计算机等级考试系列教程

微机汇编语言基础教程

（三级适用）

本系列教程编委会

电子科技大学出版社

微机汇编语言基础教程
（三级适用）
本系列教程编委会

计算机等级考试系列教程

微机汇编语言基础教程

（三级适用）

本系列教程编委会

主　　任　许宣伟
副 主 任　（以姓氏笔划为序）
　　王保强　许　远　何成彦　邵培基
本册主编　许　远
编　　委　（以姓氏笔划为序）
　　王保强　王占友　许　远　冯晓琳
　　何成彦　邵培基　陈荣红　陈周坤
　　闵显文　杨良杰　张仁津　梁　浩
　　[redacted]凯　彭小宁　焦连英　熊静琪

电子科技大学出版社
·1996·

内容简介

汇编语言是重要的程序设计语言，它是介于机器语言与高级语言之间的过渡性语言。本书深入浅出、循序渐进地介绍了汇编语言程序的基本结构、数据表示法、指令系统、伪指令、操作符和寻址方式、输入输出程序设计(主要是 BIOS 与 DOS 功能调用、显示器、键盘输入输出编程)，此外，还介绍了汇编语言中特有的语法内容，包括结构、记录、宏、堆栈。本书着重讲解了汇编语言程序设计的基本要素和高级的结构化程序设计方法，可作为扩大计算机专业人员和普通高等学校计算机基础教育的参考用书，也可作为计算机等级考试(三级)的参考教材。

计算机等级考试系列教程

微机汇编语言基础教程

本系列教程编委会

电子科技大学出版社出版
(成都建设北路二段四号) 邮编 610054
四川峨影印刷厂印刷
四川省新华书店经销

开本 787×1092 1/16 印张 19.5 字数 480 千字
版次 1996 年 6 月第一版 印次 1996 年 10 月第二次印刷
印数 3001～9000 册
中国标准书号 ISBN 7-81043-314-8/TP・115
定价：17.80 元

序

随着科学技术的迅猛发展，计算机已成为各个学科领域不可缺少的应用工具，计算机知识和应用能力已成为当代大学生知识和能力结构的一个重要组成部分，也是我国教育培养跨世纪人才最突出的需要加强的环节之一。目前在高校中普遍开展的计算机知识和应用能力等级考试正有效地推动这一目标的实现。同时，1993 年 12 月国家教委考试中心颁布的在全国进行计算机应用能力认证考试文件，将进一步推动全社会学习计算机、使用计算机的热潮。与此有关的教材和参考资料的需求与日俱增。

到目前为止，有关计算机应用等级考试的丛书为数不少，但是，这一套《计算机等级考试通用辅导教材》有让人耳目一新的感觉，它浅显易懂，循序渐进，深入浅出。全书除较系统地阐述计算机有关基础知识和上机操作外，还运用自问自答方式把初学者较易产生的疑难问题集中叙述，其讲解与前面已介绍的内容不相重复而又相互补充。每章中提供有读者自我检测题及答案，特别适合初学者又是自学为主的读者之学习要求。全书在培养读者上机操作能力方面的指导意义较为突出，书中收集的部份等级考试试题对有意参加有关等级考试的读者来说是一份有参考价值的资料。

综上所述，本书可作为非计算机专业读者学习（特别是自学）计算机知识和应用能力的培训教材或参考书。相信本书的出版将有助于推动计算机知识和应用的进一步普及，为我国全民族现代化素质的进一步提高有所裨益。

兰家隆　教授

1994. 5. 25 于成都

注：兰家隆，电子科技大学教授，四川省计算机等级考试委员会副主任。

续编暨修订说明

我们和电子科技大学出版社合作推出的《计算机等级考试通用辅导教材》系列丛书的前七册业已顺利出版，这七册分别是：《DOS 操作、文字处理及问题解答》、《BASIC 语言基础知识及问题解答》、《汉字 dBASE Ⅲ 基础知识及问题解答》、《电子表格技术基础知识及问题解答》、《PASCAL 语言基础知识及问题解答》、《C 语言基础知识及问题解答》、《FoxBase⁺ 基础知识及问题解答》。毋庸置疑，这套丛书收到了预期的效果，对计算机知识的普及尽了自己的绵薄之力，读者来信亦充分显示，本套丛书采用的编排体例是合乎读者自学需要的。

