




普通高等教育“十二五”规划教材  
电子电气基础课程规划教材

# 微机原理与接口技术

## ——基于8086和Proteus仿真

### (第2版)

■ 顾 晖 陈 越 梁惺彦 主编 ■ 包志华 主审

 中国工信出版集团



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
<http://www.phei.com.cn>

电子电气基础课程规划教材

# 微机原理与接口技术

## ——基于 8086 和 Proteus 仿真

### (第 2 版)

顾 晖 陈 越 梁惺彦 主编

鲁 松 华 琇 胡 慧 张 洁 编著

包志华 主审

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

## 内 容 简 介

本书从微型计算机系统应用的角度出发,以 Intel 8086 微处理器和 IBM PC 系列微机为主要对象,系统介绍了微型计算机的基本组成、工作原理、接口技术及应用。本书在总结第 1 版内容的基础上,调整了章节设置,增加了综合实例相关的内容。全书共 13 章,包括:数的表示与运算、8086 微型计算机系统、8086 寻址方式与指令系统、8086 汇编语言程序设计、Proteus 仿真平台的使用、存储器、输入/输出接口、可编程接口芯片、中断与中断管理、数模与模数转换及应用、总线、Proteus ISIS 仿真基础实例和 Proteus ISIS 仿真综合实例。

本书内容全面、实用性强,原理、技术与应用并重,并特别介绍了利用 EDA 工具——Proteus ISIS 的实验方法,讲述有特点和新意。本书在实例讲解方面进一步加强,在保留第 1 版基础实例的基础上,增加了综合实例一章。书中提供的实例全部在 Proteus 中调试通过,设计方案同时适用于实验箱实验的教学方式。

本书可作为高等院校电气与电子信息类专业本科生的教材,也可作为研究生教材或供有关工程技术人员参考使用。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

### 图书在版编目(CIP)数据

微机原理与接口技术:基于 8086 和 Proteus 仿真/顾晖,陈越,梁惺彦主编. —2 版. —北京:电子工业出版社,2015.8  
电子电气基础课程规划教材

ISBN 978-7-121-26616-4

I. ①微… II. ①顾… ②陈… ③梁… III. ①微型计算机—理论—高等学校—教材②微型计算机—接口—高等学校—教材 IV. ①TP36

中国版本图书馆 CIP 数据核字(2015)第 159849 号

责任编辑:凌毅

印 刷:涿州市京南印刷厂

装 订:涿州市京南印刷厂

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本:787×1 092 1/16 印张:19.75 字数:530 千字

版 次:2011 年 8 月第 1 版

2015 年 8 月第 2 版

印 次:2015 年 8 月第 1 次印刷

印 数:4 000 册 定价:39.80 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 [zltz@phei.com.cn](mailto:zltz@phei.com.cn), 盗版侵权举报请发邮件至 [dbqq@phei.com.cn](mailto:dbqq@phei.com.cn)。

服务热线:(010) 88258888。

# 再版前言

“微机原理与接口技术”是高等学校电子信息工程、通信工程、自动化、电气工程及其自动化等工科电气与电子信息类各专业的核心课程。本课程的任务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、接口电路及应用方法，使学生掌握微机系统的开发能力。为了适应教学的需要，编者在总结了多年的教学科研实践经验、对有关微型计算机技术资料进行综合提炼的基础上，编写了本书。

本书特别考虑了内容的选取与组织，注意从微机应用的需求出发，以 Intel 8086 微处理器和 IBM PC 系列微机为主要对象，系统、深入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，把微机系统开发过程中用到的硬件技术和软件技术有机地结合起来。本书在总结第 1 版使用过程中的情况后，调整了章节设置，增加了综合实例，以进一步强化本书的课程实践指导作用。本书合并了第 1 版的第 1、3 章，增加了综合实例一章。全书共 13 章，包括数的表示与运算、8086 微型计算机系统、8086 寻址方式与指令系统、8086 汇编语言程序设计、Proteus 仿真平台的使用、存储器、输入/输出接口、可编程接口芯片、中断与中断管理、数模与模数转换及应用、总线、Proteus ISIS 仿真基础实例和 Proteus ISIS 仿真综合实例。

本书有如下特色：

(1) 内容精练。本书以经典微处理器——Intel 8086 和 IBM PC 系列微机为主要对象，重点突出，内容全面。

(2) 实用性强。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基础实例和综合实例，突出强调了软硬件结合的思维方法和实践动手能力的培养，侧重微机系统的设计。

(3) 实验手段先进。本书介绍了可适用于该课程教学实践的先进的 EDA 工具——Proteus 的用法，并引入大量实例。书中实例全部按照课程内容进行规划，对于同一个问题提供了多种不同的实现方案来解决，可以使学生更好地体会到技术的发展，较好体现了“整体→局部→整体”的知识体系。而且，书中所介绍的实例方案同样适用于在实验箱上进行实验。

(4) 可读性强。书中内容力求文字精练、语言流畅。在内容安排上还注意由浅入深、分散难点。特别是在接口部分，注意形成芯片结构、编程和应用一体化的讲解体系，以便学生理解和应用。

