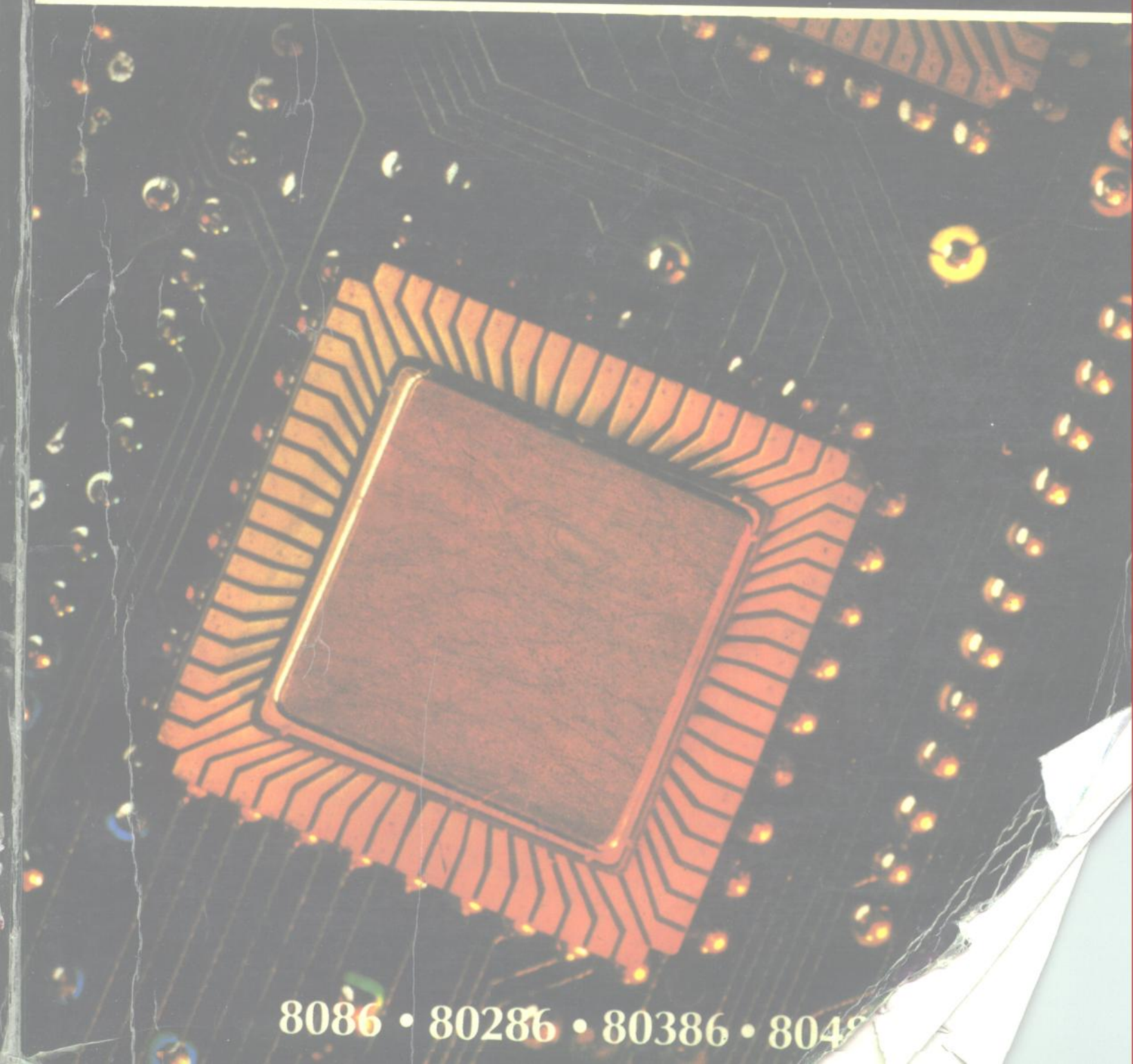
微处理器与接口技术

——程序设计和硬件

[美]Douglas V. Hall 著　　领衔翻译　赵振西

第二版

中国科学技术大学出版社



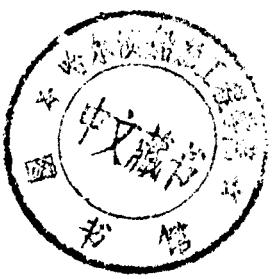
8086 • 80286 • 80386 • 804

微处理器与接口技术

——程序设计和硬件

[美]Douglas V. Hall 著

领衔翻译　赵振西



中国科学技术大学出版社

1994·合肥

(皖)新登字 08 号

图书在版编目(CIP)数据

微处理器与接口技术——程序设计和硬件/[美]Douglas V. Hall 著;赵振西 等译。—合肥:中国科学技术大学出版社,1994 年 3 月
ISBN 7-312-00571-3

Ⅰ 微处理器与接口技术——程序设计和硬件
Ⅱ [美]Douglas V. Hall 著;赵振西 等译
Ⅲ ①微处理器 ②程序设计 ③计算机接口技术 ④硬件
Ⅳ TP

凡购买中国科大版图书,如有白页、缺页、倒页者,由本社出版部负责调换

中国科学技术大学出版社出版发行
(安徽省合肥市金寨路 96 号,邮编:230026)
安徽省广告公司印刷厂排版
中国科学技术大学印刷厂印刷
全国新华书店经销

开本:880×1230/16　　印张:37.5　　字数:1238 千
1994 年 3 月第 1 版　　1994 年 3 月第 1 次印刷
印数:1—4900 册
ISBN7-312-00571-3/TP・89　　定价:42.50 元

译 者 序

本书译自美国 Macmillan/McGraw-Hill 出版公司 1992 年出版的 "MICROPROCESSORS AND INTERFACING: Programming and Hardware" (Second Edition)。该书结合 8086/80186/80286/80386/80486 微处理器家族，全面、系统和深入地论述了微型计算机与接口技术。由于作者道格拉斯 V. 霍尔知识面宽，在微型机及其应用方面有丰富的教学与工程实践经验，该书结构合理、内容充实，并成功地应用了软件与硬件相结合的叙述方法，是一本优秀的教材和技术参考书。由于该书各章有较好的独立性，通过适当的拼合与裁剪，便可用做《微型计算机原理》、《8086 汇编语言程序设计》和《微机接口技术》等课程的教材。关于该书的教学使用方法，请参考作者在序言中所提出的建议。

