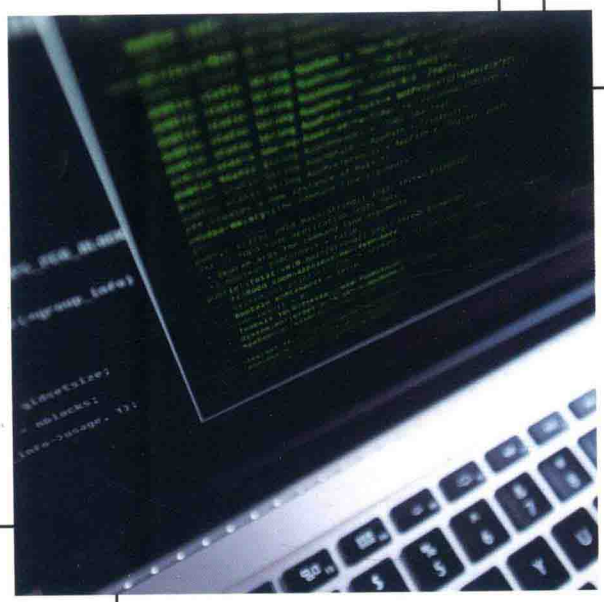
国家新闻出版改革发展项目库入库项目
高等院校计算机类规划教材
全国高等院校计算机基础教育研究会重点立项项目



微机原理与接口技术

——基于Proteus仿真

朱有产　刘淑平
王桂兰　秦金磊　编著

"互联网+"创新型教材
北邮智信

北京邮电大学出版社
www.buptpress.com

国家新闻出版改革发展项目库入库项目
高等院校计算机类规划教材
全国高等院校计算机基础教育研究会重点立项项目

微机原理与接口技术

——基于 Proteus 仿真

朱有产　刘淑平　王桂兰　秦金磊　编著

北京邮电大学出版社
www.buptpress.com

内 容 简 介

本书以 Intel 8086 微处理器和 IBM PC 系列微机为对象，从微型计算机系统的应用出发，系统地介绍了微型计算机的基本组成、工作原理、接口技术及应用。作者在总结教学经验，研究相关仿真技术和各类教材的基础上，以“项目为线，案例为点”的思路编写了各章节相关内容。本书共 10 章，包括微型计算机基础知识概述、微处理器、指令系统及汇编语言程序设计、存储器系统、输入/输出技术、可编程并行 I/O 接口芯片 Intel 8255A、可编程计数器/定时器 8253A、中断技术及 8259A、微机系统串行通信及接口、D/A 与 A/D 转换接口。本书以交通信号灯控制系统在 Proteus ISIS 仿真平台的实现为线，将其贯穿各章节内容，以在 Proteus ISIS 仿真平台中实现的案例为知识点。本书内容全面，实用性强，原理、技术与应用并重，以 Proteus ISIS 仿真实验方法进行讲述，有特点和新意。本书中提供的实例全部在 Proteus 中调试通过。

本书可作为高等院校理工科自动化、电气与电子类等相关专业本科以及成人高等教育或大专层次的教材，对研究生和从事微机测控及接口技术应用的工程技术人员也是一本很好的参考书。

图书在版编目(CIP)数据

微机原理与接口技术 ：基于 Proteus 仿真 / 朱有产等编著. -- 北京 ：北京邮电大学出版社，2021.2
ISBN 978-7-5635-6328-9

Ⅰ. ①微… Ⅱ. ①朱… Ⅲ. ①微型计算机—理论—教材②微型计算机—接口技术—教材 Ⅳ. ①TP36

中国版本图书馆 CIP 数据核字(2021)第 027097 号

策划编辑：马晓仟　　**责任编辑**：孙宏颖　　**封面设计**：七星博纳

出版发行：北京邮电大学出版社
社　　址：北京市海淀区西土城路 10 号
邮政编码：100876
发 行 部：电话：010-62282185　传真：010-62283578
E-mail：publish@bupt.edu.cn
经　　销：各地新华书店
印　　刷：保定市中画美凯印刷有限公司
开　　本：787 mm×1 092 mm　1/16
印　　张：16.25
字　　数：438 千字
版　　次：2021 年 2 月第 1 版
印　　次：2021 年 2 月第 1 次印刷