本书的编写采用集体讨论、分工编写、交叉修改的方式进行。本书的第 1、3、4、11、13 章由顾晖编写；第 6、7 由陈越编写；第 5、12 章由梁惺彦编写；第 8 章由华琇编写；第 9 章由胡慧编写；第 2、10 章由张洁编写；附录 A 由鲁松编写。全书由顾晖、陈越统稿并最后定稿。本书定稿后，由包志华教授主审。

本书配有电子课件、源程序包、部分习题解答等教学资源，读者可以登录华信教育资源网 ([www.hxedu.com.cn](http://www.hxedu.com.cn)) 免费下载。

本书的编写工作得到了南通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教师的大力支持；本书的编写还得到了广州风标电子技术公司的大力支持。在本书的编写过程中，广州风标电子技术公司的梁树先和杨炼指导了 Proteus 仿真实例的设计。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领导表示真诚的感谢！

由于编者水平有限，加之时间仓促，书中错误和不当之处在所难免，敬请读者批评指正。

编者

2015年7月

# 目 录

<b>第 1 章 数的表示与运算</b> ..... 1	2.5.4 8086 的寄存器结构..... 33
1.1 数制..... 1	习题 2..... 35
1.1.1 数制的表示..... 1	<b>第 3 章 8086 寻址方式与指令系统</b> ..... 37
1.1.2 数制之间的转换..... 2	3.1 概述..... 37
1.2 二进制数的表示与运算..... 5	3.2 8086 寻址方式..... 37
1.2.1 无符号二进制数的表示..... 5	3.2.1 立即寻址..... 37
1.2.2 无符号二进制数的运算..... 5	3.2.2 寄存器寻址..... 38
1.2.3 带符号二进制数的表示..... 6	3.2.3 直接寻址..... 38
1.2.4 带符号二进制数的运算..... 8	3.2.4 寄存器间接寻址..... 39
1.3 BCD 码的表示与运算..... 9	3.2.5 寄存器相对寻址..... 40
1.3.1 BCD 码的编码方法..... 9	3.2.6 基址变址寻址..... 41
1.3.2 8421BCD 码的加、减运算..... 10	3.2.7 相对基址变址寻址..... 41
1.4 字符的表示..... 12	3.3 8086 指令系统..... 42
习题 1..... 13	3.3.1 数据传送指令..... 43
<b>第 2 章 8086 微型计算机系统</b> ..... 14	3.3.2 算术运算指令..... 49
2.1 概述..... 14	3.3.3 位运算指令..... 55
2.1.1 微型计算机系统的工作原理..... 14	3.3.4 串操作指令..... 59
2.1.2 微型计算机系统的硬件组成..... 14	3.3.5 控制转移指令..... 62
2.2 8086 微处理器的结构..... 16	3.3.6 处理器控制指令..... 67
2.2.1 8086 的内部结构..... 16	习题 3..... 68
2.2.2 8086 的工作模式..... 18	<b>第 4 章 8086 汇编语言程序设计</b> ..... 71
2.3 8086 微处理器的引脚特性..... 18	4.1 汇编语言基础知识..... 71
2.3.1 两种工作模式的公共引脚..... 19	4.1.1 概述..... 71
2.3.2 最小模式下的引脚..... 21	4.1.2 汇编源程序的结构..... 71
2.3.3 最大模式下的引脚..... 21	4.1.3 汇编语言的语句..... 72
2.4 8086 微型计算机系统的总线时序..... 22	4.1.4 汇编语言的数据..... 74
2.4.1 基本概念..... 22	4.1.5 汇编语言的操作符与表达式..... 75
2.4.2 最小模式下的总线周期时序..... 24	4.2 汇编语言的伪指令..... 78
2.4.3 最大模式下的总线周期时序..... 27	4.2.1 变量定义伪指令..... 78
2.5 8086 微型计算机系统的硬件组成与组织..... 29	4.2.2 符号定义伪指令..... 78
2.5.1 8086 微型计算机系统的硬件组成..... 29	4.2.3 段定义伪指令..... 79
2.5.2 8086 微型计算机系统的存储器组织..... 30	4.2.4 过程定义伪指令..... 80
2.5.3 8086 微型计算机系统的 I/O 组织..... 32	4.2.5 模块定义和结束伪指令..... 81
	4.2.6 其他伪指令..... 81
	4.3 系统功能调用..... 82
	4.3.1 DOS 功能调用..... 82
	4.3.2 BIOS 功能调用..... 84