在翻译过程中，译者发现原书中有个别技术性错误，为此已在译文中做了更正和注释。原书附有英文索引，由于篇幅限制和处理上的困难，译本未附中文索引。

由于该书篇幅甚大和出版计划的紧促，故由八位同志参加了翻译工作：伍传平翻译了第 1 章，方祥翻译了第 3，7，8，14 章，张曙翻译了第 4，5，6，11 章，周建民翻译了第 10 章，岳丽华翻译了第 12 章，杨力翻译了第 13 章，李京翻译了第 15 章，赵振西翻译了序言和第 2，9 章并审校了全书译稿。

由于水平所限，译文中错误在所难免，敬请读者指正。

赵振西

1994 年 1 月

原　　序

本书是为多种类型的介绍微处理器的课程而编写的，仅需要读者具有二极管、晶体三极管和简单的数字器件等预备知识。

作为一名工程师和教师，我的经验表明，首先彻底地学好一种微处理器系列，需要时再以此牢固的基础去学习其他的系列是很有效的方法。对于本书我选择了 Intel 8086/80186/80286/80386/80486 微处理器系列。该系列的器件已被用于数以百万计的个人计算机中，包括 IBM PC/AT、IBM PS/2 以及许多“变种”机型。8086 是该系列的第一个成员，虽然它已被新型处理器所取代，但它依然是学习微处理器极好的起点。在学习多用户/多任务系统之前，读者无须知道新型微处理器的先进性能。因此在第 15 章之前，8086 被用作硬件和程序设计的主要示例。第 15 章将讨论新型微处理器的性能以及如何将这些性能用于多用户/多任务系统中。

本书的内容与组织

本书各章均以基本目的要求开头，以重要术语和概念的复习结束。每一章最后都给出了丰富的、用来强化该章中的理论和应用的练习题。

为了帮助读者重温预备知识，第 1 章中安排了对数字概念的一个简短的复习，这些概念是本书其余部分所需要的。该章还包括关于基本计算机数学以及二进制、十六进制和二—十进制算术运算的综述。

第 2—10 章

第 2—10 章向读者提供了对微处理器的综合性介绍，包括中断应用、数字与模拟接口以及工业控制。这些章节包括 8086 微处理器系列的综述与体系结构、程序设计语言、系统连接以及故障检测。

由于我是经由了真空管阶段而进入电子学领域的，所以我教授微处理器初始的倾向是从硬件方面来讲解。然而，越多地从事使用微处理器的设计工作和越多地从事微处理器的教学工作，就越意识到：学习微处理器真正的要害在于能够对其编程以使其工作。因此，第 2—5 章向读者介绍了如何为 8086 微处理器编写结构化的汇编语言程序。在这一个程序设计部分所采用的方法是：分解问题，写出求解的算法，进而将该算法简单地翻译成汇编语言。经验表明，对于一个工作程序的生产，这种方法较之只写出汇编语言指令的方法要可信得多。在第 2—5 章中仅根据解决简单程序设计问题的需要对 8086 的指令系统做了介绍，而第六章则给出了一个包括所有 8086 指令及其示例的字典，以备参考。

第 7 章讨论了信号、定时以及一个简单的基于 8086 的微型计算机的系统连接。该章中还讨论了对一台发生故障的基于 8086 的微型计算机进行故障检测的系统方法，并且介绍了使用逻辑分析仪来观测微型计算机总线信号的方法。第 8 章讨论 8086 如何响应中断，怎样写中断服务过程，以及一种称为中断优先权控制器的外围器件的操作。

第 9 章和第 10 章说明微处理器是怎样和多种低级输入与输出设备进行接口的。第 9 章说明微处理器是怎样与键盘、显示器以及继电器之类的数字设备接口的。第 10 章说明微处理器是怎样与 A/D、D/A 以及各种传感器之类的模拟量输入/输出设备进行接口的。第 10 章还说明如何将所有这些部件装配在一起来形成一个基于微处理器的磅秤和一个简单的基于微处理器的过程控制系统的。第 10 章以讨论怎样用微处理器来实现数字滤波器而结束。

第 11—15 章

第 11—15 章用来介绍诸如 IBM PC 和 IBM PS/2 系列的微型计算机的硬件、软件和外围接口。第 11 章讨论主机板线路，包括 DRAM 系统、高速缓冲存储器、数学协处理器和外围接口总线。该章还说明怎样利用图形获取程序来画图，怎样用一个仿真程序来修改设计的逻辑和时序，以及怎样使用布线程序来设计系统的印刷线路板。这些电子设计自动化工具的知识对于每个开发高速微处理器系统的人是必不可少的。

应许多来自工业界的专家的要求，第 12 章向读者介绍程序设计语言 C，该语言已被用来编写大量的系统级程序。这一章受益于这样的事实，即当读者已经熟悉了 8086 型的汇编语言时，学习 C 语言是很容易的。该章中的一节还说明怎样编写既包括 C 语言模块又包括汇编语言模块的简单程序。

第 13 章叙述了诸如 CRT 显示器、磁盘和打印机等外围设备的操作和接口。第 14 章说明怎样使微型计算机与调制解调器和网络等通信系统接口。

最后，第 15 章从讨论多用户/多任务操作系统必然提出的需求开始，进而描述了 80286、80386 和 80486 微处理器的保护方式性能是如何满足这些需求的。本书的这一部分还讨论了在多种环境下怎样为 80386 开发程序。本章和本书以介绍并行处理器、神经网络和模糊逻辑来结束。我想读者会像我一样被这些新发展的领域所吸引。

教学安排建议

灵活的组织

该课本的内容是广泛而又可灵活组织的。如果学生在基本二进制数学和数字技术原理方面具有坚实的基础，则可以很容易地删略第 1 章。

第 2—10 章

我建议将第 2—10 章作为一个教学单元，因为其中的每一章都是依赖于前一章的。这九章作为微处理器的简明教程具有理想的概括性。其余各章给教员一个机会，以便剪裁学生所需要的内容，或者供个别的学生用来进一步研究微处理器结构的最新发展。

