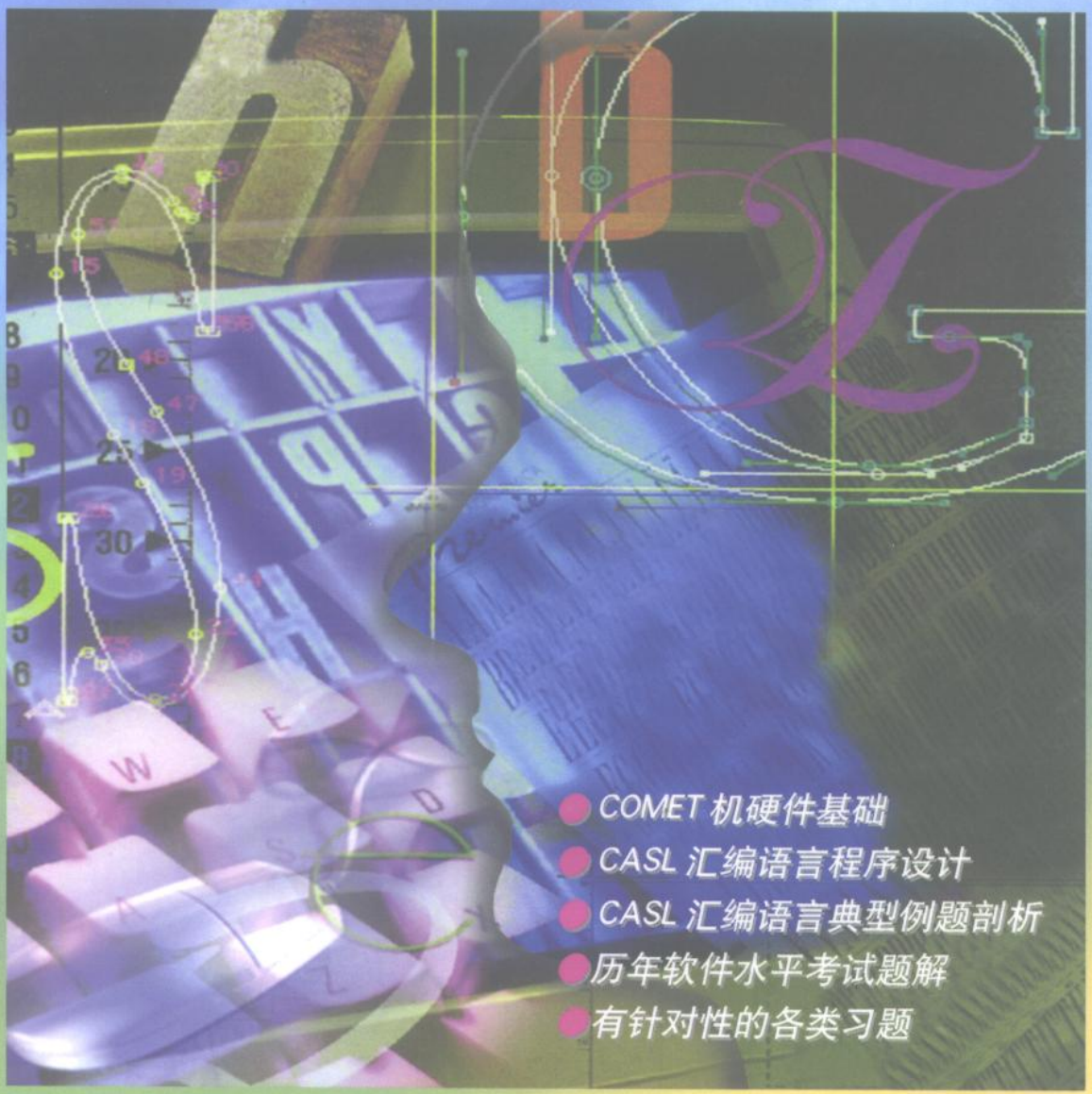
计算机软件水平考试

CASL语言精解

刘云楚　马文涛　陈　建　编著



- COMET机硬件基础
- CASL汇编语言程序设计
- CASL汇编语言典型例题剖析
- 历年软件水平考试题解
- 有针对性的各类习题

北京航空航天大学出版社

计算机软件水平考试
CASL 语言精解

刘云楚　马文涛　陈　建　编著

北京航空航天大学出版社

内 容 简 介

本书是为计算机软件专业人员学习CASL汇编语言，参加高级程序员级资格和水平考试编写的一本自学教材。

本书分为三个部分共九章。第一部分从COMET计算机硬件的基本结构出发，系统介绍了COMET计算机的寻址方式、指令系统。第二部分介绍了CASL汇编语言程序设计的方法和技巧。第三部分侧重实用，着重讲解CASL汇编语言试题的解题方法，尤其第七章结合数据结构，详细讲述了数组、字符串、链表和队列在CASL汇编语言中的应用。

本书深入浅出、循序渐进，既注意知识的系统性，又强调其实用性。书中例题丰富、取材新颖、富于启发，且大多数选自历年CASL汇编语言考题。为了使读者进行有针对性地复习，书中的后几章介绍了大量例题，所有例子均以模拟题的形式出现，可备读者应考前测试自己能力之用。书末还附有一定数量的习题，供读者进一步练习。

本书可作为参加计算机软件水平考试的辅导教材，也可以作为CASL汇编语言程序设计课程以及计算机软件专业人员的参考书。

图书在版编目(CIP)数据

计算机软件水平考试：CASL语言精解/刘云楚等编著.
—北京：北京航空航天大学出版社，1999.9
ISBN 7-81012-913-9

Ⅰ.计… Ⅱ.刘… Ⅲ.CASL语言 Ⅳ.TP312

中国版本图书馆CIP数据核字(1999)第27838号

计算机软件水平考试

CASL语言精解

刘云楚 马文涛 陈 建 编著

责任编辑 乔少杰

*

北京航空航天大学出版社出版发行

北京市学院路37号(100083) 发行部电话(010)82317024 发行部传真(010)82328026