4.4	汇编语言程序设计	84	6.4	存储器与系统的连接	133
4.4.1	程序的质量标准	84	6.4.1	存储器扩展	133
4.4.2	汇编语言程序设计的基本 步骤	85	6.4.2	存储器地址译码方法	134
4.4.3	顺序结构程序设计	85	6.4.3	8086 CPU 与存储器的连接	135
4.4.4	分支结构程序设计	86	习题 6		140
4.4.5	循环结构程序设计	88	<b>第 7 章 输入/输出接口</b>		142
4.4.6	子程序设计	90	7.1	I/O 接口概述	142
4.4.7	汇编语言程序设计举例	93	7.1.1	CPU 与 I/O 设备之间交换的 信息	142
4.5	汇编语言程序的上机过程	98	7.1.2	I/O 接口的主要功能	143
4.5.1	上机环境	98	7.1.3	I/O 接口的结构	143
4.5.2	上机过程	99	7.1.4	输入/输出的寻址方式	144
4.5.3	运行调试	99	7.2	简单 I/O 接口芯片	145
习题 4		102	7.3	CPU 与外设之间的数据传送方式	145
<b>第 5 章 Proteus 仿真平台的使用</b>		104	7.3.1	程序控制方式	146
5.1	Proteus 简介	104	7.3.2	中断方式	148
5.1.1	Proteus ISIS 编辑环境	104	7.3.3	直接存储器存取方式	148
5.1.2	Proteus ARES 编辑环境	104	7.3.4	通道控制方式和 I/O 处理器	149
5.2	Proteus ISIS 基本使用	105	习题 7		149
5.2.1	可视化界面及工具	105	<b>第 8 章 可编程接口芯片</b>		150
5.2.2	基本操作	108	8.1	可编程接口芯片概述	150
5.2.3	元件的查找与选取	109	8.2	可编程并行接口芯片 8255A	150
5.2.4	元件的使用	113	8.2.1	8255A 的内部结构及引脚 功能	150
5.2.5	连线	117	8.2.2	8255A 的工作方式	152
5.2.6	元件标签	119	8.2.3	8255A 的编程	154
5.2.7	器件标注	119	8.2.4	8255A 的应用举例	155
5.2.8	属性分配工具 (PAT)	120	8.3	可编程定时/计数器 8253/8254	160
5.2.9	全局标注器	120	8.3.1	8253 的内部结构及引脚 功能	160
5.3	Proteus ISIS 下 8086 的仿真	121	8.3.2	8253 的工作方式	162
5.3.1	编辑电路原理图	121	8.3.3	8253 的初始化	165
5.3.2	设置外部代码编译器	121	8.3.4	8253 的应用举例	167
5.3.3	添加源代码并选择编译器	124	8.4	可编程串行通信接口芯片 8251A	171
5.3.4	仿真调试	125	8.4.1	串行数据传送方式	171
习题 5		126	8.4.2	传输速率和传送距离	172
<b>第 6 章 存储器</b>		127	8.4.3	同步串行通信与异步串行 通信	172
6.1	半导体存储器的分类	127	8.4.4	通用可编程串行通信接口 芯片 8251A	173
6.1.1	RAM 的分类	128	习题 8		182
6.1.2	ROM 的分类	128			
6.2	半导体存储器的主要技术 指标	129			
6.3	典型存储器芯片介绍	129			

<b>第 9 章 中断与中断管理</b> .....	184	11.3 外部总线	238
9.1 中断的概念	184	11.3.1 RS-232C 串行总线	238
9.1.1 中断与中断源	184	11.3.2 USB 总线	241
9.1.2 中断系统的功能	184	习题 11	244
9.1.3 简单的中断处理过程	185	<b>第 12 章 Proteus ISIS 仿真基础实例</b> .....	245
9.1.4 中断源识别及优先级判断	188	12.1 基本 I/O 应用——I/O 译码	245
9.2 8086 的中断系统	190	12.1.1 功能说明	245
9.2.1 8086 的中断类型	190	12.1.2 Proteus 电路设计	245
9.2.2 中断向量和中断向量表	192	12.1.3 代码设计	248
9.2.3 8086 中的中断响应和处理过程	195	12.1.4 仿真分析与思考	248
9.3 可编程中断控制器 8259A	197	12.2 定时/计数器 8253 的应用—— 波形发生器	249
9.3.1 8259A 的结构	197	12.2.1 功能说明	249
9.3.2 8259A 的引脚	199	12.2.2 Proteus 电路设计	249
9.3.3 8259A 的中断处理过程	200	12.2.3 代码设计	251
9.3.4 8259A 的工作方式	201	12.2.4 仿真分析与思考	252
9.3.5 8259A 的编程与应用	203	12.3 并行接口芯片 8255A 的应用—— 键盘与数码管	253
9.4 中断程序设计	210	12.3.1 功能说明	253
9.4.1 中断设计方法	210	12.3.2 Proteus 电路设计	253
9.4.2 中断程序设计举例	211	12.3.3 代码设计	254
习题 9	216	12.3.4 仿真分析与思考	256
<b>第 10 章 数模与模数转换及应用</b> .....	217	12.4 中断应用——8259A 芯片的使用	257
10.1 物理信号到电信号的转换	217	12.4.1 功能说明	257
10.1.1 概述	217	12.4.2 Proteus 电路设计	257
10.1.2 几种常见的传感器	217	12.4.3 代码设计	258
10.2 数模转换及应用	218	12.4.4 仿真分析与思考	259
10.2.1 数模转换器的基本原理	218	12.5 模数转换——ADC0808 的使用	260
10.2.2 数模转换器的性能参数	221	12.5.1 功能说明	260
10.2.3 8 位 D/A 转换器 DAC0832	222	12.5.2 Proteus 电路设计	260
10.3 模数转换及应用	225	12.5.3 代码设计	262
10.3.1 模数转换器的基本原理	225	12.5.4 仿真分析与思考	264
10.3.2 模数转换器的性能参数	226	12.6 数模转换——DAC0832 的使用	264
10.3.3 8 位 A/D 转换器 ADC0808/ 0809	227	12.6.1 功能说明	264
习题 10	232	12.6.2 Proteus 电路设计	264
<b>第 11 章 总线</b> .....	233	12.6.3 代码设计	265
11.1 总线的概念	233	12.6.4 仿真分析与思考	266
11.2 系统总线	234	12.7 串行通信——8251A 的使用	266
11.2.1 ISA 总线	234	12.7.1 功能说明	266
11.2.2 EISA 总线	236	12.7.2 Proteus 电路设计	266
11.2.3 PCI 总线	236	12.7.3 代码设计	268



