐条译成机器指令。

- 高级语言编译程序。如BASIC，FORTRAN，COBOL，PASCAL，PL/I，APL，C，ALGOL等。

- 其他系统程序。

在操作系统控制之下，还可以建立多个程序子系统。例如

- 文件管理子系统
- 数据管理子系统
- 数据统计软件包
- 运筹计算软件包
- 其他专用子系统。

## 第二章 数与指令系统

计算机所处理的对象是数据，而数据是由数值来表示的，放在本章开始处，有必要对数进行叙述。

### 2.1 数

人们所熟悉的数系是十进制系统，数码为0, 1, 2, ……9, 基数为10, 进位原则是逢十进一。自右至左, 从低位到高位, 分别称为个位( $10^0$ )、十位( $10^1$ )、百位( $10^2$ )、千位( $10^3$ )等等。可是在计算机内表示数和字母字符的适当的进制系统选择是取决于经济、速度和可靠性的考虑, 由于技术上的原因, 电子计算机广泛采用二进制数来记数。计算机内进行的加减乘除运算, 都是基于二进制的, 因此我们必须了解二进制数的一些特点及它与十进制数之间的转换方法。

#### 2.1.1 二进制系统

与十进制系统相仿, 二进制系统也分定点数和浮点数, 我们先讲述二进制定点数。

在以后的表示中, 我们用( )<sub>2</sub>来表示括号内的数是二进制数。

二进制系统的基数是2, 数码只有二种: 0、1, 进位原则是逢二进一, 与十进制数相仿, 二进制数的每一位都代表一定量的数值, 如(0100101)<sub>2</sub> =  $0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$  (相仿于十进制数  $3456 = 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 6 \times 10^0$ , 仅是基数的不同而已)。

由于在计算机中, 数是以二进制形式给出, 而二进制数又有正负数之分, 那么在计算机内如何表示正负数呢? 以字长为16位的计算机作模型。计算机约定: 16位中第0位是0, 则表示此数是正数, 第0位是1, 则表示是负数, 换句话说, 第0位是符号位, 而非

数值位，其余的15位自左至右由高位至低位就表示这个数的绝对值。

例如：

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

如这个数是整数，即认为小数点在位15之后，则 $(+000010000010010)_2 = 2^{10} + 2^4 + 2^1$ ，而数 $(1000010000010010)_2 = -(2^{10} + 2^4 + 2^1)$ 。如果认为小数点在位12与位13之间，则 $10000010000010010)_2 = 2^7 + 2^1 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 3^{-2}$ ，而数 $(1000010000010010)_2 = -(2^7 + 2^1 + 1 \times 2^{-2})$ 。

须指出，由于字长的限制，用一个单元表示二进制定点数时，所能表示的数值范围也受到限制，16位字长的字能表示的最大正数（认为小数点在位15之后）是：

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

这个数相当于十进制数的 $2^{15} - 1$ ，16位字长能表示的最大负数是 $-(2^{15} - 1)$ ，因此16位字长的字能表示的定点数 $x$ 的范围为 $-(2^{15} - 1) \leq x \leq 2^{15} - 1$ 。还应指出，就是在此指定的范围内，它也只能表示有穷个有理数，并不是所有实数都能表示出来。比如当 $|x| < 2^{-15}$ ，即 $x$ 的绝对值很接近于0但不等于0时，计算机不能准确地表示出来，这是因为，能表示的绝对值最接近于0的二个数分别是 $2^{-15}$ （小数点紧在数符后）和 $-2^{-15}$ 。由此可见，一个16位字长的字所能表示的定点数 $x$ ： $-2^{15} \leq x \leq 2^{15} - 1$ 。如我们给的数，或计算机计算过程中出现的中间结果，超过了这个精度范围，计算机就无法表示了。遇到这种情况，称为发生了溢出。当 $|x| > 2^{15} - 1$ 时称上溢，当 $0 \neq |x| < 2^{-15}$ 时称下溢。

为能提高计算机所能表示的数的精度，采用两个字或三个字，甚至四个字连接起来表示一个定点数，因此有半精度定点数，双精度定点数和三倍精度定点数之称。

二进制定点数的四则运算规则，与十进制数的四则运算规则相似。

### 2.1.2 基数转换

在使用中只要有多于一个数系，数从一个系统转换到另一个数系是很重要的，在计算机中最常使用的数系是二进制、八进制、十六进制和十进制系统。下面就它们之间的转换加以叙述。