ISBN 978-7-5635-6328-9　　定价：39.80 元

·如有印装质量问题，请与北京邮电大学出版社发行部联系·

此为试读，需要完整PDF请访问：www.ertongbook.com

前　言

“微机原理与接口技术”是高等学校电子信息、自动化、电气工程等工科类各专业的基础核心课程。课程目标是使学生从系统的角度出发，掌握微机系统的基本组成、工作原理、接口技术及应用方法，具有微机系统的初步开发能力。作者在总结多年教学科研及实践经验的基础上，结合计算机仿真技术的发展，对课程相关资料进行了综合分析提炼，编写了本书。

本书在内容选取与组织上进行了革新，以 Intel 8086 微处理器和 IBM PC 系列微机为对象，从微型计算机系统的应用出发，系统地介绍了微型计算机的基本组成、工作原理、接口技术及应用。作者在总结教学经验，研究相关仿真技术和各类教材的基础上，以“项目为线，案例为点”的思路编写了各章节相关内容。本书共 10 章，包括微型计算机基础知识概述、微处理器、指令系统及汇编语言程序设计、存储器系统、输入/输出技术、可编程并行 I/O 接口芯片 Intel 8255A、可编程计数器/定时器 8253A、中断技术及 8259A、微机系统串行通信及接口、D/A 与 A/D 转换接口。本书以交通信号灯控制系统在 Proteus ISIS 仿真平台的实现为线，将其贯穿各章节内容，以在 Proteus ISIS 仿真平台中实现的案例为知识点。本书内容全面，实用性强，原理、技术与应用并重，以 Proteus ISIS 仿真实验方法进行讲述，有特点和新意。本书中提供的实例全部在 Proteus 中调试通过。

本书有如下特色。

① 以项目为线。本书以交通信号灯控制系统在 Proteus ISIS 仿真平台的实现为线，以经典的 Intel 8086 为主要对象，各章节内容基于项目连接成线，侧重微机系统的设计与实现。重点突出，内容全面。

② 以案例为点。本书从应用需求出发，在讲清基本原理的基础上，按难易程度讲解典型示例或案例及其 Proteus 实现，突出了对学生软硬件结合的思维方法和动手能力的培养。

③ 先进的实验手段。本书选用了适用于该课程教学和实践的 Proteus ISIS 仿真平台，书中案例和项目实现过程按照课程内容进行规划，既有理论设计又有仿真实现，使学生掌握知识的同时又可体会到技术的发展，本书较好地体现了从整体到局部又到整体的知识体系。

④ 可读性强。随着项目的一步步实现，课程内容由浅入深、分散难点。在接口部分，形成芯片结构、编程和项目实现的讲解体系，以便学生理解。

本书的编写采用了集体讨论、分工编写、再讨论修改、统稿的方式。本书的第 1、3 章由朱有产编写；第 2 章由刘淑平编写；第 4、8、9、10 章由王桂兰编写；第 5、6、7 章由秦金磊编写；附录由朱有产编写。本书由朱有产统稿并最后定稿。本书定稿后，由王振旗教授主审。

本书配有教辅《微机原理与接口技术辅导与实验》，包括 MASM 使用说明、Proteus 仿真平台使用说明、习题解答、MCS-51 简介及其仿真案例等内容。

本书的编写得到了华北电力大学专业建设平台领导的大力支持；得到了微机原理教学

团队全体老师的大力支持；得到了广州市风标电子技术有限公司的大力支持，公司技术人员指导了部分 Proteus 仿真实例的设计；得到了全国高等院校计算机基础教育研究会的大力支持。在此，全体编著人员向所有对本书的编写、出版等工作给予大力支持的单位和人员表示真诚的感谢！

由于作者水平有限，书中难免有错误和不妥之处，敬请广大读者提出宝贵意见。

作　者

2020 年 8 月于华北电力大学

目　录

第 1 章　微型计算机基础知识概述

本章从介绍计算机的软硬件基本组成、数制、编码及数据类型开始，重点介绍计算机的基本结构及其整机工作原理。下面先从一个项目谈起。

基本项目：交通信号灯系统。

看到这个项目，大家可能会问系统是什么样子？系统的功能要求是什么？硬件系统、软件系统用什么？开发环境选什么？

也许大家会想到这些：①使用已学过的知识；②收集项目相关资料；③需求分析；④软硬件系统设计（初步设计）；⑤开发环境的学习；⑥硬件系统详细设计与调试；⑦软件系统详细设计与调试；⑧系统整体调试；⑨现场应用安装调试；⑩试用期间问题的解决与修改。