12.7.4	仿真分析与思考	269	13.3.3	代码设计	285
<b>第 13 章</b>	<b>Proteus ISIS 仿真综合实例</b>	271	13.3.4	仿真分析与思考	290
13.1	花式跑马灯	271	13.4	电机转向和转速的控制	290
13.1.1	功能说明	271	13.4.1	功能说明	290
13.1.2	Proteus 电路设计	271	13.4.2	Proteus 电路设计	291
13.1.3	代码设计	272	13.4.3	代码设计	292
13.1.4	仿真分析与思考	276	13.4.4	仿真分析与思考	293
13.2	电子秒表	277	13.5	电子琴及乐曲播放器	294
13.2.1	功能说明	277	13.5.1	功能说明	294
13.2.2	Proteus 电路设计	277	13.5.2	Proteus 电路设计	294
13.2.3	代码设计	278	13.5.3	代码设计	295
13.2.4	仿真分析与思考	283	13.5.4	仿真分析与思考	301
13.3	电压报警器	283	习题 13		301
13.3.1	功能说明	283	<b>附录 A</b>	<b>VSM 仿真的元件库</b>	305
13.3.2	Proteus 电路设计	284	<b>参考文献</b>		307

# 第 1 章 数的表示与运算

## 1.1 数制

### 1.1.1 数制的表示

#### 1. 计数制

数制也称为计数制，是指用一组固定的数字符号和统一的规则表示数的方法。对于任意  $r$  进制数  $x$ ，可以用下式表示为

$$\sum_{i=-m}^n a_i r^i = a_{-m} r^{-m} + \cdots + a_{-2} r^{-2} + a_{-1} r^{-1} + a_0 r^0 + a_1 r^1 + \cdots + a_n r^n$$

其中：

①  $a_i$  为数码，每一种进制数都由固定的数字符号来表示，这个符号称为：数码。

二进制有 2 个数码：0 和 1；

八进制有 8 个数码：0、1、2、3、4、5、6 和 7；

十进制有 10 个数码：0、1、2、3、4、5、6、7、8 和 9；

十六进制有 16 个数码：0、1、2、3、4、5、6、7、8、9、A、B、C、D、E 和 F（字母不区分大小写）。

②  $i$  为数位，数位是指数码在一个数中所处的位置。

例如，十六进制数 56.78 从左到右的数位分别为：1、0、-1 和 -2。

③  $r$  为基数，基数是指在某计数制中，每个数位上能使用的数码的个数。

二进制基数为 2；

八进制基数为 8；

十进制基数为 10；

十六进制基数为 16。

④  $r^i$  为权，权是基数的幂，这个幂次由数位决定。

二进制第  $i$  位上的权为  $2^i$ ；

八进制第  $i$  位上的权为  $8^i$ ；

十进制第  $i$  位上的权为  $10^i$ ；

十六进制第  $i$  位上的权为  $16^i$ 。

例如，十六进制数 56.78 从左到右每一位的权分别为： $16^1$ 、 $16^0$ 、 $16^{-1}$  和  $16^{-2}$ 。

#### 2. 计算机中常用的计数制

在日常生活中，人们最常用的是十进制计数制。而在计算机中，为了便于数的存储和表示，使用的是二进制计数制。由于二进制数据书写和记忆不方便，在计算机系统中还常使用八进制和十六进制等计数制。计算机中常用计数制的属性见表 1-1。

※ 说明：

• 为了区别所使用的数制，一般用以下两种书写格式表示：

① 用括号将数字括起，后面加数制区分，数制用下标的形式给出；

表 1-1 计算机中常用计数制

数制	基数	数码	运算规则	书写后缀
二进制	2	0, 1	逢二进一, 借一当二	B
八进制	8	0, 1, 2, 3, 4, 5, 6, 7	逢八进一, 借一当八	O 或 Q
十进制	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	逢十进一, 借一当十	D
十六进制	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	逢十六进一, 借一当十六	H

② 用后缀区分, 二进制数、十进制数、八进制数、十六进制数的后缀分别为字母 B (或 b)、D (或 d)、O (或 o) 或 Q (或 q)、H (或 h)。

例如: 十六进制数 56.78 可以表示成 $(56.78)_{16}$  或 56.78H;

十进制数 56.78 可以表示成 $(56.78)_{10}$  或 56.78D。

• 汇编程序规定, 使用首字符是字母的十六进制数时, 前面需加 0 来表示。

例如: 0B56.A8H、0FFH 等。

• 在没有混淆的情况下, 十进制数可以省略后缀 D (或 d)。

## 1.1.2 数制之间的转换

### 1. 其他数制数转换为十进制数

二进制数、八进制数和十六进制数转换为十进制数的方法是: “按权展开”。

【例 1-1】将 1010.101B、23.4Q 和 FA3.4H 转换成十进制数。

解  $1010.101B = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 10.625D$

$23.4Q = 2 \times 8^1 + 3 \times 8^0 + 4 \times 8^{-1} = 19.5D$

$FA3.4H = 15 \times 16^2 + 10 \times 16^1 + 3 \times 16^0 + 4 \times 16^{-1} = 4003.25D$