然而，计算机科学在不断地发展，计算机等级考试也以更加猛烈之势席卷神州大地，全国范围内的各系统、行业、组织的等级考试已有十余种之多，8086/8088 汇编语言等课程也列入了考试范围，而数据库技术的发展也导致了 FoxBase 与 FoxPro 列入等级考试的范畴。我们深知，跟上时代前进的步伐，保持这套丛书的全面性、时效性，是刻不容缓的。

没有必要掩饰本套丛书的不足与谬误之处，读者来信以及专家的评审都诚恳地指出了这一点，因此，在续编本套丛书的过程中，我们也对已出版的几种书中存在的不妥之处进行了必要的更正。

为此，《计算机等级考试通用辅导教材》一书的主编约请国内部分高等学校从事计算机等级考试教研一线工作的教师和一些对计算机普级教育有经验的同仁，共同对原书进行了系统的修改，并根据新的国家大纲充实了原书内容，考虑到本套丛书经过广泛试用，已经成熟，故第二版时将丛书名改为《计算机等级考试系列教程》并作为正式的大专教材向大专院校推广。参与本书修订工作的院校有电子科技大学、贵州师范大学、湖南怀化高等师范专科学样的老师，以及中国电子学会的计算机专家，本书修订再版暨续编的工作量大，虽然编委会做了大量细致的工作，但肯定还有不少谬误之处，欢迎广大读者多提意见，以利再版更正。

本书是根据计算机等级考试的形式发展而编写的。原丛书中没有此书，特此说明。

本系列教程编委会

1995 年 6 月

目　录

第3篇 汇编语言程序设计技术

第 4 篇　高级汇编语言技术

第1篇 汇编语言程序设计须知

汇编语言是一种常用的计算机编程语言，学习汇编语言的方法与学习一般的高级语言有所不同，它需要一些计算机学科的理论知识。本篇从汇编语言程序设计的基础知识：数的进制、字符的表示、计算机常用编码讲起，然后介绍汇编语言学习所需的硬件知识，接着认识汇编语言的概貌及汇编程序开发的过程，为以后的学习打下基础。

第1章 基础知识(I)

——数·字符·编码

本章学习要点

- 什么是进位计数制
- 二进制数的运算
- 十六进制数的加法运算
- 原码、补码、反码的互换
- 浮点数的表示
- 十进制数的二进制表示
- 字符编码

§1.1 数制及其转换

§1.1.1 什么是进位计数制

计算机的基本功能之一就是进行计算。大家知道，计算机由数量庞大的的电子元器件与集成电路组成，那么在这些设备中如何表示数字呢？这就涉及到二进制，它是计算机的数学基础。

数制有非进位计数制和进位计数制两种。

1. 非进位计数制

非进位计数制特点是：表示数值大小的数码与它在数中的位置无关。典型的非进位计数制是罗马数字。例如，罗马数字中：Ⅰ总是代表1，Ⅱ总是代表2，Ⅳ总是代表4，Ⅴ总是代表5，X总是代表10，C总代表100等等。

非进位计数制表示数据不便、运算困难，现已不常用。

2. 进位计数制

进位计数制的特点是：表示数值大小的数码与它在数中所处的位置有关。例如，十进制数123.45，数码1处于百位上，它代表$1\times10^2=100$，即1所处的位置具有10^2权；2处于十位上，它代表$2\times10^1=20$，即2所处的位置具有10^1权；其余类推，3代表$3\times10^0=3$，而4处于小数点后第一位，代表$4\times10^{-1}=0.4$，最低位5处于小数占后第二位，代表$5\times10^{-2}=0.5$，如此等等。

十进制运算中，凡是超过10就向高位进一位，相邻两位间是十倍的关系，10称为进位“基数”。可以想象：若是二进制，则进位基数应该是2，八进制进位基数为8，十六进制则进位基数应该是16。

十进制数共有0～9十个数码，十进制数就是由这十个数码及其它一些符号（小数点、正负号）组成。相应地，二进制数的数码个数为：0与1；八进制数有八个数码：0～7；十六进制数有16个数码：0～15（10～15分别由A～F表示）。

§1.1.2 十进制数与二进制数

人们习惯于使用十进制数0～9。逢十进一，借一当十，这完全是现在人们的习惯。其实，古埃及人与古巴比伦人就曾经使用过六十进制与十二进制。那么为什么在计算机中却偏偏采用古怪的二进制呢？