大家想想，模拟交通信号灯系统的基本需求可能是：假设 A 车道与 B 车道交叉组成十字路口，A 是主道，B 是支道，直接对车辆进行交通管理。①用发光二极管模拟交通信号灯。②正常情况下，A、B 两车道轮流放行，A 车道放行 m_1 秒，其中 n_1 秒用于警告；B 车道放行 m_2 秒，其中 n_2 秒用于警告。③有紧急车辆通过时，按下某开关使 A、B 车道均为红灯，禁行（或禁行 m_3 秒），紧急情况解除后，恢复正常控制。

大家想到的系统应该是什么样的呢？不管怎样，先来回顾一下我们已学习过的相关知识。

1.1　微机的基本结构

微型计算机系统应由硬件系统和软件系统两大部分构成。大家回想一下“大学计算机基础”课程里介绍的微型计算机系统（外观、主要构成、常用部件等）。随着技术的不断进步，这些部件的功能与性能都在不断地发展，但微型计算机的基本结构没变，如图 1-1 所示。

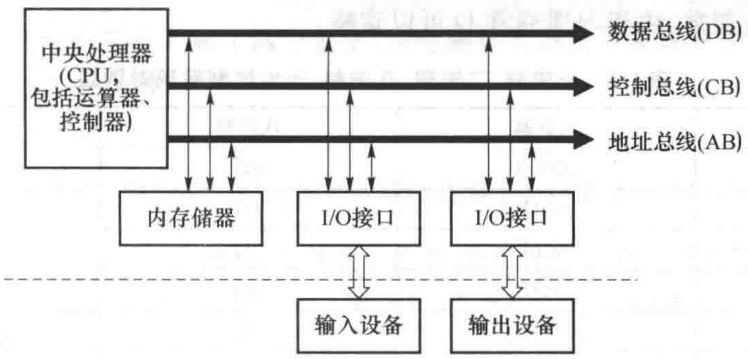


图 1-1　微型计算机的基本结构

首先要有能进行运算的部件，称为运算器；其次要有能记忆程序、原始数据和中间结果以及为了使机器能自动进行运算而编制的各种命令的器件，这种器件就称为存储器；最后要有控制器，能根据事先给定的命令发出各种控制信息，使整个计算过程能一步步地自动进行。原始的数

此为试读，需要完整PDF请访问：www.ertongbook.com

据与命令要输入，要有输入设备；计算结果（或中间过程）要输出，要有输出设备。

目前，微型计算机都采用总线结构。所谓总线就是用来传送信息的一组通信线，分内总线和外总线。内总线将微处理器、内存储器和输入输出接口部件连接起来，分为地址总线、数据总线和控制总线。数据总线用来传送各种原始数据、中间结果、程序等。计算机的各种命令（即程序）以数据形式由存储器送入控制器，经过控制器译码后变为各种控制信号，由控制总线传送，以控制运算器按规定一步步地进行各种运算和处理，控制存储器的读和写，控制输入输出设备的启停。CPU从内存储器读写数据、程序等信息时首先要给出信息所在的内存储器中的位置信息，即地址信息，由地址总线传送。微机与外设（包括其他计算机）的连接线称作外总线，其功能是实现计算机与计算机或计算机与其他外设的信息传送。

通常把运算器、控制器和存储器合在一起称为微机的主机；把各种输入输出设备称为计算机的外围设备或外部设备（Peripheral）；把运算器和控制器合在一起称为中央处理单元（CPU）；把整个CPU集成在一个集成电路芯片上，称为微处理器（Microprocessor）；整个微型计算机只安装在一块印刷电路板上，常称为单板计算机；把整个计算机集成在一个芯片上，称为单片机。不论计算机规模大小，必须同时具有CPU、存储器和输入输出设备，才能构成一台计算机。

