21世纪高等教育计算机规划教材

COMPUTER

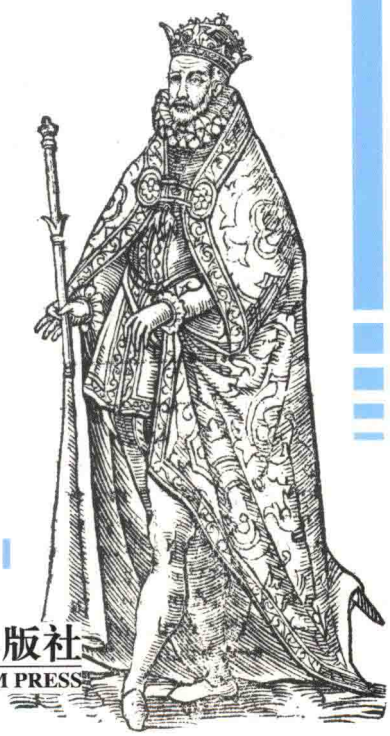
汇编语言程序设计（第2版）

A Guide for Assembly Language Programming (2nd Edition)

刘慧婷 王庆生 主编

陈洁 纪霞 钱付兰 徐怡 副主编

- 从指令执行看计算机工作原理
- 用程序实例训练和提高编程能力
- 以注重基础结合实验为编排指导



中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等教育计算机规划教材

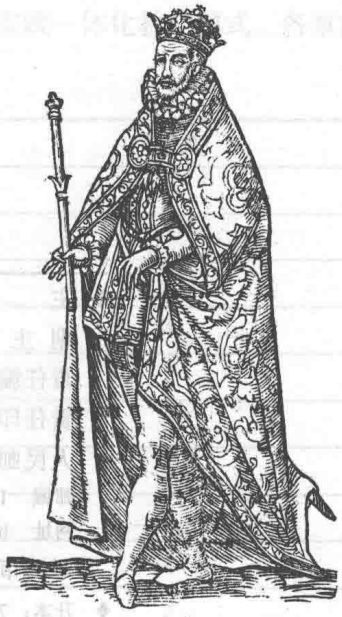
COMPUTER

汇编语言程序设计（第2版）

A Guide for Assembly Language Programming (2nd Edition)

刘慧婷 王庆生 主编

陈洁 纪霞 钱付兰 徐怡 副主编



人民邮电出版社

北京

图书在版编目（CIP）数据

汇编语言程序设计 / 刘慧婷，王庆生主编. -- 2版
. -- 北京 : 人民邮电出版社，2017.1
21世纪高等教育计算机规划教材
ISBN 978-7-115-44297-0

Ⅰ. ①汇… Ⅱ. ①刘… ②王… Ⅲ. ①汇编语言－程序设计－高等学校－教材 Ⅳ. ①TP313

中国版本图书馆CIP数据核字(2016)第314938号

内 容 提 要

本书系统地讲解了汇编语言程序设计的相关知识，全书共有11章，系统地论述了汇编语言基础知识，计算机基本原理，上机操作步骤，操作数的寻址方式，汇编语言的指令系统和伪指令，汇编语言中分支、循环和子程序的设计方法，宏指令，输入/输出和中断，并且还安排了实验部分。为了让读者能够及时地检查自己的学习效果，把握自己的学习进度，每章后面都附有丰富的习题。

本书既可以作为本科院校、高职高专各专业汇编语言课程的教材，也可以作为汇编语言培训或技术人员自学的参考资料。

◆ 主　　编　刘慧婷　王庆生
副 主 编　陈　洁　纪　霞　钱付兰　徐　怡
责任编辑　吴　婷
责任印制　沈　蓉　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市潮河印业有限公司印刷