这是因为电子元器件最容易实现的是电路的通断、电位的高低、电极的正负，而在逻辑学中也常常用到二值逻辑，这都是因为两状态的系统具有稳定性（非此及彼），以及抗干扰性等。为了保证在计算机中进行数据传送，运行中不产生差错和减少计算机硬件的成本，都必须采用二进制。

二进制数只有“0”和“1”两个数码，而且由低位向高位进位时逢二进一。

像101,110,110.011等都是二进制数,但是以上三个数也可以认为是十进制数,为了表示它们的区别,可以给这些数字加上括号和下标,标明是几进制的数,例如:

$(101)_2$,$(110)_2$表示二进制的数;$(110)_{10}$,110表示十进制的数。

一个十进制数525,在十进制中说它是5个百、2个十、5个一的和,也就是:

$$525 = 500 + 20 + 5 = 5 \times 100 + 2 \times 10 + 5 \times 1 = 5 \times 10^2 + 2 \times 10^1 + 5 \times 10^0$$

又如:

$$123.45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$

所以任意的一个十进制数都可以表示成:

$$N = d_m \times 10^m + d_{m-1} \times 10^{m-1} + \cdots + d_0 \times 10^0 + d_{-1} \times 10^{-1} + d_{-2} \times 10^{-2} + \cdots + d_{-n} \times 10^{-n} = \sum_{i=-n}^{m} d_i 10^i \qquad (n,m \geqslant 0) \qquad (1\text{-}1)$$

上式中,$\sum$是求和符号;d_i表示各个位上的数字;m表示10的次幂。

对于第一个例子的十进制数525:

$n = 0, m = 2, d_2 = 5, d_1 = 2, d_0 = 5$;

对于第二个例子的十进制数123.45:

$n = 2, m = 2, d_2 = 1, d_1 = 2, d_3 = 3, d_{-1} = 4, d_{-2} = 5$。

这里我们把10叫做权,把式(1-1)叫做十进制数的按权展开式。基数实际上表明了每一位上可取的数字的个数,如10进制:每位上可以有0,1,2,…,9十个数字;二进制每一位上可以有0、1,两个数字。于是,我们可得到一个结论:对于任意r进制数,可能出现的数字是0,1,2,…,$r-1$,共r个。

把式(1-1)中的10用r来代替:

$$N = d_m \times r^m + d_{m-1} \times r^{m-1} + \cdots + d_0 \times r^0 + d_{-1} \times r^{-1} + d_{-2} \times r^{-2} + \cdots + d_{-n} \times r^{-n} = \sum_{i=-n}^{m} d_i r^i \qquad (m \geqslant 0, n \geqslant 0) \qquad (1\text{-}2)$$

式(1-2)是任意进制的按权展开式。取式中$r = 2$,那么每一位上可取的数字就只有0和1,这就是计算机中广泛使用的二进制数。对于二进制数我们可以把式(1-2)写成:

$$N_2 = b_m \times 2^m + b_{m-1} \times 2^{m-1} + \cdots b_0 \times 2^0 + b_{-1} \times 2^{-1} + b_{-2} \times b^{-2} + \cdots + b_{-n} \times b^{-n} = \sum_{i=-n}^{m} b_i 2^i \qquad (m,n \geqslant 0) \qquad (1\text{-}3)$$

那么,上面提到的几个二进制数可以表示成:

$$(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0; \quad (110)_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$(110.011)_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$$

事实上,每一个十进制数都能找到相对应的二进制数,一些简单数字的二进制和十进制对照见表1-1所示。

表1-1 十进制与二进制对照表

十进制	10	9	8	7	6	5	4	3	2	1	0	0.5	0.25	0.125	0.0625
二进制	1010	1001	1000	111	110	101	100	11	10	1	0	0.1	0.01	0.001	0.0001

§1.1.3 八进制与十六进制数

二进制的缺点是书写较长，不便于阅读，而八进制与十六进制书写容易，易读、易记，所以通常一些二进制代码都用八进制和十六进制来表示。

对于式(1-2)取 $r=8$，就得到八进制数的展开形式。八进制数有 0，1，2，3，…，7，八个数字，运算规则是“逢八进一，借一当八”。

对于十六进制数，按照式(1-2)取 r=16，就得到 16 进制数的展开形式。但是十六进制数要有十六个数字，而常用的阿拉伯数字只有 0～9 十个数字，另外的几个数字用 A～F 来表示，参见表 1-2。