http://www.buaapress.cn.net

E-mail:pressell@publica.bj.cninfo.net

北京宏文印刷厂印装 各地书店经销

*

开本：787×1092 1/16 印张：14.25 字数：365千字

1999年10月第1版 1999年10月第1次印刷 印数：4000册

ISBN 7-81012-913-9/TP·359 定价：22.00元

前　言

本书主要面向参加计算机软件水平考试的人员。CASL 汇编语言是定义在虚拟计算机 COMET 上面的一种抽象计算机语言，1987 年日本计算机应用软件人员全国统考委员会把这种语言纳入考试的范围。1991 年我国计算机软件考试中心颁布的考试大纲规定，CASL 汇编语言是程序员级选考和高级程序员级必考的程序设计语言。1996 年颁布的考试大纲仍然将 CASL 汇编语言作为高级程序员级的必考内容。

在这种需要下，笔者编写了这本书。考虑到读者的实际需要，以大量实例阐述了 CASL 汇编语言的程序设计方法；同时，为了使读者有的放矢地复习，例题全部采用模拟题的形式。希望读者勤加练习，不断提高解题能力。

本书分为三个部分：

第一部分，包括第一、二、三、四章。主要介绍了预备知识、COMET 计算机硬件基础、寻址方式和指令系统、CASL 语句的功能及一些重点语句的详细分析，为读者学习 CASL 汇编语言的程序设计打下基础。

第二部分，包括第五章和第六章。结合具体例子介绍了 CASL 汇编语言程序设计的基本方法和特点，内容包括简单程序设计、分支程序设计、循环程序设计和子程序设计方法。尤其对考生反映比较难的子程序设计问题作了详尽的解释和分析。

第三部分，包括第七、八、九章。绝大部分内容选自历年 CASL 考题和部分近年来日本 CASL 汇编语言试题。特别是第七章，结合数据结构，详细讲述了数组、字符串、链表和队列等在 CASL 汇编语言中的应用。这是为那些应考心切的读者做热身训练、测试自己的实力作准备的。建议读者在学习这几章时不要先看试题的答案，应独立思考，自己解答之后再与正确答案比较。