第 11 章

在学生获取供设计基于计算机的系统所用工具的知识时，可以从第 11 章中选学个别的专题。DRAM 这一节是十分重要的。

第 12 章

对于新增加的关于 C 语言程序设计的第 12 章，可以讲解，或者留作课外阅读。读者至少也应仔细察看其中简单程序设计的示例以及 C 工具的开发。如果课堂时间不允许安排这一章，则可在讲稿中使用精选的例子和程序。这一章应该纳入任何没有单独开设 C 语言程序课的教学中。

第 13 章

必要时可以依照课程大纲来讲解外围设备这一章的有关部分。CRT、磁盘和打印机这三节是着重推荐的。

第 14 章

这是很重要的一章，其中给出了数据通信不断发展着的应用。应该尽可能地讲解这一章，除非教学中单设了数据通信课程。最重要的是调制解调器和局域网络这两节。

第 15 章

最后一章是新型微处理器发展的前沿。我希望所有学生都有机会学习这一章。至少，学生们应该读一读关于 386 的一节。这是最后的一章，然而这仅是他们研究微处理器的开端。

本版中的新特点

根据来自工业界和各类电子学教师的反馈意见，《微处理器与接口技术：程序设计和硬件》一书的第二版含有如下新的或改进的特点：

1. 根据教师们的意见，第 4 章和第 5 章中各专题的安排顺序做了改进；
2. 第 10 章中增加了充分阐述数字信号处理硬件和软件的一节；
3. 第 11 章中的一节叙述并举例说明了怎样使用电子设计自动化工具来开发微型计算机系统的硬件，这些工具包括图形获取程序、仿真程序和 PC 插件板布线程序；
4. 应工业界建议者的要求，增加了全新的第 12 章，其中完整地介绍了程序设计语言 C，包括含有 C 语言和汇编语言两种模块的程序示例；
5. 第 13、14 章，即介绍系统外围设备的两章，经修改反映了在 VGA 图形学、光盘存储器、激光打印机和数字视频交互等技术方面的进展。这两章现在包括汇编和 C 两种语言的接口程序举例；
6. 第 14 章中的网络部分经扩充反映了流行的重要网络；
7. 现在的第 15 章包含了对 386 和 486 微处理器性能的大量描述，并且讨论了这些性能怎样应用于诸如 Microsoft 公司的 OS/2 和 Windows 3.0 等多任务环境中；
8. 第 15 章中还增加了对神经网络计算机和模糊逻辑的介绍。

实验手册与教师手册

本书和与其配套的实验手册含有大量硬件和软件

方面的练习题，学生们做这些练习题能巩固他们在微处理器方面的知识。可以用 IBM PC 或 IBM PC 兼容计算机来对许多 8086 汇编语言程序进行编辑、汇编、链接/定位、运行和调试。

实验手册包括 40 个直接与课文配合的实验练习。每一个实验都包括本章参考书、所需设备、实验目的和实验过程。

教师手册有复习题的答案，还包括实验提示以及实验手册中精选问题的答案。

教师手册包括两张磁盘指南，其中含有课文和实验手册中所有程序的源代码。

其他目的

本书的主要目的之一是教读者学会怎样看懂微处理器和外围器件产品的数据表，因此该书含有许多数据表的有关部分。因为书中讨论了大量器件，因而，不可能包括全部数据表。如果想进行深入的研究，建议学习或查阅《Intel Microprocessors and Peripherals Handbooks》的最新版本。专科学校和大学可以从 Intel 公司大学关系部（Academic Relations Department of Intel）免费得到这些手册。本书末尾的参考书目中含有其他一些书和期刊的目录，以供读者查阅书中所讨论的各专题的更多细节。

致谢

我深切地感谢我周围的人们，他们帮助我将这本书付诸实现。感谢 Pat Hunter，她热切的鼓励和帮助使我完成了看来似乎无穷尽的细节。她校对和整理了原稿，对每章末尾的习题都求出了答案以证明是可解的。她所提出的建议和所做的贡献是难于一一列举的；感谢新英格兰理工学院康奈狄格州新不列颠分校的 Richard Cihkey，他仔细地审阅了原稿，并且提出了许多宝贵的意见；感谢 Instant Information 有限公司的 Mike Olisewsky，他帮助我确立了编写第 12 章的恰当的思路，并且提供了应在本书中反映的工业方面的见解；感谢波特兰州立大学的 Michael A. Driscoll 博士，他帮助我仔细斟酌了第 15 章的内容；感谢 Intel 公司允许我使用他们的参考手册中的许多图表，从而使本书能引导读者熟悉真实的数据手册。最后，我要感谢我的妻子 Rosemary、我的孩子 Linda、Brad、Mark、Lee 和 Kathryn 以及我的家庭的其他成员，感谢他们在重写这本书的漫长工作中对我的关心和支持。

如果你对改进本书有什么建议，或者阐明其他某个人的观点，请通过出版公司与我联系。

Douglas V. Hall

目　　次

第一章　计算机数制、代码和数字器件

在开始讨论微处理器与微型计算机之前，需要澄清用于微型计算机中的数制、代码和数字器件的一些关键概念，读者对这些概念可能是生疏的。本章简述了这些概念，如果这还不足以使你对这些概念有深刻的印象，那么请你参阅《数字电路与数字系统》的有关章节。该书由 McGraw-Hill 出版公司于 1989 年在本书之前出版。

目的要求

学完本章，应能做到：

1. 进行二进制码、十六进制码和 BCD 码间数的转换。
2. 定义术语：位，半字节，字节，字，最高有效位和最低有效位。
3. 查表找出给定字母数字字符的 ASCII 码或 EBCDIC 码。
4. 进行二进制数、十六进制数和 BCD 数的加减运算。
5. 描述门、触发器、锁存器、寄存器、ROM、PLA、动态 RAM、静态 RAM 和总线的操作。
6. 说明怎样使算术逻辑运算部件执行二进制数的算术或逻辑运算。

1.1　计算机的数制与编码

十进制复习

为了理解二进制数制的结构，首先复习一下人们熟悉的十进制数制。这是一种基数为 10 的数制。下面有一个十进制数，其每位的位值均被表示为 10 的幂

$$\begin{array}{cccccccc} 5 & 3 & 4 & 6. & & 7 & 2 \\ 10^3 & 10^2 & 10^1 & 10^0 & & 10^{-1} & 10^{-2} \end{array}$$