◆ 开本：787×1092　1/16
印张：17.25　　2017年1月第2版
字数：454千字　　2017年1月河北第1次印刷

定价：45.00元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

第 2 版前言

汇编语言程序设计课程是计算机专业高等教育的专业基础必修课程，是集理论性与应用性为一体、软件与硬件相结合的课程。

编者于 2013 年所编写的《汇编语言程序设计教程》一书自出版以来，受到了众多高等院校的欢迎。为了更好地满足广大高等院校的学生对汇编语言程序设计方法学习的需要，编者结合近几年的教学改革实践和广大读者的反馈意见，在保留原书特色的基础上，对教材进行了全面的修订，这次修订的主要内容如下。

- 对本书第 1 版中存在的一些问题进行了校正和修改。
- 对不重要的内容进行了删减，重点更加突出。
- 增加了对书中知识点的解释，通过给例题增加运行情况图来加深读者对例题中知识点的理解。

在本书的修订过程中，作者始终贯彻让本书比上一版更加浅显易懂的原则。在每一章节都尽量采用图解的方式，让读者通过看例子在 Debug 环境下的运行图来理解例子中的相关知识点。修订后的教材，内容的叙述更加准确、通俗易懂和简明扼要，这样更有利于教师的教学和读者的自学。为了让读者能够在较短的时间内掌握教材的内容，及时地检查自己的学习效果，巩固和加深对所学知识的理解，每章后还附有习题。

全书参考总教学时数为 36 学时，建议采用理论实践一体化教学模式。各章的学时分配见下表。

项目	名称	学时数
第 1 章	汇编语言基础知识	2
第 2 章	计算机基本原理	2
第 3 章	汇编语言程序实例及上机操作	2
第 4 章	操作数的寻址方式	4
第 5 章	常用指令系统	8
第 6 章	伪指令与源程序格式	3
第 7 章	分支与循环程序设计	4
第 8 章	子程序设计	4
第 9 章	宏汇编及其他高级伪操作	2
第 10 章	输入/输出和中断	3
第 11 章	输入/输出应用	2
总计		36

全书由安徽大学刘慧婷、王庆生担任主编，安徽大学的陈洁、纪霞、钱付兰和徐怡（排名按姓氏字母排序）担任副主编。其中，第 1 章和第 8 章由刘慧婷编写，

第2章和第4章由钱付兰编写，第3章由王庆生编写，第5章和第10章由陈洁编写，第6章和第7章由徐怡编写，第9章和第11章由纪霞编写，全书由刘慧婷统稿和定稿。在此，向所有关心和支持本书出版的人表示衷心的感谢！

限于编者的学术水平，不妥之处在所难免，敬请专家、读者批评指正，来信请至htliu@ahu.edu.cn。

编者

2016年10月于安徽大学

目 录

第 1 章 汇编语言基础知识

通过本章的学习，认识汇编语言的意义，重点熟悉计算机中数据和字符的常用表示方法、补码的运算，为下一步学习汇编语言程序设计打下基础。

1.1 汇编语言简介

1.1.1 机器语言与汇编语言

计算机程序是由各种程序设计语言根据编程规则实现的，计算机程序设计语言经历了从低级到高级的发展，通常分为三类：机器语言（Machine Language）、汇编语言（Assembly Language）、高级语言（High Level Language）。

机器语言：计算机硬件直接识别的程序设计语言。构成这种程序的是机器指令，机器指令是用二进制编码的指令，即编码中只含二进制 0 或 1，如 111001100011 就是一条机器指令。由于计算机主要由数字电路构成，所以机器指令由计算机直接记忆、传输、识别和加工。

机器语言被称为第一代语言，不仅复杂难记，而且还依赖于具体的机型。可见程序编写难度极大，调试修改困难，无法在不同的机型间移植，如今早已没有人用机器语言写程序了。

汇编语言：一种面向机器的用符号表示的程序设计语言，所以也叫符号语言。和机器语言不同的是，汇编语言用直观、便于记忆和理解的英文单词或缩写符号来表示指令和数据变量，例如："MOV AX，VAL" 是一条传送指令，其中 MOV 是指令操作码，AX 是 CPU 中的寄存器，VAL 是一个变量的符号表示，指令表示将变量 VAL 的值传给 AX。所以汇编指令也叫符号指令，这些符号称为助记符。汇编指令集和伪指令集及其使用规则的统称就是汇编语言。汇编语言被称为第二代语言。