微机软件系统分系统软件和应用软件两大类。系统软件用来对构成微机的各部件进行管理和协调，使它们有条不紊高效率地工作，如操作系统、高级语言、数据库系统等；应用软件是针对不同应用、实现用户要求的功能软件，如IE、MIS程序、高校的综合教务管理等。

1.2 微型计算机的基本知识

1.2.1 计算机中的常用数制

人们通常用十进制数来计数和计算，而计算机只识别由“0”和“1”构成的二进制数。当一个较大的数用二进制来表示或人们在书写计算机程序时，数据位数既长又难记忆，常采用八进制、十六进制和十进制计数制。

表1-1列出了4种进位制中数的表示法，其中数后B（Binary）表示二进制数；Q（Octal的缩写为字母“O”，为区别于数字“0”，写为“Q”）表示八进制数；H（Hexadecimal）表示十六进制数；D（Decimal）表示十进制数，由于习惯通常D可以省略。

表1-1 十进制、二进制、八进制、十六进制数码对照表

十进制	二进制	八进制	十六进制
0	0000B	0Q	0H
1	0001B	1Q	1H
2	0010B	2Q	2H
3	0011B	3Q	3H
4	0100B	4Q	4H
5	0101B	5Q	5H
6	0110B	6Q	6H
7	0111B	7Q	7H
8	1000B	10Q	8H

续表

十进制	二进制	八进制	十六进制
9	1001B	11Q	9H
10	1010B	12Q	AH
11	1011B	13Q	BH
12	1100B	14Q	CH
13	1101B	15Q	DH
14	1110B	16Q	EH
15	1111B	17Q	FH

那什么是计数制呢？任意进制的数 N 均可表示为

$$N=a_{n-1}X^{n-1}+a_{n-2}X^{n-2}+\cdots+a_0X^0+a_{-1}X^{-1}+\cdots+a_{-(m-1)}X^{-(m-1)}+a_{-m}X^{-m}$$

上式中数 N 的大小不但取决于其中的数字本身，而且还取决于各数字所在的位置，故上式称为数的位置计数法。其中，对每一个数位 $a_{n-1},\cdots,a_0,a_{-1},\cdots,a_{-m}$ 赋予一定的位值 $X^{n-1},\cdots,X^0,X^{-1},\cdots,X^{-m}$，则称各位值为权。每个数位上的数字所表示的量是这个数字和权的乘积。相邻两位中高位的权与低位的权之比如果是个常数，则此常数称为基数，式中用 X 表示。式中从 a_0X^0 起向左是数的整数部分，向右是数的小数部分。数位 $a_i(n-1\geqslant i\geqslant -m)$ 可以在 0，1，…，$X-1$ 共 X 种基数中任意取值。m 和 n 为幂指数，均为正整数。基数 X 取不同值，便得到不同进位计数制的表达式。

1. 十进制数

十进制数的表达式为 $(N)_{10}=\sum_{i=-m}^{n-1}a_i\times 10^i$。其特点是：$a_i$ 只能在 0～9 这 10 个数字中取值；每个数位上的权都是 10 的 i 次方；在加、减运算中，采用“逢十进一”“借一当十”的规则。例如

$$2\,346.18=2\times10^3+3\times10^2+4\times10^1+6\times10^0+1\times10^{-1}+8\times10^{-2}$$

2. 二进制数

二进制数的表达式为 $(N)_2=\sum_{i=-m}^{n-1}a_i\times 2^i$。其特点是：$a_i$ 只能是 0 或 1；每个数位上的权都是 2 的 i 次方；在加、减运算中，采用“逢二进一”“借一当二”的规则。例如

$$11001.101\text{B}=1\times2^4+1\times2^3+0\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$

因为二进制计数制中只有 0 和 1 两个数字，用电路实现最为方便且运算也特别简单，所以电子计算机内部均采用此计数制。

3. 八进制数

八进制数的表达式为 $(N)_8=\sum_{i=-m}^{n-1}a_i\times 8^i$。其特点是：$a_i$ 只能在 0～7 这 8 个数字中取值；每个数位上的权都是 8 的 i 次方；在加、减运算中，采用“逢八进一”“借一当八”的规则。例如

$$257.16\text{Q}=2\times8^2+5\times8^1+7\times8^0+1\times8^{-1}+6\times8^{-2}$$

4. 十六进制数

十六进制数的表达式为 $(N)_{16}=\sum_{i=-m}^{n-1}a_i\times 16^i$。其特点是：$a_i$ 只能在 0～15 这 16 个数字中取值（其中 0～9 这 10 个数字借用十进制中的数码，10～15 这 6 个数分别用 A、B、C、D、E、F 表示）；每个数位上的权都是 16 的 i 次方；在加、减法运算中，采用“逢十六进一”“借一当十六”的规则。例如

$$63BE.FAH=6\times16^3+3\times16^2+11\times16^1+14\times16^0+15\times16^{-1}+10\times16^{-2}$$