书中还备有一定数量的习题及答案，供读者进一步练习用。

第一、二章由马文涛编写，第三、四章由湖南银行学校陈建编写，其余各章由刘云楚编写。全书由刘云楚总纂和定稿。

由于作者水平所限，加之时间仓促，书中错误和不妥之处在所难免，敬请读者批评指正。

编著者

1999 年 6 月

目　录

第一章 预备知识

本章主要介绍数据在计算机中的表示方法、计算机中数据的算术运算和逻辑运算等。

1.1 数制及其转换

1.1.1 进位计数制

在日常生活和工程设计中，使用最普遍的是十进制数。而在计算机内部，数据以二进制数的形式表示，用二进制形式对这些数据进行算术逻辑运算，并且以二进制形式存储数据。同一个数，在不同的数制下面有不同的表示方式，这便构成了不同进制的数制系统。

一、二进制数

进位计数制是一种计数的方法。一个任意的十进制数 X 可以表示为：

$$\begin{aligned} X &= a_n a_{n-1} \cdots a_1 a_0 a_{-1} a_{-2} \cdots a_{-m} \\ &= a_n \times 10^n + a_{n-1} \times 10^{n-1} + \cdots + a_{-1} \times 10^{-1} + a_{-2} \times 10^{-2} + \cdots + a_{-m} \times 10^{-m} \\ &= \sum_{i=-m}^{n} a_i \times 10^i \end{aligned}$$

十进制数是人们最熟悉的一种数制，其基数为 10，即其数码的个数为 10，且遵循逢十进一的规则。在上面公式中 10^i 称为该位数字的权，每个数乘以其权所得到的乘积之和即为所表示数的值。

二进制数的基数为 2，只有 0，1 两个数码，并遵循逢二进一的规则。显然，它的权是 2^i。因此，任何一个二进制数可以表示为：

$$\begin{aligned} (X)_2 &= a_n a_{n-1} \cdots a_1 a_0 a_{-1} a_{-2} \cdots a_{-m} \\ &= a_n \times 2^n + a_{n-1} \times 2^{n-1} + \cdots + a_{-1} \times 2^{-1} + a_{-2} \times 2^{-2} + \cdots + a_{-m} \times 2^{-m} \\ &= \sum_{i=m}^{n} a_i \times 2^i \end{aligned}$$

n 位二进制数可以表示 2^n 个数。例如，3 位二进制数可以表示 8 个数(见表 1.1)。

表 1.1

二进制数	000	001	010	011	100	101	110	111
对应的十进制数	0	1	2	3	4	5	6	7

而 4 位二进制数可以表示 16 个数，且它们与十进制数的对应关系如表 1.2 所示。

表 1.2

二进制数	0000	0001	0010	0011	0100	0101	0110	0111
对应的十进制数	0	1	2	3	4	5	6	7
二进制数	1000	1001	1010	1011	1100	1101	1110	1111
对应的十进制数	8	9	10	11	12	13	14	15

二、八进制数

八进制数遵循逢八进一的规则。八进制数的基数为8，其权值是8^i，构成八进制数制系统的是以下8个数字：

0,1,2,3,4,5,6,7

一个八进制数的值等于它的各位数字与其权乘积之和，任何一个八进制数可表示为：

$$(X)_8 = a_n a_{n-1} \cdots a_1 a_0 a_{-1} a_{-2} \cdots a_{-m}$$

$$= a_n \times 8^n + a_{n-1} \times 8^{n-1} + \cdots + a_{-1} \times 8^{-1} + a_{-2} \times 8^{-2} + \cdots + a_{-m} \times 8^{-m}$$

$$= \sum_{i=-m}^{n} a_i \times 8^i$$

三、十六进制数

十六进制数遵循逢十六进一的规则。十六进制数的基数为16，构成十六进制数制系统的是以下16个符号数字。

0,1,2,3,4,5,6,7,8,9,*A*,*B*,*C*,*D*,*E*,*F*

一个十六进制数的值等于它的各位数字与其权的乘积之和。任何一个十六进制数可表示为：

$$(X)_{16} = a_n a_{n-1} \cdots a_1 a_0 a_{-1} a_{-2} \cdots a_{-m}$$

$$= a_n \times 16^n + a_{n-1} \times 16^{n-1} + \cdots + a_{-1} \times 16^{-1} + a_{-2} \times 16^{-2} + \cdots + a_{-m} \times 16^{-m}$$

$$= \sum_{i=-m}^{n} a_i \times 16^i$$

表1.3给出了0～15十进制数的二进制、八进制和十六进制对照表。

表 1.3

十进制	二进制	八进制	十六进制
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8

续表 1.3

十进制	二进制	八进制	十六进制
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1.1.2 数制间的转换