对于 MOV AX，VAL 这样的符号指令，比较简洁易读，但是计算机并不识别助记符，只能识别二进制编码的机器指令，因此需要通过一种翻译程序把汇编语言源程序翻译成机器码，才能提交计算机执行。这种翻译程序叫汇编程序，这种对汇编语言源程序的翻译过程简称汇编。汇编语言的出现，大大改善了编程条件，使更多的人可以进行程序设计了。

尽管用汇编语言编写的程序要比机器代码更容易理解，但每条汇编语言指令均对应一条机器指令，因而与机器语言并没有本质区别，因此汇编语言仍然属于面向机器的低级语言。

为了克服低级语言程序不好理解、编程调试困难，不易移植的弊端，人们迫切希望有一种近乎自然语言或数学表达形式的程序设计语言，使程序设计工作能避开与机器硬件相关，而着重于解决问题的算法本身，于是便产生了高级语言，例如 Basic、C、Java 等。高级语言被称为第三代语言。

用高级语言编写的源程序也必须经过编译和连接，将其转换为机器语言程序提交给计算机执行，或将其转换为一种中间代码，通过解释程序解释运行。

无论用什么语言编程，最终在计算机硬件中执行的程序都是由机器码组成的，因此汇编语言是离机器语言最近的。

1.1.2 汇编语言的组成

汇编语言由以下三类指令组成。

（1）汇编指令：机器码的助记符，有对应的机器码，它是汇编语言的核心。

（2）伪指令：没有对应的机器码，由编译器执行，计算机并不执行。

（3）其他符号：如+、-、*、/等，由编译器识别，没有对应的机器码。

1.1.3 为什么要学习汇编语言

高级语言易学好用，那为什么还要学习汇编语言呢?

（1）学习汇编语言对于从事计算机应用开发有重要作用。汇编语言程序是由符号指令写成的，本质上还是机器语言，与具体机型的硬件结构密切相关，可直接、有效地控制计算机硬件，运行速度快，程序短小精悍，占用内存容量少。在某些特定应用场合更能发挥作用，如实时控制系统，需要对硬件设备直接进行数据的输入/输出和控制，如在嵌入式系统和智能化仪器的开发中，需要更好地利用有限的硬软件资源，发挥硬件的功能。

（2）学习汇编语言是从根本上认识和理解计算机工作过程的最好方法，通过汇编语言指令，可以清楚地看到程序在计算机中如何一步步执行，有利于更深入理解计算机的工作原理和特点，单纯地介绍计算机的硬件知识或一门高级语言的程序设计是不可能做到这点的。汇编语言把软件和硬件紧密地结合在一起，起到连接硬件和软件的桥梁作用，掌握汇编语言对今后学习其他计算机相关课程非常有利。

1.2 计算机中数据的表示

1.2.1 不同进位计数制及其相互转换

1. 二进制数

进位计数制是一种计数方法，我们最熟悉的是十进制数，如423.5可表示为：

$423.5=4\times10^2+2\times10^1+3\times10^0+5\times10^{-1}$

注意到这里每位数字只能取0到9共10个数字符号，因此基数为10，逢10进1。不同位置上的数字代表的“权”是不同的，如百位4，该位的权值为10^2，第k位的权值为10^k。

计算机为便于数字电路对数据存储及计算的物理实现，采用二进制数。二进制数只有0和1两个数码，基数为2，逢2进1。不同位置上的数码代表的“权”不同，即各位权值为2^k。例如二进制数：$101101(B)=1\times2^5+1\times2^3+1\times2^2+1\times2^0=45(D)$。