因此，由十进制数 5346.72 的各位可知，它有 5 个千、3 个百、4 个十、6 个一、7 个十分之一和 2 个百分之一。任一数制所需的符号数与其基数相等，因而十进制中有 10 个符号，即 0～9。任一位的计数超过最大符号值时，该位就回到 0 并且使下一较高位增加 1。汽车里程表就是这样一个很好的例子。

用任一数的幂作为数位的位值均能构成一种数制，但是有些基数更有用。制作一种能存储和处理 10 个不同电平的电子线路是困难的，而制作一种能处理两个电平的电子线路却是相对容易的。因此，在数字系统中惯于用**二进制**即**基数为** 2 的数制来表示数。

二进制

图 1-1 (a) 列出了二进制数每位的位值。每个二进制位表示一个 2 的幂。常把二进制位称为**位**或**比特**(bit)。二进制小数点右边的数位表示小于 1 的小数。二进制仅用两个符号，即 0 和 1，因此二进制计数为 0，1，10，11，100，101，110，111，1000 等。图 1-1 (b) 给出了 2^1～2^{32} 的 2 的幂值，供参考。

$$\begin{array}{cccccccccccc} & & 1 & 0 & 1 & 1 & 0. & & 1 & 1 & & \\ 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 & 2^{-1} & 2^{-2} & & \\ 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 & \frac{1}{2} & \frac{1}{4} & & \end{array}$$

(a)

$2^1 =$	2	$2^{17} =$	131 072
$2^2 =$	4	$2^{18} =$	262 144
$2^3 =$	8	$2^{19} =$	524 288
$2^4 =$	16	$2^{20} =$	1 048 576
$2^5 =$	32	$2^{21} =$	2 097 152
$2^6 =$	64	$2^{22} =$	4 194 304
$2^7 =$	128	$2^{23} =$	8 388 608
$2^8 =$	256	$2^{24} =$	16 777 216
$2^9 =$	512	$2^{25} =$	33 554 432
$2^{10} =$	1 024	$2^{26} =$	67 108 864
$2^{11} =$	2 048	$2^{27} =$	134 217 728
$2^{12} =$	4 096	$2^{28} =$	268 435 456
$2^{13} =$	8 192	$2^{29} =$	536 870 912
$2^{14} =$	16 384	$2^{30} =$	1 073 741 824
$2^{15} =$	32 768	$2^{31} =$	2 147 483 648
$2^{16} =$	65 536	$2^{32} =$	4 294 967 296

(b)

图 1-1　(a) 二进制位值；(b) 2 的幂

二进制数常称为**二进制字**，简称为**字**。几种具有特定位数的二进制字已有了特定的名称。4 位二进制字称为**半字节**；8 位二进制字称为**字节**；16 位二进制

字常仅称为**字**；32 位二进制字则称为**双字**。二进制字的最右一位，即**最低有效位** (least significant bit)，常表示为 LSB；其最左一位，即**最高有效位** (most significant bit)，通常表示为 MSB。

把二进制数转换成等值的十进制数，只要将其每位数与该位的十进制位值相乘后加起来即可。例如，二进制数 101 相当于：

$$(1\times2^2)+(0\times2^1)+(1\times2^0),$$

即 $4+0+1$=十进制数 5。

对于二进制数 10110.11，则有：

$$(1\times2^4)+(0\times2^3)+(1\times2^2)+(1\times2^1)+(0\times2^0)+(1\times2^{-1})+(1\times2^{-2})$$

$$=16+0+4+2+0+0.5+0.25$$

$$=$$ 十进制数 22.75。

把十进制数转换成二进制数有两种常用的方法。第一种方法（见图 1-2 (a)）即为二进制转换成十进制方法的逆过程。例如，把十进制数 21（有时记为 21_{10}）转换成二进制数，首先从中减去一个合适的 2 的最大幂的数。对于 21_{10}，其合适的 2 的最大幂的数为 16 即 2^4。21 减去 16 后余 5。在 2^4 位处置 1，并看看 2 的下一个较小幂的数是否适于余数。因为 $2^3=8$，8 不适于余数 5，所以在 2^3 位处置 0。然后，试 2 的下一个较小幂，对本例为 2^2 即 4，它适于余数 5，因此在 2^2 位处置 1。把 2^2（即 4）从老余数 5 中减去剩下一新余数 1。因为 $2^1=2$，它不适于这个余数，所以在该位置 0。因为 $2^0=1$，它正好适于余数 1，所以在 2^0 位置 1。结果表明，21_{10} 等于二进制数 10101。这种转换过程似乎难以描述，但却容易实现。请读者试着把 46_{10} 转换为二进制数。其结果应为 101110。

另一种转换法如图 1-2 (b) 所示。把十进制数除以 2，并按图示记下商和余数。接着用 2 来除这个商，连续除下去直到商为 0 为止。所得的这列余数便是与该十进制数等价的二进制数。请注意，如果是自顶向下依次进行除法运算的，则 MSD 位于该列的底部，而 LSD 位于该列的顶部。如图 1-2 (b) 右侧所示，可通过把所得的二进制数再转换成十进制数的方法验证所得到的二进制数是正确的。

要把小于 1 的十进制数转换成二进制数，只要用 2 与其连乘并记下进位数，直到十进制小数点右边的量变为 0 时为止，如图 1-2 (c) 所示。所得的这列进位数就代表与十进制数等价的二进制数，其最高有效位位于进位数列的顶部。由此可得，十进制数 0.625 等于二进制数 0.101。对于用这种方法不能精确转换的十进制值（即十进制小数点右边的量永不变为 0），可持续转换过程直至得到所需二进制位数为止。

此处比较一下用十进制与用二进制表示数时所需的位数是有趣的。在十进制中，一位能表示 10^1 个数：0～9；两位能表示 10^2 即 100 个数：0～99；三位能表示 10^3 即 1000 个数：0～999。在二进制中也存在类似情况。一个二进制位能表示两个数：0 和 1；两个二进制位能表示 2^2 即 4 个数：0～11；三个二进制位能表示 2^3 即 8 个数：0～111。由此可见，N 个十进制位能表示 10^N 个数，N 个二进制位能表示 2^N 个数。因而 8 个二进制位能表示 2^8 即 256 个数，即十进制数 0～255。