一、二进制数与十进制数之间的转换

1. 二进制数转换为十进制数

二进制数各位数字与其对应的权的乘积之和即为该二进制数所对应的十进制数。

例如：

$$(11001.11)_2 = 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2}$$
$$= 25.75$$

2. 十进制数转换为二进制数

任意十进制数转换为二进制数可按以下步骤进行：

第一步，先把十进制数的整数部分(25)转换为二进制数。常用的方法有两种：

① 除以 2 取余法

2 | 25 ____ 余 1 低位
2 | 12 ____ 余 0
2 | 6 ____ 余 0
2 | 3 ____ 余 1
2 | 1 ____ 余 1 高位
0

最后算得结果：$(25)_{10}=(11001)_2$

② 待定系数法

第一步，把数展开成 2 的整数次幂之和，找到对应位上 a_i 的值。

例：$25=1\times 2^4+1\times 2^3+1\times 2^0$

即：$a_4=1$ $a_3=1$ $a_2=0$ $a_1=0$ $a_0=1$ 那么$(25)_{10}=(11001)_2$

第二步，把十进制数的小数部分转换为二进制数。

假定将十进制小数(0.375)转换成二进制小数。常用的方法是把给定的十进制小数乘以 2，取积的整数部分，得到二进制小数的小数点后的第 1 位；乘积的小数部分再乘以 2，积的整数部分为小数点后的第 2 位；一直重复做下去，就可得到希望的二进制小数。

	整数部分	
$0.375\times 2=0.75$	0	小数点后第 1 位数
$0.75\times 2=1.5$	1	小数点后第 2 位数

0.5×2=1.0　　1　　小数点后第3位数

第三步，把整数部分和小数部分所转换成的二进制数结果合并，即完成转换任务。

$$(25.375)_{10} = (11001.011)_2$$

二、二进制与十六进制间的转换

二进制的基数是2。十六进制的基数是16，恰好是2的4次乘方，那么1位十六进制数对应4位二进制数，根据这个特点，可以很方便地实现二进制数与十六进制数的相互转换。

1. 二进制数转换为十六进制数

任意一个二进制数，对于小数点左边自右向左每4位二进制化为1位十六进制数；对于小数点右边自左向右每4位二进制数化为1位十六进制数，就可以实现转换。在转换过程中，在二进制左边(高位)可根据需要任意地加0(对于无符号数)，以使总的位数正好是4的倍数。

例如：二进制数　　1011 1101 0111 1111

十六进制数　　B　D　7　F

二进制数　　10 1010 1110 0011.1101 10

十六进制数　　2　A　E　3．D　8

2. 十六进制数转换为二进制数

对于任意1个十六进制数，把每位十六进制数化为4位二进制数即可转换为二进制数。

例如：十六进制数　　4　B　F．　E　C

二进制数　　100 1011 1111.1110 1100

三、二进制数与八进制数间的相互转换

1. 二进制数转换为八进制数

任意一个二进制数，对于小数点左边自右向左每3位二进制数化为1位八进制数；对于小数点右边自左向右每3位二进制数化为1位八进制数，这样就可完成转换。

例如：二进制数　　101 101 000 111.101 101 11

八进制数　　5　5　0　7　．　5　5　6

2. 八进制数转换为二进制数

任意一个八进制数，把每位八进制数化为3位二进制数即可完成转换任务。

例如：八进制数　　3　5　7　．　6　3

二进制数　　11 101 111．110 011

四、十进制数转换为八进制数和十六进制数

1. 十进制数转换为八进制数

任意一个十进制数，对于整数部分N采用“除8取余法”化为八进制数。即用8去除N，得一整数商和一余数，此余数是八进制数的最低位；再用8去除所得的商，得一整数商和一余数，此余数是八进制数的次低位。这样一直除下去，直到商为0时为止。对于小数部分M采用“乘8取整法”。即用8去乘M，得一乘积，其整数部分就是八进制小数的最高位；再用8乘上面所得积的小数部分，又得一乘积，其整数部分是八进制小数的次高位。这样一直乘下去，直到乘积的小数部分为0时为止。