表 1-2 给出八进制数、十进制数、二进制数、十六进制的对照表。

表 1-2 八进制数、十进制数和二进制数的对照表

十进制	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
二进制	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000
八进制	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20
十六进制	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10

与二进制数转换为十进制数的方法一样，八进制、十六进制的数都可以按照权展开的方法来转换为十进制数，例如：

$$(2B30)_{16}=2\times16^3+11\times16^2+3\times16^1+0\times16^0=(11056)_{10}$$

$$(24)_{16}=2\times16^1+4\times16^0=(36)_{10}$$

$$(81)_8=8\times8^1+1\times8^0=(65)_{10}$$

$$(134)_8=1\times8^2+3\times8^1+4\times8^0=(92)_{10}$$

八进制数可用括号加上下标来表示，如：$(123)_8$，$(376)_8$ 等，以便与 10 进制数区别开。

十六进制可以用相同的方法来表示，如：$(3FD)_{16}$，$(068E)_{16}$ 等。

§1.1.4 汇编语言中各种数的表示

由于十进制数的英文是“Decimal”，所以在汇编语言中用数字后加上英文“d”或“D”来表示，如：

126=(126)$_{10}$=126D=126d。

二进制数的英文是“Binary”，所以在汇编语言中也可用二进制数后加上“B”或“b”来表示二进制数，如：

(11000)$_2$=11000B=11000b

十六进制数可以在数字后加上“H”或“h”来表示，如：

(3FD)$_{16}$=3FDH=3FDh

同样，八进制数可以在数字后加上“Q”、“O”或“q”、“o”来表示，如：

(15)$_8$=13Q=13O=13q=13o

注意：在汇编语言中，所有数值常数必须以 0～9 之间的某个数字打头，特别是十六进制数的第一位若是 A～F，则应在前增加“0”。例如：值 FEBH 应写成 0FEBH，而非 FEB 或 FEBH。

§1.1.5 不同进制之间的转换

1. 二进制数转换成十进制数

二进制数据转换成十进制的方法很简单，只需按权展开然后相加即可，例如：

$$(1011.01)_2 = 1\times2^3+0\times2^3+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$$
$$= 8+0+2+1+0+0.25 = (11.025)_{10}$$

2. 十进制数转换成二进制数

整数部分和小数部分分别用不同方法进行转换。

(1)整数部分："除2倒取余法"

即将十进制数反复除以2，取其余数作为相应二进制数最低位 K_0，再除以2得余数 K_1，直到最后一次相除商为0时得到最高位 K_{n-1}，则：$K_{n-1}K_{n-2}\cdots K_1K_0$ 即为转换所得一进制数。

【例1-1】将$(121)_{10}$转换成二进制数，运算过程如下：

```
2 | 121   ……余 1(K0)        ↑ (低位)
2 |  60   ……余 0(K1)        |
2 |  30   ……余 0(K2)        |
2 |  15   ……余 1(K3)        |
2 |   7   ……余 1(K4)        |
2 |   3   ……余 1(K5)        |
2 |   1   ……余 1(K6)        ↓ (高位)
    0
```

故 $(121)_{10} = K_6K_5K_4K_3K_2K_1K_0 = (1111001)_2$

(2)小数部分："乘2取整法"

将十进制小数乘2，取乘积整数部分作为相应二进制数小数点后最高位 K_{-1}，反复乘2逐次得到 $K_{-2}, K_{-3}, \cdots, K_{-m}$。直到乘积的小数部分为0或小数位后的位数达到精度要求为止。

【例1-2】将$(0.8125)_{10}$转换。成二进制数。

```
   0.812 5
 ×      2
 ---------
   1.6250   ……整数 1(K-1)     ↑ (高位)
   0.6250                     |
 ×      2                     |
 ---------                    |
   1.2500   ……整数 1(K-2)     |
   0.2500                     |
 ×      2                     |
 ---------                    |
   0.5000   ……整数 0(K-3)     |
 ×      2                     |
 ---------                    |
   1.0000   ……整数 1(K-4)     ↓ (低位)
```