#### 二进制——八进制转换

由于  $8=2^3$ ，故一个二进制数在八进制系统中是很容易表示的。对一个二进制数，自小数点为分隔，向右（小数部分）依次三位一组或向左（整数部分）依次三位一组进行分划，然后对每个组获得等价的八进制数（见二——八进制转换表）。类似地，对每个八进制数字是由等价的三个二进制数字来代替。

例，二转八

$$(111101011001.01100101)_2 = (011,101,011,001,011,001,010)_2 \\ = (3531.312)_8$$

八转二

$$(3456.123)_8 = (011,100,101,110,001,010,011)_2 \\ = (11100101110,001010011)_2$$

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

二——八进制转换表

#### 二进制——十六进制转换

十六进制的基数是16，数码是0,1,2, ..., 9,A,B,C, D,E,F（其中A,B,C,D,E,F分别表示10,11,12,13,14,15），进位原则是逢十六进一。由于  $16=2^4$ ，故二——十六转换时，与二——八转换相仿，只不过是4位一组来分划。

例子, 二转十六

$$(111101011001.01100101)_2 = (1111, 0101, 1001.0110, 0101)_2 \\ = (F59.65)_{16}$$

十六转二

$$(E3D.4F)_{16} = (1110, 0011, 1101.0100, 1111)_2 \\ = (111000111101.01001111)_2$$

十六进制	0	1	2	3	4	5	6	7
二进制	0000	0001	0010	0011	0100	0101	0110	0111
十六进制	8	9	A	B	C	D	E	F
二进制	1000	1001	1010	1011	1100	1101	1110	1111

## 二——十六基数转换器

转换成十进制

二转十

$$(111011011)_2 = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ = 32 + 16 + 8 + 2 + 1 + 0.25 + 0.125 \\ = 59.375$$

八转十

$$(123.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0 + 4 \times 8^{-1} \\ = 64 + 16 + 3 + 0.5 \\ = 83.5$$

十六转十

$$(E3.8)_{16} = 14 \times 16^1 + 3 \times 16^0 + 8 \times 16^{-1} \\ = 227.5$$

十进制转换成r进制

对这个转换分成二部分:

(1) 把一个整数从十进制转换成r进制, 使用基数相除的办法, 这由给定的十进制数被基数r逐次相除来实现。在每次相除中, 余数

部分表示了转换后的新数字，从第一次相除得到的余数给出新数的最低有效数字，然后第一次相除的商再被基数  $r$  除，产生一个新的商和余数，这过程继续至商小于基数  $r$ 。

例 把 4857 转换成八进制数

8	4857	1	第一次余数
8	607	7	第二次余数
8	75	3	第三次余数
8	9	1	第四次余数

最高位  $\rightarrow$  1

故八进制数为 11371

(2) 对于十进制小数的转换，使用与  $r$  重复相乘的方法。首先用基数  $r$  乘小数，其结果的整数部分取作为新数的最高有效数字，它的小数部分再被基数  $r$  乘，第二次相乘的结果又产生整数和小数，整数部分是新数的第二个数字。小数部分再与  $r$  相乘。这过程继续至相乘的小数部分变成 0。

例 把  $(0.6328125)_{10}$  转换成八进制数。

$$0.6328125 \times 8 = 5.0623 \quad \text{整数是 5}$$

$$0.625 \times 8 = 0.5000 \quad \text{整数是 0}$$

$$0.5000 \times 8 = 4.000 \quad \text{整数是 4, 小数部分为 0}$$

故  $0.6328125 = (0.504)_8$ 。

又例 把 0.4 转换为二进制数

$$0.4 \times 2 = 0.8 \quad \text{整数为 0}$$

$$0.8 \times 2 = 1.6 \quad \text{整数为 1}$$

$$0.6 \times 2 = 1.2 \quad \text{整数为 1}$$

$$0.2 \times 2 = 0.4 \quad \text{整数为 0}$$

$$0.4 \times 2 = 0.8 \quad \text{整数为 0}$$

⋮  
⋮

$\approx 10 \approx$

相乘过程中小数不为0，即表示二进制系统中不存在0.4的有限精确表示，这时就应视你所需的精度。例如这儿取 $(0.4)_{10} = 0.0110$ 。

综上所述  $4857.6328125 = (11371.504)$ 。

### 2.1.3 浮点数

早期计算机只设计成具有定点运算部件，因此程序员所负有的责任应兼顾小数点位，当然这是一件化费时间的工作，而由于计算机发展到已具有完成浮点运算的能力，故现在计算机上都具有浮点运算能力，这就使得计算机在处理数值量中允许最大的灵活性。