可以看出，上面的二进制数按权展开就得到对应的十进制数。为便于明确计数制而不致误解，通常在二进制数后面加（B），在十进制数后面加（D）（也可用下标2和下标10），即：

101101(B)=45(D)，也可表示为101101B=45D，或$(101101)_2=(45)_{10}$

n位二进制数可以表示2^n个数，例如4位二进制数可以表示$2^4=16$个数，如表1-1所示。

表 1-1　　二进制与十进制

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

2. 十六进制数

用二进制数表示一个较大的数总是不太方便，为便于程序员表示数据，通常还采用十六进制。

十六进制数有 16 个数码，基数为 16，逢 16 进 1。第 k 位置上的数码代表的权值为 16^k。每位的数码作如下规定：0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F，共 16 个数码。其中 A，B，C，D，E，F 分别表示十进制的 10，11，12，13，14，15。十六进制数后面要加上 H 以示区别。

例如十六进制数：

$5FH = 5\times16^1+15\times16^0=80+15=95D$

显然十六进制表示比二进制表示来得简洁，该数用二进制表示则为：

5FH =01011111B

注意到一位十六进制数用四位二进制数表示即可，反之亦然。可见二进制数与十六进制数有着简单直接的互相转换关系，这是因为十六进制数的基数是 2 的幂。不仅如此，二进制数与 2^k 进制数都可以简单直接地互相转换。

实际生活中存在多种计数制，如计时采用时分秒就是 60 进制。其道理都是一样的。

3. 二进制数、十六进制数转换为十进制数

如前所述，各位二进制数乘以对应的权之和即得到十进制数。如：

例 1.1　$N=101101.1B = 1\times2^5+1\times2^3+1\times2^2+1\times2^0+1\times2^{-1}=45.5D$

各位十六进制数乘以对应的权之和即得到十进制数。如：

例 1.2　$N=5FH = 5\times16^1+15\times16^0=80+15=95D$

4. 十进制数转换为二进制数

这里介绍常用的两种方法。

（1）降幂法（适用于数值不大的数），降幂法就是先写出小于此数的各位二进制权值，然后再求和。

例 1.3　求 N=13.5D 的二进制数。小于此数的各位二进制权值为：

8　　4　　2　　1　　0.5

显然应选 8，再选 4，而不能选 2（因为 8+4+2=14），再选 1，最后选 0.5，所以：

13.5D=8+4+1+0.5 =1101.1B

```
  1000
  0100
  0001
+   0.1
------
  1101.1
```

（2）除法（又叫除 2 取余法，仅适用于整数部分），除 2 取余法就是不断除以 2，记下余数，

直到商为0为止。

例 1.4 求 N=13D 的二进制数。

13/2=6　　余 1　(b_0)

6/2=3　　余 0　(b_1)

3/2=1　　余 1　(b_2)

1/2=0　　余 1　(b_3)

13D= $b_3b_2b_1b_0$ =1101B

特别注意，对于二进制数的小数部分除了用降幂法也可采用乘法，即不断乘以 2，并记下整数，而小数部分再乘以 2，直到结果的小数部分为 0 为止。注意：并非所有的十进制小数都能用二进制完全表示，如小数 0.3，这时按实际需要取一定精度表示即可。

例 1.5 求 N=0.625D 的二进制数。

$0.625\times2=1.25$　　($b_{-1}=1$)

$0.25\times2=0.5$　　($b_{-2}=0$)

$0.5\times2=1.0$　　($b_{-3}=1$)

N=0.625D= $b_{-1}b_{-2}b_{-3}$ =0.101B

5. 十进制数转换为十六进制数

和十进制数转换为二进制数类似，也有降幂法和除法。

（1）降幂法（适用于数值不大的数），降幂法就是先写出小于此数的各位十六进制数权值，然后再求和。

例 1.6 求 N=95D 的十六进制数。小于此数的各位十六进制权值为：

16　　1

显然应选 16×5，再选 1×F，所以

N=95D=80+15=16×5+1×F=5FH

（2）除法（又叫除 16 取余法，仅适用于整数部分），除 16 取余法就是不断除以 16，记下余数，直到商为 0 为止。

例 1.7 求 N=95D 的十六进制数。

95/16=5　　余 15 (h_0)