故 $(0.8125)_{10}=0.K_{-1}K_{-2}K_{-3}K_{-4}=(0.1101)_2$

【例 1-3】将$(25.25)_{10}$转换成二进制数。

对于这种既有整数又有小数的十进制数，可对其整数与小数部分分别转换成二进制数，再把两者连接起来。

$(25)_{10}=(11001)_2$；　$(0.25)_{10}=0.01)_2$

$(25.25)_{10}=(11001.01)_2$

3. 十进制数转换成其它进制的数

可将二进制数与十进制数的相互转换方法推广到其它数制与十进制数的互换，不同之处是应该考虑具体数制的进位基数。例如，八进制进位基数是8，十六进制基数是16，而转换算法完全是一样的，读者可参看图1-1。

- 八进制→十进制：以8为基数按权展开并相加
- 十进制→八进制
  - 整数部分：除8倒取余
  - 小数部分：乘8取整

(a)八进制数与十进制的互换

- 十六进制→十进制：以16为基数按权展开并相加
- 十进制→十六进制
  - 整数部分：除16倒取余
  - 小数部分：乘16取整

(b)十六进制数与十进制的互换

图 1-1

例如：

$(266)_{10}=(10A)_{16}$

$266\div16=16$ ……A

$16\div16=1$ ……0

$1\div16=0$ ……1

$(72)_{10}=(110)_8$

$72\div8=9$ ……0

$9\div8=1$ ……1

$1\div8=0$ ……1

4. 二进制数与八进制数、十六进制数之间的转换

前面已述及，由于八和十六都是二的整数倍，就使得二进制数与八进制、十六进制数之间的转换相对要容易得多。

显然，一位十六进制数需要用四位二进制数来表示，而一位八进制数要用三位二进制数来表示。

如：十六进制的A用二进制数表示是：1010；$(7)_8$用二进制数来表示是：111。

那么我们就可以用很简单的方法来实现二进制、八进制、十六进制之间的转换。

(1)将二进制数转化为十六进制数可以将该二进制数从低位起，每四位为一组，最高一组不足四位的前面用零补齐，分别对应一个十六制数字，将这些数字由低位向高位排列就得到该数的十六进制表示形式。

(2)将二进制数转换为八进制数可以将该二进制数从低位算起，每三位为一组，最高一组不足三位的，前面用零补齐，它们分别对应一个八进制数，将这些数字由低位向高位排列就得到该数的八进制表示形式。

例：

$(1110101101)_2=(3AD)_{16}$

0011,1010,1101

↓ ↓ ↓

3 A D

$(100000001000)_2=(808)_{16}$

1000,0000,1000

↓ ↓ ↓

8 0 8

$(1100101101)_2=(1455)_8$

001,100,101,101

↓ ↓ ↓ ↓

1 4 5 5

$(100000001000)_2=(4010)_8$

100,000,001,000

↓ ↓ ↓ ↓

4 0 1 0

相反地,要把一个十六进制数或八进制数转换为二进制数,可以把该十六进制数或八进制数的每一位分别用四位(或三位)二进制数来表示,不足四位时,前面应补零凑满位。

(3)将十六进制数转化为二进制数时每个十六进制数与四位二进制数与四位二进制数相对应,若不足四位时应在前面补零,这样就得到该十六进制数的二进制表示。

(4)将八进制数转化为二进制数时,每一个八进制与三位二进制数相对应,若不足三位应在前面补零,这样就得到该八进制数的二进制表示。

$(ABC)_{16}=(101011001101)_2$

A B C

↓ ↓ ↓

1010 1100 1101

$(87F)_{16}=(100001111111)_2$

8 7 F

↓ ↓ ↓

1000 0111 1111

$(321)_8=(011010001)_2$

3 2 1

↓ ↓ ↓

011 010 001

$(567)_8=(101110111)_2$

5 6 7

↓ ↓ ↓

101 110 111

§1.2 二进制与十六进制的运算

§1.2.1 二进制数如何运算

因为二进制数只由0、1两个数字,所以它的四则运算特别简单。其运算规则如表1-3(a)、表1~3(b)、表1-3(c)与表1-3(d)所示:

表1-3(a) 加法

+	0	1
0	0	1
1	1	10

表1-3(b) 减法

−	0	1
0	0	1
1	1	0

表1-3(c) 乘法

×	0	1
0	0	0
1	0	1

表1-3(d) 除法

/	0	1
0	无意义	0
1	无意义	1

对于加法运算，按“逢二进一”，作减法时；只要遵循“借一当二”的法则就行了；对于二进制数，由于二进制乘数与被除数中只有1和0两种情况，相乘运算要比十进制数相乘的“九九乘法表”法则简单多了。