通俗的说，可用形式  $C \times B^e$  来表示任何数值量，如十进制数7086能写成  $7.086 \times 10^3$  或写成  $708.6 \times 10^1$ ，采用这种表示，一个数由二个部分组成——尾数和阶码（指数），在计算机内通常尾数用小数值形式来表示，小数点是在小数部分的最高有效位的左边，这种表示称为规格化尾数。如1110100.01浮点表示有二部分：尾数是0.111010001，阶码是+7的二进制数，而非尾数

0.0111010001，阶码+8的二进制数。可归纳如下： $\frac{1}{2} \leq \text{尾数} < 1$ ，即尾数的最高位有效值1，而阶码部分随之相应地起变化。

很显然，对浮点数的乘除运算，是分二部分分别完成的：尾数与尾数相运算，阶码与阶码相运算，而对浮点数的加减运算，须注意阶码的一致性。

### 2.1.4 奇偶校验

奇偶校验是一种用来检测数据错误的常用技术，错误是由电气干扰或设备故障而使数据的某些位值发生变化而引起的，奇偶校验是在数据中附加一个额外位（奇偶位）来实现的。

☆ × × × × × × ×

└───┬───  
奇偶位位置

≈ 11 ≈

在偶校验中，奇数位可以是1也可以是0，决定的原则是使8位字符中1的总数为偶数。

故：R = 11010010, S = 01010011

奇校验的情形正好相反。如采用奇校：R = 01010011,  
S = 11010010

在偶校验系统中，奇偶校验是检查正在处理的每个代码，是否具有偶数个1，如若，则出现数据出错，这个字符就无意义。

可以看出，刚才讨论的奇偶校验是不能检测出全部错误，如在偶校验系统中，在R中改变了二位，则仍有偶数个1。

即：正确 R = 11010010      4个1

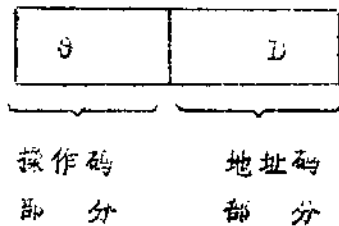
实际上 R = 11011110      6个1 仍不发生出错

有关奇偶校验更详细的信息，请参见“计算机硬件基础”部分。

## 2.2 指令系统

### 2.2.1 指令

计算机完成某种操作是由一串特殊的电信号(称为控制信息)控制机器内某些部件而达到的，计算机的这种控制信息就是指令，不同的控制信息表示不同的指令，而一条指令能使计算机完成某种操作，如  $1-1 \times 1$  等等。一条指令完成某种控制功能，必须有二大部分信息，一部分是告诉计算机的哪一种操作(操作码部分)，另一部分告诉机器从内存的哪些单元取出参与运算的数(操作数)，故指令的代码形式可写成

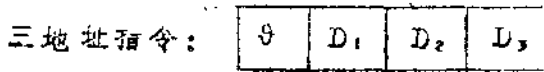


O是某个操作如+、-、 $\times$ 、 $\div$ 等等，D是操作数的地址。



指令的具体形式与计算机内部结构(硬件)有密切关系,不同的机器结构有不同的指令形式。

指令形式:



地址的部分有三个地址

D<sub>1</sub>为第一操作数在内存的单元地址。

D<sub>2</sub>为第二操作数在内存的单元地址。

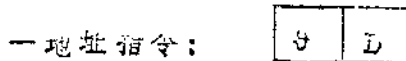
D<sub>3</sub>为本次运算结果存入内存中去的单元地址。

指令的功能是:  $(D_1) \theta (D_2) \longrightarrow D_3$ ,

(D)表示地址为D的单元的内容



功能:  $(D_1) \theta (D_2) \longrightarrow D_2$



功能:  $(L) \theta (L) \longrightarrow L$  L是累加寄存器

在‘计算机硬件基础’中,我们已经知道寄存器是如何一回事,在此不加以重述。

可以看出以上三种指令形式都没有专门的地址码表示下一条指令的地址,这是因为在一般情况下,指令总是顺序执行的,故只要设一个指令计数器,每执行完一条指令后,指令计数器加1就可以指出下一条指令地址,当然对于跳转指令,只要把指令中的跳转地址置入指令计数器即可。

### 2.2.2 指令系统

一台计算机由硬设备实现其功能并供程序人员使用的各种不同类型的指令的总和称为该机的指令系统,不同机器指令系统所包含的指令种类和数目是不同的。在设计指令系统过程中,应考虑指令系统的完备性。何谓完备性?是讲对一切可以计算的问题都可以用这台计算机的指令系统编制程序,而且能在该机上实现自动计算并得到正确结