1.2.2 各种数制间的转换

人们习惯使用十进制数，而计算机只认识二进制数。为了书写方便，编写汇编程序时多采用十六进制数。因此在不同场合需要不同进位计数制之间的转换。

1. 非十进制数到十进制数的转换

非十进制数到十进制数的转换只要将非十进制数按相应的权表达式展开，按十进制运算规则求表达式的值，即得到它所对应的十进制数。

例 1-1 将二进制数 1101.001B、十六进制数 1A.CH 转换为十进制数。

解：根据二进制数的权展开，有

$$1101.001B=1\times2^3+1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+0\times2^{-2}+1\times2^{-3}=13.125$$

根据十六进制数的权展开，有

$$1A.CH=1\times16^1+10\times16^0+12\times16^{-1}=26.75$$

2. 十进制数转换为非十进制数

(1) 十进制数转换为二进制数

十进制数转换为二进制数时，整数和小数部分应分别转换。整数部分采用“除 2(基)取余”法，即连续除以 2 并取余数作为结果，直至商为 0，将得到的余数从低位(最先得到的余数)到高位(最后得到的余数)依次排列即得到转换后二进制数的整数部分。对小数部分采用“乘 2(基)取整”法，即对小数部分连续乘 2，以最先得到的乘积的整数部分为最高位，直至小数部分为零或达到所要求的精度。

例 1-2 将十进制数 100.625 转化为二进制数。

解：

整数部分：

```
2 | 100
  2 | 50   ……余数 0
    2 | 25   ……余数 0
      2 | 12   ……余数 1
        2 | 6   ……余数 0
          2 | 3   ……余数 0
            2 | 1   ……余数 1
              0   ……余数 1
```

最低位 ↑ 最高位

小数部分：

```
  0.625
×     2
  1.250   ……整数 1
×     2
  0.500   ……整数 0
×     2
  1.000   ……整数 1
```

最高位 ↓ 最低位

所以得 100.625=1100100.101B。

(2) 十进制数转换为八进制数、十六进制数

与十进制数转换为二进制数的方法类似，整数部分采用“除 8、16(基)取余”法，小数部分采用“乘 8、16(基)取整”法。

例 1-3 将十进制数 100.625 转化为十六进制数。

解：

整数部分：

```
16 | 100
   16 | 6   ……余数 4
      0   ……余数 6
```

↑

小数部分：

```
  0.625
×    16
  3750
×   625
 10.000   ……整数 A
```

所以得 100.625=64.AH，同理得 100.625=144.5Q。

3. 二进制数与八进制数、十六进制数间的转换

由于 $2^4=16$，故一位十六进制数能够表示的数值恰好相当于 4 位二进制数能够表示的数值。将二进制数转换为十六进制数的方法是：对整数部分从小数点开始向左每 4 位一组，若最后一组不足 4 位，则在其左边补零直到 4 位；对小数部分从小数点开始向右每 4 位一组，若最后一组不足 4 位，则在其右边补零直到 4 位。然后将每组二进制数用对应的十六进制数代替，则得到转换结果。