【例1-4】加法与减法运算举例。

$(1011011)_2+(1010.11)_2=?$

```
  1011011
+)  1010.11
-----------
  1100101.11
```

$(1010110)_2-(1101.11)_2=?$

```
  1010110
-)  1101.11
-----------
  1001000.01
```

【例1-5】乘法与除法运算举例。

$(1011.01)_2\times(101)_2=?$

```
    1011.01
*)       101
---------------
    1011.01
   0000.00
+) 1011.01
---------------
  111000.01
```

$(100100.01)_2\div(101)_2=?$

```
          111.01
      ------------
101 ) 100100.01
--)   101
      ---
        1000
      -)  101
        ----
         110
       -) 101
         ---
          0101
           101
          ----
             0
```

由上式可见，二进制乘法可归结为“加法与移位”；二进制除除法运算可归结为“减法与移位”。做二进制除法的方法与做十进制除法的方法相同，在列竖式计算时，够除则在商上写1，不够除则写0，按此方法依次除下去，直到余数为零为止，在除不尽的情况下，可以根据需要计算到指定的精度即可。

§1.2.2　二进制数的逻辑运算

1. 什么是逻辑运算

逻辑，是指“条件”与“结论”之间的关系。因此，逻辑运算是指对“因果关系”进行分析的一种运算，运算结果并不表示数值大小，而是表示逻辑概念，成立还是不成立。

计算机中的逻辑关系是一种二值逻辑，二值逻辑很容易用二进制的“0”或“1”表示，例如：“真”与“假”、“是”与“否”、“成立”与“不成立”等。若干位二进制数组成的逻辑数据，位与位之间无“权”的内在联系，对两个逻辑数据进行运算时，每位之间相互独立，运算是按位进行的，不存在算术运算的进位与借位，运算结果也是逻辑数据。

2. 逻辑代数与逻辑变量

逻辑代数是实现逻辑运算的数学工具，由英国乔治·布尔(George Boole)于1894年首

先提出的，所以又称为布尔代数。

逻辑代数通过符号变量表示命题，研究命题及其条件的关系。符号变量又称为逻辑变量，常用英文字母 A、B、C……等表示。逻辑变量只有两种取值："真"和"假"，对应于二进制的 1 和 0。

逻辑代数是以逻辑变量为研究对象的，与普通代数有许多相似之处，有一套运算规则和基本定律，但与普通代数也有区别，主要在于逻辑代数演算的是逻辑关系，而普通代数演算的是数值关系。

3. 三种基本的逻辑关系

在逻辑代数中有三个基本的逻辑关系：**与、或、非**。其它复杂的逻辑关系均可由这三个基本逻辑关系组合而成。

(1)"与"逻辑

做一件事情取决于多种因素时，当且仅当所有因素都满足时才去做，否则就不做，这种因果关系称为"与"逻辑。用来表示和推演"与"逻辑关系的运算称为"与"运算，常用·、∧、∩或 AND 等运算符表示，"与"运算规则如表 1-2，两个二进制数进行与运算是按位进行的。

两个逻辑变量 a、b 进行与运算，在数学上可记为 F＝a AND b，F 是 A、B 的逻辑函数。对于 F＝a AND b，由"与"运算规则知：当且仅当 A＝1、B＝1 时，才有 F＝1，否则 F＝0。

(2)"或"逻辑

做一件事取决于多种因素时，只要其中有一个因素得到满足就去做，这种因果关系称"或"逻辑。"或"运算常用＋、∨、∪或 OR 等运算符表示，"或"运算规则如表 1-5，两个二进制数进行或运算是按位进行的。

(3)"非"逻辑

"非"逻辑实现逻辑否定，即进行"求反"运算，非运算规则见表 1-6。常在逻辑变量上面加一横线表示。例如，A 的"非"写成 $\overline{A}$。

表 1-4

a	b	a AND b
1	1	1
1	0	0
0	1	0
0	0	0

表 1-5

a	b	a OR b
1	1	1
1	0	1
0	1	1
0	0	0

表 1-6

a	NOT a
1	0
0	1

【例 1-6】二进制数的逻辑运算举例。

X＝10111001，Y＝11110011　　X＝10100001，Y＝10011011

求 X AND Y＝？　　求 X OR Y＝？

```
      10100001                10111001
OR)   10011011          AND)  11110011
-------------------     -------------------
      10111011                10110001
```

X＝10100001，求 NOT X＝？

NOT X ＝01011110