5/16=0　　余 5　(h_1)

N=95D= h_1h_0 =5FH

同样特别注意，对于十进制数的小数部分除了用降幂法也可采用乘法，即不断乘以 16，并记下整数，而小数部分再乘以 16，直到结果的小数部分为 0 为止。不再举例。

6. 二进制数和十六进制数的相互转换

由于十六进制数的基数 $16=2^4$，故一位十六进制数由四位二进制数组成，相互转换极为简单。

例 1.8 N=1011111.11(B)=01011111.1100(B)=5F.C(H)

注意到从二进制数转换到十六进制数时，二进制数的整数部分从最低位开始每 4 位一组，不足 4 位的，高位补 0 补足 4 位。小数部分从最高位开始每 4 位一组，不足 4 位的，低位补 0 补足 4 位。

1.2.2 二进制数和十六进制数的运算

1. 二进制数的运算规则

加法规则：0+0=0，　0+1=1，　1+1=0　（进位 1）

乘法规则：0×0=0，　0×1=0，　1×1=1

2. 十六进制数的运算

十六进制数的加减运算只要遵循逢 16 进 1 规则即可，当然也可先把十六进制数转换为二进制数，运算后的结果再转换为十六进制数。

例 1.9

```
    43A5
+   5A34
--------
    9DD9
```

例 1.10

```
    5A34
-   43A5
--------
    168F
```

十六进制数的乘、除法运算可以先把十六进制数转换为十进制数，运算后的结果再转换为十六进制数。十六进制数的乘法也可以用十进制数的乘法规则计算，但结果用十六进制数表示。

例 1.11

```
    2A34
×   0025
--------
    D304
+   5468
--------
    61984(H)
```

1.2.3　带符号数的补码表示

数分为正数和负数，计算机中的数是用二进制来表示的，数的符号也用二进制来表示，所谓带符号数就是最高位是符号位，一般规定正数的符号位为 0，负数的符号位为 1。把一个数连同其符号在内数值化表示叫机器数，机器数的表示可以用不同的码制，常用的有原码、补码、反码。这里只介绍最常用的补码。

补码表示法中的正数用符号位+绝对值表示，即数的最高位为 0，其余部分为该数的绝对值。例如，用 8 位二进制来表示，

$[+1]_{补}$=00000001，$[+127]_{补}$=01111111，$[+0]_{补}$=00000000。

负数用补码表示时，方法是对其正数各位取反，然后最低位加 1。我们把这种对二进制数取反加 1 的运算叫作求补运算。负数用补码表示时，其符号位必定为 1。

例 1.12　用 8 位二进制来表示，求$[-3]_{补}$。

先写出+3：　　0000 0011

各位取反为：　1111 1100

最低位加 1 为：1111 1101

$[-3]_{补}$=1111 1101，或用十六进制表示，$[-3]_{补}$=FDH。

读者也许会问，如何用 16 位二进制来表示$[-3]_{补}$呢？其实只要在刚求出的$[-3]_{补}$的前面加上 8 个 1 就可以了，即$[-3]_{补}$=1111 1111 1111 1101 ，或$[-3]_{补}$=FFFDH。这叫符号扩展。对负数的符号扩展只需在前面补 1，对正数的符号扩展只需在前面补 0，符号扩展并没有改变数的大小，只是改变