例如，把十进制数 323.3125 转换为八进制数：

$323 \div 8 = 40$ 余 3，即 $a_0 = 3$

$40 \div 8 = 5$ 余 0，即 $a_1 = 0$

$5 \div 8 = 0$ 余 5，即 $a_2 = 5$

故有 $(323)_{10} = (503)_8$

对小数部分采用“乘 8 取整法”：

$0.3125 \times 8 = 2.5$ 即 $a_{-1} = 2$

$0.5 \times 8 = 4$ 即 $a_{-2} = 4$

故有 $(0.3125)_{10} = (0.24)_8$

最后转换结果：

$$(323.3125)_{10} = (503.24)_8$$

2. 十进制数转换为十六进制数

转换方法与十进制数转换为八进制数完全类似，只是将“除 8 取余法”变为“除 16 取余法”，将“乘 8 取整法”变为“乘 16 取整法”。

例如，把十进制数 323.3125 转换为 16 进制数：

$323 \div 16 = 20$ 余 3，即 $a_0 = 3$

$20 \div 16 = 1$ 余 4，即 $a_1 = 4$

$1 \div 8 = 0$ 余 1，即 $a_2 = 1$

故有 $(323)_{10} = (143)_{16}$

对小数部分采用“乘 16 取整法”：

$$0.3125 \times 16 = 5.0 \quad \text{即 } a_{-1} = 5$$

故有 $(0.3125)_{10} = (0.5)_{16}$

最后转换结果：

$$(323.3125)_{10} = (143.5)_{16}$$

1.2 原码、补码和反码

1.2.1 机器数与真值

在计算机内，有符号数的编码称为机器数。机器数所表示的数值称为真值。

数有正、负两种，在计算机中数的符号是用数码表示的。一般情况下，用 0 表示正数，用 1 表示负数。通常符号位放在数的最高位。例如：$x_1 = (+1000011)_2$，$x_2 = (-1000011)_2$，它们在机器中表示为：

$$x_1 = 01000011$$

$$x_2 = 11000011$$

1.2.2 机器中常用的几种码制表示法

不同的计算机中，有符号数的表示方法并不完全相同。通常有 3 种表示法，下面分别加以

介绍。

一、原码表示法

1. 定 义

在有符号数的绝对值前面附加一个符号位就称为原码。在计算机中，正号用0表示，负号用1表示。设 $m+1$ 位二进制数的绝对值为 $X_m\cdots X_0$，则原码的定义为：

$$[X]_{原}=\begin{cases}0X_m\cdots X_0, & \text{若 } X \text{ 的真值} \geqslant 0\\ 1X_m\cdots X_0, & \text{若 } X \text{ 的真值} < 0\end{cases}$$

例如，(+71)和(−71)在计算机中用原码表示为(设机器字长为8位)：

$$[+71]_{原}=01000111$$
$$[-71]_{原}=11000111$$

2. 真值零的原码表示法

对于真值零，有正零和负零两种表示法：

$$[+0]_{原}=000\cdots000$$
$$[-0]_{原}=100\cdots000$$

3. 原码表示的数的个数

n 位二进制位有 2^n 个不同的状态，即可以表示 2^n 个数。但由于零占掉了两个原码，所以，用 n 位二进制位只能表示 (2^n-1) 个原码数。

例如，$n=16$，则16位二进制位能表示65 535(即 $2^{16}-1$)个原码数。

原码表示法比较简单，但其缺点是加减法运算复杂。若要把两个数相加，计算机必须首先判断这两个数符号是否相同，如果相同则相加，如果不同则进行减法运算。在进行减法时，计算机首先比较两个数的绝对值的大小，然后用大数减去小数，再选取绝对值大的数的符号作为结果的符号。由此可见，计算机内的数不适合用原码表示法。