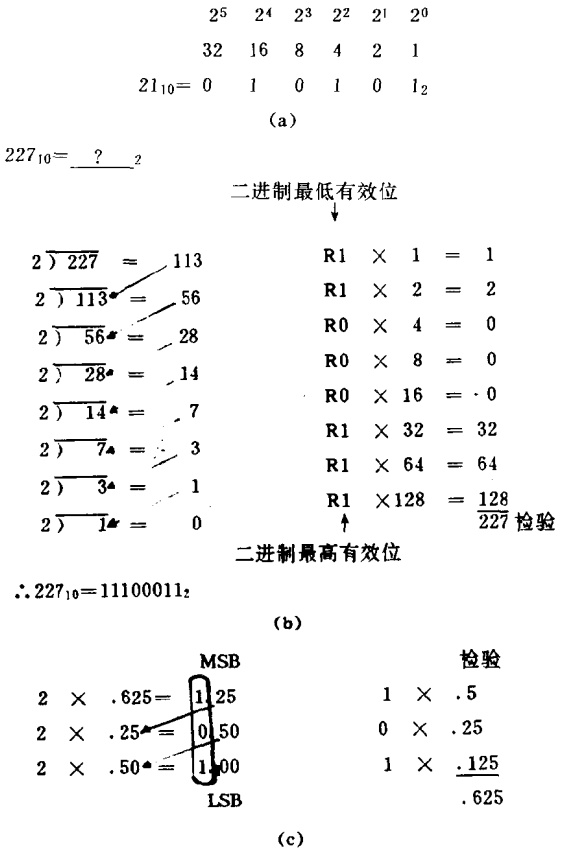


图 1-2 十进制转换成二进制

(a) 位值法；(b) 除以 2 法；(c) 十进制小数转换

十六进制

二进制并不是一种很紧致的码。就是说，用它来表示一个数时，所需的位数比用其他数制（比如十进制）要多。12 个二进制位仅能表示一个最大为 4095 的十进制数。计算机需要用二进制数据，但是用计算机工作的人记忆很长的二进制数时会很麻烦。解决这个问题的一个办法是用**十六进制** (hexadecimal) 即**基数为** 16 的数制。

图 1-3 (a) 列出了十六进制的一些数位的位值。十

六进制常常用缩写词 hex 表示。由于其基数为 16，因此它得有 16 个适当的符号，每位用一个符号。十六进制码的符号列于图 1-3 (b) 的表中。在十进制符号 0～9 用完之后，用字母 A～F 表示数值 10～15。

16^3 16^2 16^1 16^0 16^{-1} 16^{-2} 16^{-3}

4096 256 16 1 $\frac{1}{16}$ $\frac{1}{256}$ $\frac{1}{4096}$

(a)

Dec	Hex	Dec	Hex
0	0	8	8
1	1	9	9
2	2	10	A
3	3	11	B
4	4	12	C
5	5	13	D
6	6	14	E
7	7	15	F

(b)

$227_D = ?$ Hex

16) 227 = 14

16) 14 = 0

$227_{10} = E3_{16}$

LSD

R3 × 1 = 3

RE × 16 = 224

227

MSD

(c)

图 1-3 十六进制数

(a) 位值；(b) 符号；(c) 十进制转换成十六进制

如前所述，一个十六进制位等于 4 个二进制位。要把二进制数 11010110 转换成十六进制数，需先从二进制小数点向左以 4 个二进制位为一组来分组，然后用十六进制符号记下每组 4 个二进制位的值：

二进制 1101 0110

十六进制 D 6

0110 一组的值等于 6，1101 一组的值等于 13。因为在十六进制中 13 用 D 表示，所以二进制数 11010110 等于十六进制数 D6。常在数后用“H”标明这个数为十六进制数。比如，十六进制数 D6 常写成 D6H。可见，8 个二进制位仅用两个十六进制位就可表示。

要把某个数从十进制转换成十六进制，可按图 1-3 (c) 所示的我们已熟悉的方法去做。结果表明，227_{10} = E3H。可见，十六进制是一种比十进制更为紧致的码。两个十六进制位能表示出最大为 255 的十进制数。4 个十六进制位能表示出最大为 65535 的十进制数。

为了阐述十六进制数如何用于数字逻辑，8088A 微处理器的用户手册会指出，其 8 位宽的数据总线在给定的操作中能容置 3FH。把 3FH 转换成二进制便得到用 0 和 1 表示的形式，这可以用接到并行线上的示波器或逻辑分析仪观察到。3FH 只不过是一种速记形式，它比等值的二进制数记起来容易且不易出错。

BCD 码

标准 BCD 码

频率记数器、数字电压表或计算器等的输出用十进制显示，因而常用到**二-十进制**即 BCD **码**。BCD 码用 4 位二进制码逐个地表示十进制数的每一位。见表 1-1，最简单的 BCD 码是用标准二进制码的前 10 个数来表示 BCD 数 0～9 的。十六进制码 A～F 为无效 BCD 码。要把十进制数转换成等值的 BCD 数，只要用等值的 4 位二进制数来表示十进制数的每一位即可，如下所示：

十进制 5 2 9

BCD 0101 0010 1001

把上述过程反过来即可把 BCD 数转换成相应的十进制数。

表 1-1 常 用 数 码

十进制	二进制	八进制	十六进制	二进制编码十进制				反 射 葛莱码	七段显示 (1=on)	
				8421	BCD	余 3			a b c d e f g	显示
0	0000	0	0		0000	0011	0011	0000	1 1 1 1 1 1 0	0
1	0001	1	1		0001	0011	0100	0001	0 1 1 0 0 0 0	1
2	0010	2	2		0010	0011	0101	0011	1 1 0 1 1 0 1	2
3	0011	3	3		0011	0011	0110	0010	1 1 1 1 0 0 1	3
4	0100	4	4		0100	0011	0111	0110	0 1 1 0 0 1 1	4
5	0101	5	5		0101	0011	1000	0111	1 0 1 1 0 1 1	5
6	0110	6	6		0110	0011	1001	0101	1 0 1 1 1 1 1	6
7	0111	7	7		0111	0011	1010	0100	1 1 1 0 0 0 0	7
8	1000	10	8		1000	0011	1011	1100	1 1 1 1 1 1 1	8
9	1001	11	9		1001	0011	1100	1101	1 1 1 0 0 1 1	9
10	1010	12	A	0001	0000	0100	0011	1111	1 1 1 1 1 0 1	A
11	1011	13	B	0001	0001	0100	0100	1110	0 0 1 1 1 1 1	B
12	1100	14	C	0001	0010	0100	0101	1010	0 0 0 1 1 0 1	C
13	1101	15	D	0001	0011	0100	0110	1011	0 1 1 1 1 0 1	D
14	1110	16	E	0001	0100	0100	0111	1001	1 1 0 1 1 1 1	E
15	1111	17	F	0001	0101	0100	1000	1000	1 0 0 0 1 1 1	F