### 2. 十进制数转换为其他数制数

把十进制数转换为其他数制数的方法很多, 通常采用的方法有降幂法和乘法。

#### (1) 降幂法

假设要转换的十进制数为  $N$ 。

步骤 1: 找出最接近  $N$  并小于等于  $N$  的  $r$  进制位权值  $r^i$ ;

步骤 2: 找到满足  $0 \leq C < r$  的最大数  $C$ , 使得  $N - C \times r^i < r^i$ ,  $C$  即为转换结果 ( $r$  进制数) 第  $i$  位的位码  $a_i$ ;

步骤 3: 计算  $N - C \times r^i$ , 并用此值作为新的  $N$  值, 即  $N \leftarrow N - C \times r^i$ ;

步骤 4:  $i$  自减 1, 即  $i \leftarrow i - 1$ , 得到下一个位权值  $r^i$ 。

重复步骤 2~步骤 4, 直至  $N$  为 0 或转换结果达到所需精度。

#### (2) 乘法

采用乘法把十进制数转换为二、八或十六进制数的过程是: 待转换的数的整数部分除以基数取余, 直至商为 0; 小数部分乘以基数取整, 直至积为整数或转换结果的小数位数达到所需精度要求。

【例 1-2】把十进制数 117.8125 转换成二进制数。

解 方法一: 降幂法

小于该数 117.8125 的位权值有: 64、32、16、8、4、2、1、0.5、0.25、0.125 和 0.0625, 按下列过程求出每位的位码。

$N$	$C$	$r^j$	$=$		$a_i$	
117	-1	$2^6$	=	53	$(a_6=1)$	高位
53	-1	$2^5$	=	21	$(a_5=1)$	
21	-1	$2^4$	=	5	$(a_4=1)$	
5	-0	$2^3$	=	5	$(a_3=0)$	
5	-1	$2^2$	=	1	$(a_2=1)$	
1	-0	$2^1$	=	1	$(a_1=0)$	
1	-1	$2^0$	=	0	$(a_0=1)$	
0.8125	-1	$2^{-1}$	=	0.3125	$(a_{-1}=1)$	
0.3125	-1	$2^{-2}$	=	0.0625	$(a_{-2}=1)$	
0.0625	-0	$2^{-3}$	=	0.0625	$(a_{-3}=0)$	
0.0625	-1	$2^{-4}$	=	0	$(a_{-4}=1)$	低位



根据上述过程，可求得  $117.8125D=1110101.1101B$ 。

方法二：乘除法

运算过程如下：

整数部分：	商	余数	
$117 \div 2 = 58$	$\rightarrow$	1	整数部分低位
$58 \div 2 = 29$	$\rightarrow$	0	
$29 \div 2 = 14$	$\rightarrow$	1	
$14 \div 2 = 7$	$\rightarrow$	0	
$7 \div 2 = 3$	$\rightarrow$	1	
$3 \div 2 = 1$	$\rightarrow$	1	
$1 \div 2 = 0$	$\rightarrow$	1	整数部分高位

小数部分：	积	整数	
$0.8125 \times 2 = 1.625$	$\rightarrow$	1	小数部分高位
$0.625 \times 2 = 1.25$	$\rightarrow$	1	
$0.25 \times 2 = 0.5$	$\rightarrow$	0	
$0.5 \times 2 = 1.0$	$\rightarrow$	1	小数部分低位



※ 说明：

- 整数部分取余数时，先得到的数值是转换结果整数部分的低位，后得到的是转换结果整数部分的高位；
- 小数部分取整时，先得到的数值是转换结果小数数部分的高位，后得到的是转换结果小数数部分的低位。

根据上述过程，可求得： $117.8125D=111 0101.1101B$ 。

【例 1-3】把十进制数 48956 转换成十六进制数。

解 方法一：降幂法

小于该数 48956 的位权值有：4096、256、16 和 1，按下列过程求出每位的位码。

$N$	$C$	$r^j$	$=$	$a_i$		
48956	-11	$16^3$	=	3900	$(a_3=B, 11 \text{ 的十六进制数码为 } B)$	高位
3900	-15	$16^2$	=	60	$(a_2=F, 15 \text{ 的十六进制数码为 } F)$	
60	-3	$16^1$	=	12	$(a_1=3, 3 \text{ 的十六进制数码为 } 3)$	
12	-12	$16^0$	=	0	$(a_0=C, 11 \text{ 的十六进制数码为 } C)$	低位



根据上述过程，可求得  $48956D = BF3CH$ 。

方法二：乘除法

商	余数		
$48956 \div 16 = 3059$	$\rightarrow 12$	(对应的十六进制数码为 C)	低位
$3059 \div 16 = 191$	$\rightarrow 3$	(对应的十六进制数码为 3)	↑
$191 \div 16 = 11$	$\rightarrow 15$	(对应的十六进制数码为 F)	↑
$11 \div 16 = 0$	$\rightarrow 11$	(对应的十六进制数码为 B)	高位