二、反码表示法

1. 定 义

一个正数的反码等于该数的原码。一个负数的反码等于该负数所对应的正数的原码按位取反，即0变1，1变0。

例如，(+71)和(−71)在计算机中用反码表示为(设机器字长为8位)：

$$[+71]_{反}=01000111$$
$$[-71]_{反}=10111000$$

2. 真值零的反码表示法

对于真值零，有正零和负零两种表示法：

$$[+0]_{反}=000\cdots000$$
$$[-0]_{反}=111\cdots111$$

3. 反码表示的数的个数

由于零表示的不唯一性，n 位反码和 n 位原码一样，只能表示 (2^n-1) 个数。

例如，$n=16$，则16位二进制位能表示65 535(即 $2^{16}-1$)个反码数。

三、补码表示法

1. 定 义

一个正数的补码等于该数的原码。一个负数的补码等于该数的反码在末位上加 1。

例如，(+71)和(−71)在计算机中用补码表示为(设机器字长为 8 位)：

$$[+71]_{补}=01000111$$

$$[-71]_{补}=10111001$$

2. 真值零的补码表示法

真值零的补码表示为：

$$[+0]_{补}=000\cdots000$$

$$[-0]_{补}=2^n+000\cdots000$$

由于 2^n 在机器中表示不出来，自动丢掉，所以，在补码系统中，零的表示是唯一的。

3. 补码表示的数的个数

在补码系统中，由于零有唯一的表示，因而，n 位二进制数能表示 $2n$ 个补码数。

4. 补码的运算规则

补码的加法运算规则： $[X+Y]_{补}=[X]_{补}+[Y]_{补}$

补码的减法运算规则： $[X-Y]_{补}=[X]_{补}+[-Y]$ 补

根据补码的运算规则可以得出以下结论：

① 任何两个数相加，无论其正负号如何，只要对它们的补码实施加法运算即可得到正确结果，只不过这结果是补码形式。

② 减法求补变加法。若两个数相减，对减数求补，则变成被减数与[减数]$_{补}$相加即可得到正确结果，只不过这结果是补码形式。

从最高有效位向高位的进位，由于机器字长限制自动丢失。

四、码制表示法小结

1. 如果真值 X 为正数，则$[X]_{原}=[X]_{反}=[X]_{补}$；如果为负数，则均有不同的表示。

2. 如果 X 为正数，则$[X]_{原}$、$[X]_{反}$和$[X]_{补}$表示数的范围相同；如果 X 为负数，则$[X]_{补}$比$[X]_{原}$和$[X]_{反}$表示的数多一个。

3. 如果 X 为零，则补码表示是唯一的，而原码和反码有两种不同的表示。

1.3 计算机的数码系统

1.3.1 符号编码

计算机不但要处理数值领域的问题，而且要处理大量非数值领域的问题。例如：文字、字母、符号以及一些专用符号，以便表达文字信息。这些信息也要写成二进制格式的代码，存入计算机中才能对它们进行加工处理。国际上普遍运用一些标准化代码，最常用的是 ASCII 码(American Standard Code for Information Interchange，美国国家信息交换标准字符码)和 EBCDIC 码(Extended Binary Coded Decimal Interchange Code，扩展 BCD 交换码)等。

ASCII码规定7位二进制数表示一个符号。其中包括数字符号、字母、特殊符号等可打印的字符，其范围为：

20 H ～ 7E H

还有若干个控制代码，其范围为：

0 ～ F H 及 7F H

全部ASCII编码见附录B。在计算机中，常用8位二进制表示一个字符。这是因为8位二进制正好为一个字节。在以字节编址的存储器中，一个单元正好存储一个字符。常用第8位作为奇偶效验位，以提高信息传输的可靠性。

EBCDIC编码是用8位二进制表示一个字符，因此，它可以表示256个符号。IBM公司在它的各类大型机上广泛采用EBCDIC编码。

1.3.2 十进制数的几种编码

二—十进制(BCD)编码是用4位二进制数表示1位十进制数。它具有二进制的形式，又具有十进制的特点。

4位二进制数，有16种不同的状态，从中选出10个状态来表示十进制数的0～9，有多种选择方法。这里主要介绍8421码和两种无权码。

一、8421编码—BCD码

8421码也叫BCD码，是一种有权码，从高位开始各位的权分别是8、4、2、1。一个十进制数要用8421码表示，把这个十进制数的每一位数字用8421码书写即可。例如，十进制数48，用BCD码表示为0100 1000，其中前4位表示4，后4位表示8。而用二进制表示则为00110000。