葛莱码

葛莱码(Gray code)是另一种重要的二进制码，常用于给计算机控制的车床之类机器的轴位数据编码。这种码与标准二进制码具有相同的可能组合，但是排列次序不一样，请参看表 1-1 中的 4 位码的示例。请注意，用葛莱码计数时，每次仅改变一个二进制位。

要构造一个比表 1-1 更大的葛莱码表，简便的方法是观察 0 和 1 的分布规律并按这个规律扩展下去。其最低位的一列以 0 开始，接着向下则是两个 1 与两个 0 交替出现。下一个较高位的一列以两个 0 开始，接着就是 4 个 1 与 4 个 0 交替出现。更高位的第三列以 4 个 0 开始，接着是 8 个 1 与 8 个 0 交替出现。至此可以看出规律来了。试计算出十进制数 16 的葛莱码，它应为 11000。

七段显示码

图 1-4 (a) 示出了通常用于数字仪表等仪器中的七段显示管的各段标识符。图 1-4 (b) 为共阴极发光二极显示管，表 1-1 列出了用这类显示管显示 0～9 以及 A～F 所需的逻辑电平。对于如图 1-4 (c) 所示的共阳极发光二极显示管，仅需把表 1-1 中的段码取反即可。

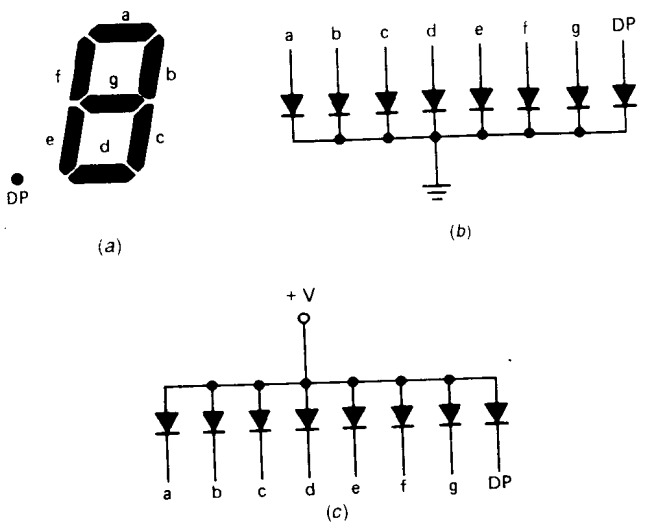


图 1-4 七段发光二极管显示

(a) 段标号；(b) 共阴极型简图；(c) 共阳极型简图

字母数字码

在与计算机或在计算机之间通信时，需要用基于二进制的代码表示字母和数字。用于这种目的的通用代码，其每个字具有 7 或 8 位，称为字母数字码。为了查出这种码中的可能错误，常常增加一个附加位，即**奇偶校验位**，作为其最高有效位。

奇偶校验是一个用于鉴别数据字中具有奇数个 1 还是偶数个 1 的术语。若某个数据字具有奇数个 1，则称这个字具有**奇校验**。二进制字 0110111 中有 5 个 1，因而它具有奇校验。二进制字 0110000 具有偶数个 1 (两个)，因而它具有**偶校验**。

奇偶校验位的实际应用如下所述。正发送数据字的系统检查该字的奇偶校验。若该数据字为奇校验，则该系统会把奇偶校验位置成 1，使得该数据字加上奇偶校验位之后成为偶校验。若该数据字为偶校验，则发送系统会把奇偶校验位重置成 0，使得该数据字加上奇偶校验位之后也为偶校验。接收系统检查收到的数据字加上奇偶校验位的奇偶校验。若接收系统检查出其为奇校验，则意味着出现了某种错误，它就告知发送系统重发这个数据。因此称这种系统为采用偶校验的系统。用类似的方法也可以建立采用奇校验的系统。

ASCII 码

表 1-2 列出了几种字母数字码。其中的第一种为 ASCII 码，即美国信息交换用标准代码。此表列出的是 7 位 ASCII 码。用 7 位可以编码出多达 128 个字符，这足以表示所有的大小写字母、数字、标点符号以及控制符。ASCII 码是这样排列的：如果仅用到大写字母、数字和一些控制符，那么较低的 6 位就满足了。若要奇偶校验，则可对基本 7 位码增加一个奇偶校验位作为最高有效位。例如，二进制字 11000100 为具有奇校验的大写字母 D 的 ASCII 码。表 1-3 给出了用于 ASCII 码表中的控制符符号的含义。

EBCDIC 码

在 IBM 设备中常遇到另一种字母数字编码——扩展的二-十进制交换代码，即 EBCDIC 码。这种码没有奇偶校验位时为 8 位。要进行奇偶校验时可增加一个第 9 位作为奇偶校验位。为了节省空间，表 1-2 中的 EBCDIC 码的 8 个二进制位是用等价的两个十六进制位表示的。