根据上述过程，可求得  $48956D = BF3CH$ 。

### 3. 其他数制之间的转换

#### (1) 二进制数与八进制数之间的转换

由于八进制数以 8 为基数，而  $8=2^3$ ，所以 3 位二进制数对应 1 位八进制数，对应关系见表 1-2。

表 1-2 二进制数与八进制数对应表

二进制数	000	001	010	011	100	101	110	111
八进制数	0	1	2	3	4	5	6	7

二进制数转换为八进制数的转换过程是：以小数点为界，整数部分向左，小数部分向右，每 3 位二进制数为一组，用 1 位八进制数表示；不足 3 位的，整数部分高位补 0，小数部分低位补 0。

八进制数转换为二进制数的过程与上述过程相反，把每位八进制数用 3 位二进制数表示即可。

**【例 1-4】** 把数  $11010.101B$  转换为八进制数。

$$11010.101B = \underline{011} \underline{010} . \underline{101}B = 32.5Q$$

**【例 1-5】** 把数  $34.56Q$  转换为二进制数。

$$34.56Q = \underline{011} \underline{100} . \underline{101} \underline{110}B = 11100.10111B$$

#### (2) 二进制数与十六进制数之间的转换

由于十六进制数以 16 为基数，而  $16=2^4$ ，所以 4 位二进制数对应 1 位十六进制数，对应关系见表 1-3。

表 1-3 二进制数与十六进制数对应表

二进制数	0000	0001	0010	0011	0100	0101	0110	0111
十六进制数	0	1	2	3	4	5	6	7
二进制数	1000	1001	1010	1011	1100	1101	1110	1111
十六进制数	8	9	A	B	C	D	E	F

二进制数转换为十六进制数的过程是：以小数点为界，整数部分向左，小数部分向右，每 4 位二进制数为一组，用 1 位十六进制数表示；不足 4 位的，整数部分高位补 0，小数部分低位补 0。

十六进制数转换为二进制数时的过程与上述过程相反，把每位十六进制数用 4 位二进制数表示即可。

**【例 1-6】** 把二进制数  $11010.101B$  转换为十六进制数。

$$11010.101B = \underline{0001} \underline{1010} . \underline{1010}B = 1A.AH$$

【例 1-7】把十六进制数 56.78H 转换为二进制数。

$$56.78H = 0101\ 0110.0111\ 1000B = 1010110.01111B$$

## 1.2 二进制数的表示与运算

任意一个二进制数  $x$  都可以表示为：

$$x = (-1)^S \times M \times 2^E$$

式中， $S$  表示符号位， $S=0$  时， $x$  为正数； $S=1$  时， $x$  为负数。 $M$  是尾数， $E$  是阶码。当  $E$  是固定值时，数  $x$  的小数点位置固定，称为定点数；当  $E$  的值可变时，数  $x$  的小数点位置是浮动的，称为浮点数。

计算机中，常用的定点数有：纯整数和纯小数。本书只涉及定点纯整数的表示与运算。关于浮点数等相关知识，读者可以自行参考《计算机组成原理》等课程的相关内容进行学习。

计算机处理的数包括带符号数和无符号数两种类型。无符号数不分正负，带符号数有正数和负数之分。计算机中对于无符号数和带符号数的处理方法是不同的。读者在处理数时，要注意区分。

带符号数的二进制格式中包括符号位和数值位两部分，通常用最高位作为符号位。带符号数连同符号位在内的数值化表示形式称为机器数，而这个数本身的值称为真值。而带符号数的运算则根据编码方式的不同，有不同的运算规则。

### 1.2.1 无符号二进制数的表示

在某些情况下，计算机要处理的数全是正数，此时不需要考虑符号位的表示问题，这样的数称为无符号数。无符号数的二进制格式中的数位都是数值位。8 位无符号整数的表数范围是：0~255D，16 位无符号整数的表数范围是：0~65535D。

在计算机中，无符号整数常用来表示地址。

### 1.2.2 无符号二进制数的运算

在计算机中，无符号数的运算采用二进制数的算术和逻辑运算规则进行运算。

#### 1. 算术运算规则

二进制数的算术运算包括加法、减法、乘法和除法 4 种，运算规则见表 1-4。

表 1-4 二进制数的算术运算规则

运算名	运算符	运算规则	说明
加法	+	$0+0=0$ $1+0=1$ $0+1=1$ $1+1=10$	逢二进一
减法	-	$0-0=0$ $1-0=1$ $0-1=1$ $1-1=0$	借一当二
乘法	×	$0 \times 0=0$ $0 \times 1=0$ $1 \times 0=0$ $1 \times 1=1$	
除法	÷	$0 \div 1=0$ $1 \div 1=1$	除数不得为 0

#### 2. 逻辑运算规则

常用的逻辑运算有与、或、非和异或 4 种，运算规则见表 1-5。

表 1-5 二进制数常用的逻辑运算规则

运算名	运算符	运算规则
与 (AND)	$\wedge$	$0 \wedge 0=0$ $0 \wedge 1=0$ $1 \wedge 0=0$ $1 \wedge 1=1$
或 (OR)	$\vee$	$0 \vee 0=0$ $0 \vee 1=1$ $1 \vee 0=1$ $1 \vee 1=1$
非 (NOT)	$-$	$\bar{0}=1$ $\bar{1}=0$
异或 (XOR)	$\oplus$	$0 \oplus 0=0$ $0 \oplus 1=1$ $1 \oplus 0=1$ $1 \oplus 1=0$