十进制数字符号共有10个，4位二进制数可有16种不同编码。用4位二进制数表示一个十进制数的符号。n位十进制数就要用$4\times n$位二进制数。

例如，3579的BCD编码为：

0011 0101 0111 1001

反过来，一个BCD编码也很容易表示为十进制数。

例如，BCD编码 1000 0111 0110 0101 它所表示的十进制数为8765。

二、4位无权码

4位无权码主要有余3码和格雷码，它们与十进制符号的对应关系见表1.4。

余3码是由8421码加0011得来的，各位没有权的关系。格雷码也是一种无权码，它的编码规则是使相邻两代码之间只有一位是不同的。这就使得代码变换是连续的，它用于计数器译码，在产生各种控制信号时特别有用。

表 1.4

十进制符号	有权码 8421 码	4 位无权码		
		余 3 码	格雷码	
0	0000	0011	0000	0000
1	0001	0100	0001	0100
2	0010	0101	0011	0110
3	0011	0110	0010	0010
4	0100	0111	0110	1010
5	0101	1000	1110	1011
6	0110	1001	1010	0011
7	0111	1010	1000	0001
8	1000	1011	1100	1001
9	1001	1100	0100	1000

1.4 数的算术运算

1.4.1 二进制数的算术运算

加法规则：

0+0=0

0+1=1

1+0=0

1+1=0（进位 1）

乘法规则：

0×0=0

0×1=0

1×0=0

1×1=1

二进制数减法与十进制数减法类似，够减时是直接相减，不够减时服从向高位借 1 为 2 的规则。

例： 二进制数的算术运算

$$\begin{array}{r} 10111 \\ +11011 \\ \hline 110010 \end{array} \qquad \begin{array}{r} 11101 \\ -10011 \\ \hline 01010 \end{array} \qquad \begin{array}{r} 10101 \\ \times 11001 \\ \hline 10101 \\ 00000 \\ 00000 \\ 10101 \\ +10101 \\ \hline 1000001101 \end{array}$$

1.4.2 八进制数的算术运算

八进制加法：当两个一位数之和 S 小于 8 时，与十进制数同样处理；若两个一位数之和≥

8时，则应该用进位1和($S-8$)取代。

八进制减法和十进制数减法类似，够减时直接相减，不够减时服从向高位借1为8的规则。

八进制乘法可用十进制数乘法规则来计算，但结果必须用八进制数表示。

例： 八进制数的算术运算(后缀为Q表示八进制)

```
  632 Q      632 Q          632 Q
+ 547 Q    - 547 Q        × 547 Q
-------    -------    ------------
 1401 Q     063 Q          5466
                           3150
                         + 4002
                    ------------
                        437366 Q
```

1.4.3 十六进制数的算术运算

十六进制加法：当两个一位数之和S小于16时，与十进制数同样处理；当两个一位数之和$S\geqslant 16$时，则应该用进位1及($S-16$)取代。

十六进制减法也和十进制数减法类似，当够减时则可直接相减，当不够减时服从向高位借1为16的规则。

十六进制乘法可用十进制数的乘法规则，但结果必须用十六进制数表示。

例： 十六进制数的算术运算(后缀为H表示十六进制)

```
  2D34 H     2D34 H          2D34H
+ 05CB H   - 05CB H        × 05CB H
---------  ---------    ------------
  32FF H     2769 H          1F13C
                            21E70
                          + E204
                    ------------
                      105DC3C H
```

1.5 数的逻辑运算

逻辑代数的变量值只有两个："0"和"1"。在逻辑代数中存在着逻辑与、逻辑或、逻辑非和异或运算。

1.5.1 "逻辑与"运算(AND)

"逻辑与"运算又称逻辑乘，运算符为·或∧。从表1.5可以看出逻辑乘的运算规则：只有两个变量取值全为1时，Z的值才是1，其它情况全为0。

逻辑乘的这种关系也扩展到任意个变量相乘。假定$D=A\cdot B\cdot C$，只有当A、B、C三个变量全为1时，D的值才为1。

1.5.2 "逻辑或"运算(OR)

"逻辑或"运算又称逻辑加，运算符为+或∨。从表1.6可以看出逻辑或的运算规则：只要两个变量中任意一个取值为1时，Z的值就是1，只有当两个变量全为0时，Z的值才是0。