1.2 二进制数、十六进制数以及 BCD 数的算术运算

二进制算术

表 1-2　常用字母数字码

ASCII 符号	7 位 ASCII 的十六进制码	EBCDIC 符号	EBCDIC 的十六进制码	ASCII 符号	7 位 ASCII 的十六进制码	EBCDIC 符号	EBCDIC 的十六进制码	ASCII 符号	7 位 ASCII 的十六进制码	EBCDIC 符号	EBCDIC 的十六进制码
NUL	00	NUL	00	*	2A	*	5C	T	54	T	E3
SOH	01	SOH	01	+	2B	+	4E	U	55	U	E4
STX	02	STX	02	,	2C	,	6B	V	56	V	E5
ETX	03	ETX	03	—	2D	—	60	W	57	W	E6
EOT	04	EOT	37	.	2E	.	4B	X	58	X	E7
ENQ	05	ENQ	2D	/	2F	/	61	Y	59	Y	E8
ACK	06	ACK	2E	0	30	0	F0	Z	5A	Z	E9
BEL	07	BEL	2F	1	31	1	F1	[	5B	|	AD
BS	08	BS	16	2	32	2	F2	x	5C	NL	15
HT	09	HT	05	3	33	3	F3	]	5D	¦	DD
LF	0A	LF	25	4	34	4	F4	∧	5E		5F
VT	0B	VT	0B	5	35	5	F5	—	5F	—	6D
FF	0C	FF	0C	6	36	6	F6	、	60	RES	14
CR	0D	CR	0D	7	37	7	F7	a	61	a	81
S0	0E	S0	0E	8	38	8	F8	b	62	b	82
SI	0F	SI	OF	9	39	9	F9	c	63	c	83
DLE	10	DLE	10	:	3A	:	7A	d	64	d	84
DC1	11	DC1	11	;	3B	;	5E	e	65	e	85
DC2	12	DC2	12		3C		4C	f	66	f	86
DC3	13	DC3	13	=	3D	=	7E	g	67	g	87
DC4	14	DC4	35		3E		6E	h	68	h	88
NAK	15	NAK	3D	?	3F	?	6F	i	69	i	89
SYN	16	SYN	32	@	40	@	7C	j	6A	j	91
ETB	17	EOB	26	A	41	A	C1	k	6B	k	92
CAN	18	CAN	18	B	42	B	C2	l	6C	l	93
EM	19	EM	19	C	43	C	C3	m	6D	m	94
SUB	1A	SUB	3F	D	44	D	C4	n	6E	n	95
ESC	1B	BYP	24	E	45	E	C5	o	6F	o	96
FS	1C	FLS	1C	F	46	F	C6	p	70	p	97
GS	1D	GS	1D	G	47	G	C7	q	71	q	98
RS	1E	RDS	1E	H	48	H	C8	r	72	r	99
US	1F	US	1F	I	49	I	C9	s	73	s	A2
SP	20	SP	40	J	4A	J	D1	t	74	t	A3
!	21	!	5A	K	4B	K	D2	u	75	u	A4
"	22	"	7F	L	4C	L	D3	v	76	v	A5
#	23	#	7B	M	4D	M	D4	w	77	w	A6
$	24	$	5B	N	4E	N	D5	x	78	x	A7
%	25	%	6C	O	4F	O	D6	y	79	y	A8
&	26	&	50	P	50	P	D7	z	7A	z	A9
'	27	'	7D	Q	51	Q	D8	{	7B	{	8B
(	28	(	4D	R	52	R	D9	|	7C	¦	4F
)	29	)	5D	S	53	S	E2	}	7D	}	9B
								~	7E	⊄	4A
								DEL	7F	DEL	07

表 1-3　控制符定义

NULL	空行	DEL	删除
SOH	标题开始	DC1	设备控制 1
STX	正文开始	DC2	设备控制 2
ETX	正文结束	DC3	设备控制 3
EOT	传送结束	DC4	设备控制 4
ENQ	询问	NAK	否认
ACK	承认	SYN	同步空转
BEL	报警	ETB	信息块传送结束
BS	退格	CAN	作废
HT	横向制表	EM	记录媒体结束
LF	换行	SUB	代替
VT	纵向制表	ESC	转义
FF	改换格式	FS	字段分隔
CR	回车	GS	字组分隔
SO	移出	RS	记录分隔
SI	移入	US	单元分隔

加　法

图 1-5 (a) 为两个二进制位与低位加产生的进位 (C_{IN}) 相加的真值表。图 1-5 (b) 给出了按规则把两个 8 位二进制数相加的结果。假定 $C_{IN}=1$，则

$1+0+C_{IN}=0$ 并向高位进位，

$1+1+C_{IN}=1$ 并向高位进位，

因为任一位的结果只能为一个 1 或 0。

输入			输出	
A	B	C_{IN}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S=A\oplus B\oplus C_{IN}$

$C_{out}=A\cdot B+C_{IN}\ (A\oplus B)$

(a)

```
  10011010
+ 11011100
-----------
 [1]01110110
  ↑
  ———进位
```

(b)

图 1-5　二进制加法

(a) 两个位与进位相加的真值表；(b) 两个 8 位二进制数相加

2 的补码符号-数值二进制

手写一个表示诸如温度这类物理量的数时，可以简单地在数前置一个“+”号表示它是正数，或在其前写一个“−”号表示它是负数。然而，要把温度这类既可为正也可为负的数值存入计算机存储器，则有一个问题。由于计算机存储器只能存储 1 和 0，因而必须建立某种方法以便用 1 或 0 来表示数的符号。

一种常用的表示带符号数的方法是，把数据字的最高有效位留作**符号位**，而用数据字的其他位表示量的大小（值）。用 8 位字工作的计算机将其最高有效位（第 8 位）用作符号位，而将其余的较低 7 位用于表示数的绝对值。通常规定，符号位为 0 表示正数，为 1 表示负数。

为了使带符号数便于运算，要把负数的数值用一种称为 2 的补码的特殊形式来表示。将数据字的每一位取反后加 1，便得二进制数的补码。用几个例子会有助于讲清楚这种方法。

数字 $+7_{10}$ 用 8 位符号-数值形式表示为 00000111。其符号位为 0，表明该数为正数。正数的数值直接用二进制表示，所以 00000111 在低位表示 7_{10}。

用 8 位 2 的补码符号-数值形式表示 -7_{10} 时，则需从 +7 的 8 位码即 00000111 做起。变反其每一位（包括最高有效位），得 11111000。然后加 1 得 11111001。这就是 -7_{10} 的正确表示。图 1-6 多给出了一些用 8 位符号-数值形式表示的正数和负数的例子。请读者练习每个例子，看看能否得出同样的结果。

如下所述，将上述过程反过来便可得到符号-数值形式表示的数的数值。如果符号位为 0，表明该数为正数，那么它的 7 个低位就直接表示该数的二进制数值。如果符号位为 1，表明该数为负数，那么其数值是以 2 的补码表示的。要得出负数的标准二进制表示的数值，则取反数据字的每一位（包括符号位）后加 1 即可。例如数据字 11101011，每一位取反后则为 00010100，然后加 1 则成为 00010101，这等于 21_{10}。由此可知，11101011 的原数为 -21_{10}。请再试着转换图 1-6 中的几个数。