例 1-4　将二进制数 110011011.101101B 转换为十六进制数。

解：

二进制数：0001　1001　1011.1011　0100

十六进制数：1　9　B.　B　4

所以得 110011011.101101B=19B.B4H。

将十六进制数转换为二进制数与上述过程相反，读者可自己试试。

由于 $2^3=8$，故一位八进制数恰好等于 3 位二进制数能够表示的数值。转换方法同上。

例 1-5　将二进制数 1110011011.1011011B 转换为八进制数。

解：

二进制数：001　110　011　011.101　101　100

八进制数：1　6　3　3.　5　5　4

所以得 1110011011.1011011B=1633.554Q。

1.2.3　无符号二进制数

在编程解决实际问题时，首先要涉及数据及数据类型。在 80X86 系列微机中，常用的数据类型包括无符号整数、有符号整数、BCD 数、字符串、位、浮点数。

在 80X86 系列微处理器中，参加运算的整数操作数可为 8 位长的字节、16 位长的字；在 80386/80486 CPU 及其以后的微处理器中，参加运算的整数操作数还可为 32 位长的双字。整数分为无符号数和有符号数两种。所谓无符号数，是指对应的 8 位、16 位、32 位二进制数全部用来表示数值本身，没有用来表示符号位的位，因而为正整数。

1. 二进制数的算术运算

(1) 加法运算

0+0=0　　0+1=1　　1+0=1　　1+1=0(有进位)

(2) 减法运算

0−0=0　　1−0=1　　1−1=0　　0−1=1(有借位)

(3) 乘法运算

0*0=0　　0*1=0　　1*0=0　　1*1=1

计算机中乘法运算是通过加法和移位运算实现的。

(4) 除法运算

除法是乘法的逆运算，在计算机中是通过减法和右移运算实现的。每右移一位相当于除以 2，右移 n 位就相当于除以 2^n，手工计算时与十进制除法一样。

(5) 无符号二进制数表示范围

一个 n 位无符号整数所表示的数值范围是 $0\sim2^n-1$。例如：8 位无符号整数所表示的数值

范围是0～255；16位无符号整数所表示的数值范围是0～65 535；32位无符号整数所表示的数值范围是0～(4G－1)，其中G为2^{30}。

(6) 无符号二进制数的溢出判断

最高有效位有进位或借位，则产生溢出，运算结果就不对了。

2. 二进制数的逻辑运算

(1) "与"(又称逻辑乘)运算

$$1\wedge 1=1 \quad 1\wedge 0=0 \quad 0\wedge 1=0 \quad 0\wedge 0=0$$

(2) "或"(又称逻辑加)运算

$$0\vee 0=0 \quad 0\vee 1=1 \quad 1\vee 0=1 \quad 1\vee 1=1$$

(3) "非"运算

二进制0做逻辑非运算得1，二进制1做逻辑非运算得0。

(4) "异或"运算

$$0\oplus 0=0 \quad 1\oplus 1=0 \quad 0\oplus 1=1 \quad 1\oplus 0=1$$

1.2.4 有符号数的表示方法

由于计算机只能识别由0和1组成的数或代码，所以有符号数的符号也只能用0和1来表示，一般用"0"表示正，用"1"表示负，这种表示方法将符号数码化了，连同一个符号位在一起的一个数称为机器数。而直接用"＋"号和"－"号来表示其正负的数为有符号数(该机器数)的真值。例如

机器数01010101B，真值：＋85 或 ＋1010101B

机器数11010101B，真值：－85 或 －1010101B

由于数值部分的表示方法不同，所以机器数有3种表示方法：原码、反码和补码。它们具有字节、字及双字3种不同长度的整数类型。

1. 机器数的表示法

(1) 原码

对一个二进制数而言，若用最高位表示数的符号(常以"0"表示正数，以"1"表示负数)，其余各位表示数值本身，则称为该机器数的原码表示法。例如，设$X=+1011100$B，$Y=-1011100$B，则$[X]_{原}=01011100$B，$[Y]_{原}=11011100$B。$[X]_{原}$和$[Y]_{原}$分别为X和Y的原码，称为机器数的原码表示。下面介绍原码$[X]_{原}$和真值X之间的关系：