了位数。读者可自行验证。

我们已经知道对负数用补码表示时，方法是对其正数取反加 1，即作求补运算。其实这个方法是根据补码定义得出的。下面给出证明。

补码定义：

（X≥0 时）　$[X]_{补}$=符号+|X|　　---（1）

（X＜0 时）　$[X]_{补}=2^n-|X|$　　---（2）

现在我们把（2）式进一步改写，

$[X]_{补}=2^n-|X|=(2^n-1-|X|)+1$　　---（3）

请注意（3）式右边括号中的$(2^n-1-|X|)$，就是对|X|取反码。再+1 就得到$[X]_{补}$。由此可见，求负数的补码，方法是对其正数取反加 1。

现在我们把（3）式进一步改写，

$|X|=2^n-[X]_{补}=(2^n-1-[X]_{补})+1$　　---（4）

注意到上式右边的$(2^n-1-[X]_{补})+1$，就是对$[X]_{补}$再求补码，就得到|X|。由（3）式和（4）式说明了以下结论：

$$[X]_{补} \xleftrightarrow{\text{求补}} [-X]_{补}$$

例 1.13　依据补码定义写出以下各数的补码，以 8 位二进制表示。

$[-1]_{补}=2^8-1=1\ 0000\ 0000-1=1111\ 1111$，直接由（2）式得到。

$[-127]_{补}=2^8-127=(2^8-1-127)+1$

$=(1111\ 1111-0111\ 1111)+1$

$=1000\ 0000+1$

$=1000\ 0001$

依据补码定义求一个数的补码表示，有些繁琐，用取反加 1 的规则更为简便。

例 1.14　识别以下各数的十进制值。

$[a]_{补}$=1111 1111，求补后为 0000 0001 = $[1]_{补}$，所以，$a=-1$

$[b]_{补}$=1000 0000，求补后为 1000 0000 = $[128]_{补}$，所以，$b=-128$

$[c]_{补}$=1000 0001，求补后为 0111 1111 = $[127]_{补}$，所以，$c=-127$

前面我们都是以 8 位二进制数讨论，8 位二进制数可以表示 2^8=256 个数，当它们是补码表示的数时，所能表示的数值的范围是$-128 \leqslant N \leqslant +127$。

1.2.4　补码的加法和减法

计算机中主要是用补码表示数据，因此我们应关注补码的加法和减法。

补码的加法规则是：

$[X+Y]_{补}=[X]_{补}+[Y]_{补}$　　----（5）

补码的减法规则是：

$[X-Y]_{补}=[X]_{补}+[-Y]_{补}$　　----（6）

这个规则说明了用 2 个数的补码相加就可以完成 2 个数的加减法，得到的还是补码。

下面给出证明：

需明确 X>0，Y>0，先看（5）式，由补码定义可知，$[X+Y]_{补}=X+Y=[X]_{补}+[Y]_{补}$

（5）式得证。再看（6）式：

如果[X−Y]≥0，由补码定义可知，（6）式左边应有$[X-Y]_{补}=X-Y$，

而(6)式右边=X+（2^n−Y）=X−Y+2^n =X−Y

如果[X−Y]<0，（或者说 Y>X），

应有$[X-Y]_{补}=2^n-|X-Y|=2^n-(Y-X)=2^n-Y+X=[-Y]_{补}+[X]_{补}$

（6）式得证。

下面给出例子说明补码的加法运算。

例 1.15　8 位补码的加法运算。

十进制	二进制
25	0001 1001
+（−32）	+ 1110 0000
−7	1111 1001
32	0010 0000
+（−25）	+ 1110 0111
7	0000 0111
	1↙

从例中可以看出补码的加法很简便，不必考虑数的正负，符号位参与运算即可，计算结果都是正确的。从最高有效位向高位的进位由于机器字长的限制而自动丢弃，但结果依然正确。同时这个进位被保存到机器中的标志寄存器中，其作用以后再说明。

1.2.5　无符号数的表示

如果要处理的数全是正数，保留符号位就没有必要，我们可以把最高有效位也作为数值，这样的数就叫无符号数。用 8 位二进制来表示的无符号数的数值范围是 0≤*N*≤255，用 16 位二进制来表示的无符号数的数值范围是 0≤*N*≤65535。在计算机中最常见的无符号数是表示内存单元的地址。例如：1100 0010B=C2H=194D，而不再表示一个负数。