	符号位	
+ 7	0	0000111
+ 46	0	0101110
+105	0	1101001
− 12	1	1110100
− 54	1	1001010
−117	1	0001011
− 46	1	1010010

} 符号及数值的 2 的补码

图 1-6　用符号位及 2 的补码表示的正负数

图 1-7 给出了几个这种带符号二进制数加法的例子。符号位也像其他位一样相加。图 1-7 (a) 给出了两个正数相加的结果，其符号位为 0，因此它是正数。图 1-7 (b) 给出了第二个例子：把 −9 与 +13 相加，即等效于 13−9。其和的符号位为 0，表明所得结果 4 是正数，而且是以二进制原码表示的。

图 1-7 (c) 给出了把 −13 与一个较小的正数 +9 相加的结果。其和的符号位为 1，表明它是负数且其数值是以 2 的补码表示的。把用 2 的补码表示的结果转换成用二进制原码表示的带符号数的方法为

1. 变反每一位以得出反码。
2. 加 1。
3. 前置一个负号以表明所得结果是负数。

作为最后一个例子，图 1-7 (d) 给出了把两个负数相加的结果。该结果的符号位为 1，因此它是负数，而且是以 2 的补码形式表示的。变反每一位后再加 1，并在所得数前置一个负号，便得到该数较易识别的表示形式。

现在让我们看一下能够用 8 位的符号-数值形式

表示的数的范围。8 个位最多能够表示出 2^8，即 256 个数。由于既要表示正数，又要表示负数，所以这个范围的一半要为正，另一半要为负。因此，该范围为 −128～+127。下面是这些值的符号-数值形式的二进制表示：

01111111	+127
⋮	
00000001	+1
00000000	0
11111111	−1
⋮	
10000001	−127
10000000	−128

观察以上这些数的结构可注意到，该图中把 128～255 的普通代码下移用来表示 −128～−1。

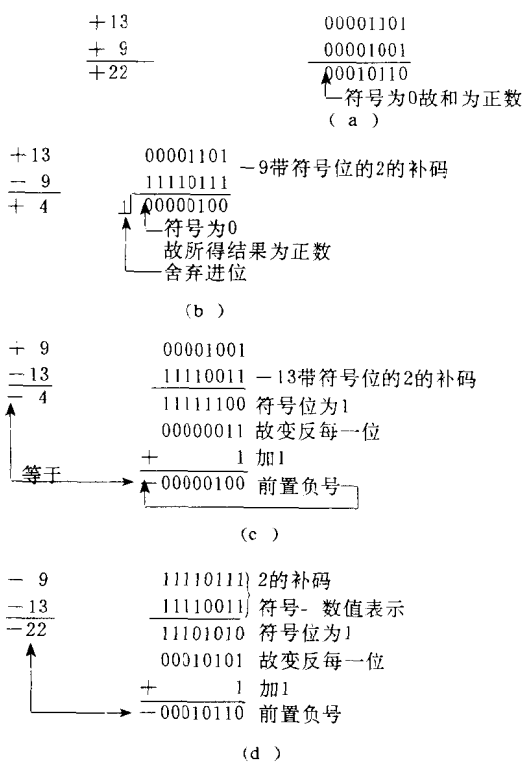


图1-7 带符号二进制数加法

(a)+9与+13相加；(b)−9与+13相加；(c)+9与−13相加；(d)−9与−13相加

若计算机把带符号数作为 16 位字存储，则可以表示的数范围更大。由于 16 位给出 2^{16} 即 65，536 个可能的值，所以用 16 位符号-数值形式表示的数范围为 −32，768～+32，767。以 16 位符号-数值形式表示的数的运算方法与 8 位时相同。

减　法

做二进制减法有两种常用的方法：笔算法和 2 的补码加法。图 1-8 (a) 列出了两个二进制位 A 与 B 的二进制减法真值表。该真值表还包含有从低位来的借位 B_{IN} 的作用。图 1-8 (b) 为两个 8 位数相减的笔算法的例子。该方法借助于真值表来做二进制减法，与做十进制减法的方法相同。

另一种进行二进制减法的方法是，把减数（底数）的 2 的补码表示与被减数（顶数）相加。这种方法的运算过程如图 1-8 (c) 所示。首先以符号-数值形式表示出被减数；然后将减数的负数用 2 的补码符号-数值形式表示；最后把所得的这两部分相加。对于图 1-8 (c) 中的例子，其结果的符号位为 0，表明它是正数且以原码表示。相加后产生的最终进位可舍弃。图 1-8 (d) 给出了这种减法的另一个例子。该例中减数大于被减数。同样，也把被减数以符号-数值形式表示，将减数的负数用 2 的补码符号-数值形式表示，并把所得的这两部分相加。该例所得结果的符号位为 1，表明它是负数，而且其数值是以 2 的补码表示的。如图 1-8 (d) 所示，将每一位变反后加 1 并前置一个减号，则可把所得结果转换成更易识别的形式。

做带符号数加法或减法时可能出现的问题是**上溢**和**下溢**。当两个带符号数相加所得的和的位数大于用来表示这个数值的位数时，所得结果将会上溢到符号位并给出一个不正确的答案。例如，带符号正数 01001001 与带符号正数 01101101 相加，结果是 10110110。它的最高有效位为 1，指示它是负数，这对于两个正数之和显然是不正确的。类似，做 8 位带符号数减法时，所得数值大于 128 会引起下溢到符号位，也产生一个不正确的结果。

为了简明起见，给出的例子用的是 8 位，但是这种方法适用于任何位数。这种方法似乎有点笨拙，但在计算机或微处理器中却是容易实现的，因为它仅需要简单的变反和加法操作。

乘　法

做二进制乘法有几种方法。图 1-9 所示的方法称为笔算法，它与十进制数相乘的方法相同。把底数（即乘数）的最低有效位与顶数（即被乘数）相乘，并记下所得的部分积。然后把乘数的下一位与顶数相乘，并把所得的部分积左移一位记在上一个部分积的下面。把所有的部分积加起来便得到总积。用手做乘法时这种方法很方便，但它对计算机来说是不实用的，因为所需的这种移位对计算机实现起来是困难的。

计算机所用的乘法之一是重复加法。比如，要完成 7×55，计算机只是把 7 个 55 加起来。然而，对于大数字相乘，这种方法是很慢的。例如，要完成 786×253，用这种方法就要做 252 次加法运算。