$$\text{当 } X\geqslant +0 \text{ 时，}[X]_{原}=X$$

$$\text{当 } X\leqslant -0 \text{ 时，}[X]_{原}=2^{n-1}-X$$

其中n为二进制数的位数，最高位为符号位。

值得注意的是，二进制原码表示中有＋0和－0之分，即

$$[+0]_{原}=000\cdots 00\text{B}(n\text{位二进制数，最高位为符号位})$$

$$[-0]_{原}=100\cdots 00\text{B}(n\text{位二进制数，最高位为符号位})$$

(2) 补码

补码的定义源于同余的概念$X+NK=X(\text{mod } K)$，其中K为模，N为任意整数。其含义是数X与该数加上其模的任意整数倍之和相等。

一个二进制数，若以2^n为模(n为二进制数位数，它通常与计算机中机器数的长度一致)，它的补码叫作2补码，后文把2补码简称为补码，即

$$\text{当 } 0\leqslant X\leqslant 2^{n-1}-1 \text{ 时，}[X]_{补}=X$$

$$当 -2^{n-1} \leqslant X \leqslant 0 \text{ 时，} [X]_{补} = 2^n + X$$

同理，一个十进制数若以 10^n 为模，它的补码叫作 10 补码。

可见，正数的补码与原码相同，只有负数才有求补码的问题。

① 根据定义求补码

$[X]_{补} = 2^n + X = 2^n - |X|$，$X < 0$，mod 2^n，即负数 X 的补码等于模 2^n 加上其真值，或者减去其真值的绝对值。例如，$X = -1010111\text{B}$，$n = 8$，则 $[X]_{补} = 2^8 + (-1010111\text{B}) = 100000000\text{B} - 1010111\text{B} = 10101001\text{B}(\text{mod } 2^8)$。

这种方法因为要做一次减法，很不方便。

② 利用原码求补码

一个负数的补码等于其原码除符号位不变外，其余各位按位求反，再在最低位加 1，简称取反加 1。例如，$X = -1010111\text{B}$，则 $[X]_{原} = 11010111\text{B}$，$[X]_{补} = 10101000\text{B} + 1 = 10101001\text{B}$。

如果将 $[X]_{补}$ 再求一次补码，就得到 $[X]_{原}$，即有 $[[X]_{补}]_{补} = [X]_{原}$。例如，$[X]_{原} = 11010111\text{B}$，$[X]_{补} = 10101001\text{B}$，则 $[[X]_{补}]_{补} = 11010110\text{B} + 1 = 11010111\text{B} = [X]_{原}$。

上述求法在数学上都可以证明，在此从略。

③ 简便的直接求补法

直接从原码求它的补码：从最低位起，到出现第一个 1 以前（包括第一个 1）原码中的数字不变，以后逐位取反，但符号位不变。

例 1-6　试求 $X_1 = -1010111\text{B}$，$X_2 = -1110000\text{B}$ 及 $X_3 = +1110001\text{B}$ 的补码 $[X_1]_{补}$、$[X_2]_{补}$ 和 $[X_3]_{补}$。

解：方法自选，在此利用原码求。

$$X_1 = -1010111\text{B}$$

由原码求补码：$[X_1]_{原} = 1\ 1010111\text{B} + 1 \rightarrow [X_1]_{补} = 10101001\text{B}$

（符号位不变；1010111B 取反；末位加1）

同理，$X_2 = -1110000\text{B} \rightarrow [X_2]_{原} = 11110000\text{B} \rightarrow [X_2]_{补} = 10010000\text{B}$。

对于正整数 $X_3 = +1110001\text{B}$，$[X_3]_{原} = 01110001\text{B} = [X_3]_{补}$。

值得注意的是：$[-128]_{补} = 10000000\text{B}$，$[0]_{补} = 00000000\text{B}$。

(3) 反码

若二进制数 $X = X_{n-1}X_{n-2}\cdots X_1X_0$，则反码表示的严格定义如下：

$$[X]_{反} = X, \qquad 2^{n-1} > X \geqslant 0$$

$$[X]_{反} = 2^n - 1 + X, \quad 0 \geqslant X > -2^{n-1}$$

其中，X 表示二进制数真值，其反码记为 $[X]_{反}$，n 表示包括符号位和数值部分在内的二进制数位数。

在求反码时，对正数来讲，其表示方法同原码。但对负数而言，其反码的数值部分为真值的各位按位取反，符号位不变。例如，若 $X = +10010\text{B} = 18$，$Y = -10010\text{B} = -18$，设 n 为 8，则 $[X]_{原} = 00010010\text{B}$，$[Y]_{原} = 10010010\text{B}$，$[X]_{反} = 00010010\text{B}$，$[Y]_{反} = 11101101\text{B}$。