【例 1-8】无符号二进制数的算术运算举例。

$$1010\ 1010\text{B} + 0101\ 1101\text{B} = 1\ 0000\ 0111\text{B}$$

$$1010\ 1010\text{B} - 0101\ 1101\text{B} = 0100\ 1101\text{B}$$

【例 1-9】无符号二进制数的逻辑运算举例。

$$1010\ 1010\text{B} \wedge 0101\ 1101\text{B} = 0000\ 1000\text{B}$$

$$1010\ 1010\text{B} \vee 0101\ 1101\text{B} = 1111\ 1111\text{B}$$

$$1010\ 1010\text{B} \oplus 0101\ 1101\text{B} = 1111\ 0111\text{B}$$

### 1.2.3 带符号二进制数的表示

机器数可以有多种不同的编码表示方法，常见的编码方式有：原码、反码和补码。

#### 1. 原码

原码表示法规定：最高位是符号位，用 0 表示正数，用 1 表示负数。数值部分用该数的二进制绝对值表示。

数  $x$  的原码记作  $[x]_{\text{原}}$ ，如机器字长为  $n$ ，则整数原码的定义如下：

$$[x]_{\text{原}} = \begin{cases} x & 0 \leq x \leq 2^{n-1} - 1 \\ 2^{n-1} + |x| & -(2^{n-1} - 1) \leq x \leq 0 \end{cases}$$

当机器字长  $n=8$  时，有：

$$[+0\text{D}]_{\text{原}} = 0000\ 0000, \quad [-0\text{D}]_{\text{原}} = 1000\ 0000$$

$$[+1\text{D}]_{\text{原}} = 0000\ 0001, \quad [-1\text{D}]_{\text{原}} = 1000\ 0001$$

$$[+45\text{D}]_{\text{原}} = 0010\ 1101, \quad [-45\text{D}]_{\text{原}} = 1010\ 1101$$

$$[+127\text{D}]_{\text{原}} = 0111\ 1111, \quad [-127\text{D}]_{\text{原}} = 1111\ 1111$$

当机器字长  $n=16$  时，有：

$$[+0\text{D}]_{\text{原}} = 0000\ 0000\ 0000\ 0000, \quad [-0\text{D}]_{\text{原}} = 1000\ 0000\ 0000\ 0000$$

$$[+1\text{D}]_{\text{原}} = 0000\ 0000\ 0000\ 0001, \quad [-1\text{D}]_{\text{原}} = 1000\ 0000\ 0000\ 0001$$

$$[+45\text{D}]_{\text{原}} = 0000\ 0000\ 0010\ 1101, \quad [-45\text{D}]_{\text{原}} = 1000\ 0000\ 0010\ 1101$$

$$[+32767\text{D}]_{\text{原}} = 0111\ 1111\ 1111\ 1111, \quad [-32767\text{D}]_{\text{原}} = 1111\ 1111\ 1111\ 1111$$

按照定义，设  $n$  为字长，则原码的表数范围是： $-(2^{n-1}-1) \sim +(2^{n-1}-1)$ 。

例如，8 位二进制原码的表数范围是： $-127\text{D} \sim +127\text{D}$ ，16 位二进制原码的表数范围是： $-32767\text{D} \sim +32767\text{D}$ 。

原码表示法简单直观，与真值之间的转换方便，但由于符号位不能参与运算，而且对于数 0 有 +0 和 -0 两种表示形式，所以用它进行运算不方便。

#### 2. 反码

反码表示法规定：一个正数的反码和原码相同；一个负数的反码的符号位与其原码的符号位相同，数值位通过对其原码的数值部分按位求反得到。

数  $x$  的反码记作  $[x]_{\text{反}}$ ，如机器字长为  $n$ ，则整数反码的定义如下：

$$[x]_{\text{反}} = \begin{cases} x & 0 \leq x \leq 2^{n-1} - 1 \\ (2^n - 1) - |x| & -(2^{n-1} - 1) \leq x \leq 0 \end{cases}$$

当机器字长  $n=8$  时，有：

$$\begin{aligned} [+0D]_{\text{反}} &= 0000\ 0000, & [-0D]_{\text{反}} &= 1111\ 1111 \\ [+1D]_{\text{反}} &= 0000\ 0001, & [-1D]_{\text{反}} &= 1111\ 1110 \\ [+45D]_{\text{反}} &= 0010\ 1101, & [-45D]_{\text{反}} &= 1101\ 0010 \\ [+127D]_{\text{反}} &= 0111\ 1111, & [-127D]_{\text{反}} &= 1000\ 0000 \end{aligned}$$

当机器字长  $n=16$  时，有

$$\begin{aligned} [+0D]_{\text{反}} &= 0000\ 0000\ 0000\ 0000, & [-0D]_{\text{反}} &= 1111\ 1111\ 1111\ 1111 \\ [+1D]_{\text{反}} &= 0000\ 0000\ 0000\ 0001, & [-1D]_{\text{反}} &= 1111\ 1111\ 1111\ 1110 \\ [+45D]_{\text{反}} &= 0000\ 0000\ 0010\ 1101, & [-45D]_{\text{反}} &= 1111\ 1111\ 1101\ 0010 \\ [+32767D]_{\text{反}} &= 0111\ 1111\ 1111\ 1111, & [-32767D]_{\text{反}} &= 1000\ 0000\ 0000\ 0000 \end{aligned}$$