1.2.6　字符的表示

除了数值以外，人们有时还需要用计算机处理字符或字符串。例如，从键盘输入或打印输出信息都是以字符方式输入/输出的。字符包括：

字母：A，B，C，D……

数字：0，1，2，3……

专门符号：+，−，×，/，SP（space 空格）……

非打印字符：CR（carriage return 回车），LF（line feed 换行）……

这些字符必须采用二进制的编码方式，目前采用最常用的美国信息交换标准代码 ASCII（American Standard Code for Information Interchange）来表示。

这种代码用一个字节（8 位二进制码）来表示一个字符，其中低 7 位为字符的 ASCII 值，故能表示 128 个符号和代码，最高位一般用作检测校验位，见表 1-2。

为了能表示更多的符号，将 7 位 ASCII 码扩充到 8 位，就可以表示 256 个符号和代码，称为扩充的 ASCII 码。

表 1-2　　部分常用字符的 7 位 ASCII 码表（十六进制表示）

字符	ASCII 码	字符	ASCII 码	字符	ASCII 码
NUL	00	A	41	a	61
BEL	07	B	42	b	62
LF	0A	C	43	c	63
FF	0C	D	44	d	64
CR	0D	E	45	e	65
SP	20	F	46	f	66
#	23	G	47	g	67
$	24	H	48	h	68
%	25	I	49	i	69
0	30	J	4A	J	6A
1	31	K	4B	K	6B
2	32	L	4C	L	6C
3	33	M	4D	m	6D
4	34	N	4E	N	6E
5	35	O	4F	O	6F
6	36	P	50	P	70
7	37	Q	51	Q	71
8	38	R	52	R	72
9	39	S	53	S	73
:	3A	T	54	T	74
;	3B	U	55	U	75
<	3C	V	56	V	76
=	3D	W	57	w	77
>	3E	X	58	X	78
?	3F	Y	59	Y	79
@	40	Z	5A	Z	7A

1.2.7　基本逻辑运算

四种基本逻辑运算见表 1-3。

表 1-3　　四种基本逻辑运算

A	B	AND	OR	XOR	NOT A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

（1）“与”运算（AND），又叫逻辑乘，可用符号·或∧来表示，只有当逻辑变量 A、B 都为 1 时，“与”运算的结果才为 1。

（2）“或”运算（OR），又叫逻辑加，可用符号+或∨来表示，只要变量 A、B 其中有一个为

1 时，“或”运算的结果就为 1。

（3）“异或”运算（XOR），可用符号∀来表示，只有当变量 A、B 中仅一个为 1 时，“异或”运算的结果才为 1。

（4）“非”运算（NOT），对变量 A 取反，即如果 A=1，则 $\overline{A}$=0，反之亦然。

逻辑运算都是按位操作，例如，X=0011，Y=1011，均为二进制数，则有：

X（AND）Y=0011，X（OR）Y=1011，X（XOR）Y=1000，（NOT）X=1100

本章小结

本章介绍了汇编语言的组成，计算机中数和符号的表示方式，补码的加减运算和逻辑运算的规则，为后面章节的学习打下基础。

习题 1

1.1　什么是机器语言？什么是汇编语言？简述汇编语言的特点。

1.2　汇编程序与汇编源程序的区别是什么？

1.3　把下列十进制数转换为二进制数和十六进制数。

（1）67　　（2）34　　（3）254　　（4）123

1.4　把下列二进制数转换为十六进制数和十进制数。

（1）01101101　　（2）10110010　　（3）111111

1.5　作下列十六进制数的运算。

（1）5A+64　　（2）86-49　　（3）123-9A　　（4）43×2B

1.6　根据补码定义把下列十进制数表示为 8 位二进制补码。

（1）64　　（2）-24