同样，对负数的补码而言，再求反一次会得到其原码。

在反码表示中，数 0 也有两种表示形式：$[+0]_{反} = 00000000\text{B}$，$[-0]_{反} = 11111111\text{B}$。

2. 有符号数的运算

上面介绍了计算机中有符号数的表示方法：原码、补码及反码。用原码表示数时，最大的优点是直观。浮点数的有效数字常用原码表示，进行二进制乘除法运算时，也多采用原码表示法。

计算机中有符号二进制数进行加减运算时采用补码形式。补码进行减法运算可用加法来代替，且符号位也可以和数一起参加运算，这使计算机的运算速度大大提高，同时也简化了计算机

的硬件结构，即$[X\pm Y]_{补}=[X]_{补}+[\pm Y]_{补}$，其中$|X|$、$|Y|$、$|X+Y|<2^{n-1}$。

例 1-7 用补码进行下列运算(设$n=8$)。

①(+20)+(−13)；②(−20)+(+13)；③(−20)+(−13)；④(+20)+(+13)。

解：

①
```
   0001 0100B  [+20]补
 + 1111 0011B  [- 13]补
-------------------------
 1 0000 0111B  [+7]补
```
↑最高位(符号位)为0，结果为正

符号位的进位，自然丢掉

②
```
   1110 1100B  [-20]补
 + 0000 1101B  [+13]补
-------------------------
   1111 1001B  [- 7]补
```
最高位(符号位)为1，结果为负

③
```
   1110 1100B  [-20]补
 + 1111 0011B  [-13]补
-------------------------
 1 1101 1111B  [-33]补
```
↑最高位(符号位)为1，结果为负

符号位的进位，自然丢掉

④
```
   0001 0100B  [+20]补
 + 0000 1101B  [+13]补
-------------------------
   0010 0001B  [+33]补
```
最高位(符号位)为0，结果为正

例 1-8 用补码进行下列运算(设$n=8$)。

① 196−19；②(−56)−(−17)。

解：

① $X=96, Y=19$，则

$[X]_{补}=[X]_{原}=0110\ 0000B$

$[Y]_{补}=[Y]_{原}=0001\ 0011B$

$[-Y]_{补}=1110\ 1101B$

```
  0110 0000B  [X]补
+ 1110 1101B  [-Y]补
-------------------------
  0100 1101B  [X-Y]补=[X-Y]原=+77
```
↑符号位为0，结果为正

② $X=-56, Y=-17$，则

$[X]_{原}=1011\ 1000B$，$[X]_{补}=1100\ 1000B$

$[Y]_{原}=1001\ 0001B$，$[Y]_{补}=1110\ 1111B$

$[-Y]_{补}=0001\ 0001B$

```
  1100 1000B  [X]补
+ 0001 0001B  [-Y]补
-------------------------
  1101 1001B  [X-Y]补
```
符号位为1，结果为负，对$[X-Y]_{补}$再求补，得$[X-Y]_{原}=1010\ 0111B$

前面曾讲过无符号数的运算规则，那计算机中两个无符号数到底是怎样做加、减运算的呢？

两个无符号数进行加法，只要和的绝对值不超过整个字长，就不溢出，则和也一定为正数的补码形式，它等于和的原码；两个无符号数相减，可用减数变补与被减数相加来求得。所谓减数变补，是指将整个减数各位取反后末位加1。

例 1-9 两个无符号数进行下列运算($n=8$)：①129−79；②79−129。

解：

① 设$X=129$，$Y=79$，则

$X=1000\ 0001B$

$Y=0100\ 1111B$

$[-Y]_{补}=[Y]_{变补}=1011\ 0001B$

```
  1000 0001B  X
+ 1011 0001B  [-Y]变补
-------------------------
 1 0011 0010B  [X-Y]补
```
↑

即$[X-Y]_{原}=[X-Y]_{补}=00110010B=50$

② 设$X=79$，$Y=129$，则

$X=0100\ 1111B$

$Y=1000\ 0001B$

$[-Y]_{补}=[Y]_{变补}=0111\ 1111B$

```
  0100 1111B  X
+ 0111 1111B  [-Y]补
-------------------------
 0 1100 1110B  [X-Y]补
```
↑

即$[X-Y]_{补}=11001110B$，$[X-Y]_{原}=10110010B=-50$

此为试读，需要完整PDF请访问：www.ertongbook.com