按照定义，设  $n$  为字长，则反码的表数范围是： $-(2^{n-1}-1) \sim +(2^{n-1}-1)$ 。

例如，8 位二进制反码的表数范围是  $-127D \sim +127D$ ，16 位二进制反码的表数范围是  $-32767D \sim +32767D$ 。

反码与原码的表数范围相同，而且数 0 有 +0 和 -0 两种表示形式。所以，用反码进行运算也不方便。

根据反码求真值的方法是：若反码的最高位为 0，则该数是正数，其后的数值部分就是其真值；若反码的最高位为 1，则该数是负数，将其后的数值部分按位取反后，即可得到真值。

### 3. 补码

补码表示法规定：一个正数的补码和反码、原码相同；一个负数的补码的符号位与其原码的符号位相同，其余位可通过将其反码数值部分加 1 得到（但有一个负数例外，详见表 1-6 下面的说明）。

数  $x$  的补码记作  $[x]_{\text{补}}$ ，如机器字长为  $n$ ，则整数补码的定义如下：

$$[x]_{\text{补}} = \begin{cases} x & 0 \leq x < 2^{n-1} - 1 \\ 2^n - |x| & -(2^{n-1} - 1) \leq x < 0 \end{cases}$$

当机器字长  $n=8$  时，有：

$$\begin{aligned} [+0D]_{\text{补}} &= 0000\ 0000, & [-0D]_{\text{补}} &= 0000\ 0000 \\ [+1D]_{\text{补}} &= 0000\ 0001, & [-1D]_{\text{补}} &= 1111\ 1111 \\ [+45D]_{\text{补}} &= 0010\ 1101, & [-45D]_{\text{补}} &= 1101\ 0011 \\ [+127D]_{\text{补}} &= 0111\ 1111, & [-127D]_{\text{补}} &= 1000\ 0001 \end{aligned}$$

按照定义，设  $n$  为字长，则补码能表示的整数范围是： $-2^{n-1} \sim +(2^{n-1}-1)$ 。

例如，8 位二进制补码表示的整数范围是  $-128D \sim +127D$ ，16 位二进制补码表示的整数范围是  $-32768D \sim +32767D$ 。

补码比原码、反码所能表示的数的范围大，数 0 的补码只有一种表示形式。

根据补码求真值的方法是：若补码的最高位为 0，则该数是正数，其后的数值部分就是其真值；若反码的最高位为 1，则该数是负数，将其后的数值部分按位取反加 1 后，即可得到真值（但有一个负数例外，详见表 1-6 下面的说明）。



表 1-6 是部分 8 位二进制数的原码、反码和补码表示形式的对照表。

表 1-6 部分 8 位二进制数的原码、反码和补码对照表

真值		带符号数		
十进制格式	二进制数格式	原码	反码	补码
0	0000 0000	0000 0000	0000 0000	0000 0000
1	0000 0001	0000 0001	0000 0001	0000 0001
...	...	...	...	...
+126	0111 1110	0111 1110	0111 1110	0111 1110
+127	0111 1111	0111 1111	0111 1111	0111 1111
-128	-1000 0000	无	无	1000 0000
-127	-0111 1111	1111 1111	1000 0000	1000 0001
...	...	...	...	...
-1	-0000 0001	1000 0001	1111 1110	1111 1111
-0	-0000 0000	1000 0000	1111 1111	0000 0000

特别说明：-128D 没有原码和反码，其补码通过定义式求得。

## 1.2.4 带符号二进制数的运算

计算机中普遍采用补码来表示带符号数。补码的算术运算包括加减运算和乘除运算，本书仅讨论补码的加减运算。补码的逻辑运算规则与 1.2.2 节介绍的无符号数的逻辑运算规则相同。

### 1. 补码的加减运算规则

采用补码表示的带符号数进行加减运算时，符号位和数值位同时参与运算，运算结果仍然是补码。任何两数相加，无论正负，只要把它们的补码相加即可。加法运算得到的结果是两数和的补码。任何两数相减，无论正负，只要把减数相反数的补码与被减数的补码相加即可。减法运算结果是两数差的补码。运算公式如下：

$$[x+y]_{\text{补}} = [x]_{\text{补}} + [y]_{\text{补}}$$

$$[x-y]_{\text{补}} = [x]_{\text{补}} + [-y]_{\text{补}}$$

从上面的公式可以看出，补码的减法运算可以转换成加法来完成的，因此，在计算机中利用加法器就可以实现补码的加法和减法运算。

### 2. 补码加减运算的溢出判断

由于计算机的字长有限，因此，所能表示的数是有范围的。当运算结果超过这个范围时，运算结果将出错，这种情况称为溢出。产生溢出的原因是：数值的有效位占据了符号位。补码加减运算溢出的判定有以下两种方法。

(1) 利用符号位判别运算结果是否溢出

- 若两个同号数相加，结果的符号位与之相反，则溢出；
- 若两个异号数相减，结果的符号位与减数相同，则溢出；
- 若两个异号数相加或两个同号数相减，则不溢出。

(2) 利用运算过程中的进位产生情况判别运算结果是否溢出

- 若次高位(最高数值位)和最高位(符号位)不同时产生进位或借位，则溢出；
- 若次高位(最高数值位)和最高位(符号位)都产生进位或借位，则不溢出。