



21世纪高职高专规划教材

计算机系列

微型计算机原理 与接口技术

李兰友 万振凯 李静东 编著



清华大学出版社

<http://www.tup.tsinghua.edu.cn>



北方交通大学出版社

<http://press.njtu.edu.cn>



21 世纪高职高专规划教材·计算机系列

微型计算机原理与接口技术

李兰友 万振凯 李静东 编著

清华大学出版社

北方交通大学出版社

·北京·

内 容 简 介

本书是根据“计算机应用技术”专业“微型计算机原理与接口技术”课程教学大纲编写的。全书共9章。第1~6章介绍微型计算机基础、微处理器、指令与指令系统、汇编语言程序设计、总线、存储器等有关微型计算机原理的基础知识；第7~9章介绍微型计算机数据传送方法，串、并行通信及接口，常用的控制接口，磁盘、光盘接口，人机交互接口等有关微型计算机接口的知识及实用技术。全书各章有学习要点提示、例题解析和练习题；附录A为本书的实验指导书，附录B为模拟试题及答案。全书简明易懂，突出概念和实用技术。

本书可作为高等职业技术教育、高等教育自学考试、中等专科教育“微型计算机原理与接口技术”课程的教材，亦可供工程技术人员参考。

版权所有，翻印必究。

本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

图书在版编目 (CIP) 数据

微型计算机原理与接口技术 / 李兰友, 万振凯, 李静东编著. —北京: 北方交通大学出版社, 2003. 11

(21世纪高职高专规划教材·计算机系列)

ISBN 7-81082-170-9

I. 微… II. ①李… ②万… ③李… III. ①微型计算机—理论—高等学校: 技术学校—教材②微型计算机—接口设备—高等学校: 技术学校—教材 IV. TP36

中国版本图书馆 CIP 数据核字 (2003) 第 064227 号

责任编辑: 谭文芳 特邀编辑: 林欣

印刷者: 北京东光印刷厂

出版发行: 北方交通大学出版社 邮编: 100044 电话: 010-51686045, 62237564

清华大学出版社 邮编: 100084

经 销: 各地新华书店

开 本: 787×1092 1/16 印张: 14.5 字数: 371 千字

版 次: 2003年11月第1版 2003年11月第1次印刷

印 数: 5000册 定价: 21.00元

21 世纪高职高专规划教材·计算机系列
编审委员会成员名单

主任委员	李兰友	边奠英			
副主任委员	周学毛	崔世钢	王学彬	丁桂芝	陈跃安
	赵伟	韩瑞功	汪志达		
委员	(按姓名笔画排序)				
	马辉	万志平	万振凯	王永平	王建明
	丰继林	左文忠	叶华	叶伟	付慧生
	江中	刘炜	刘建民	刘晶	曲建民
	孙培民	邢素萍	华铨平	吕新平	陈小东
	陈月波	李长明	李可	李志奎	李琳
	李源生	李群明	李静东	邱希春	沈才梁
	宋维堂	汪玉华	汪繁	张文明	张权范
	张宝忠	张爱娟	张琦	金忠伟	林长春
	林文信	苗长云	竺士蒙	周智仁	孟德欣
	柏万里	宫国顺	柳炜	胡敬佩	姚策
	赵英杰	高娟	高福成	贾建军	徐建俊
	殷兆麟	唐健	黄斌	章春军	曹豫莪
	程琪	韩其睿	韩劼	裘旭光	童爱红
	谢婷	曾瑶辉	管致锦	熊锡义	潘玫玫
	薛永三	操静涛	鞠洪尧		

出版说明

高职高专教育是我国高等教育的重要组成部分，它的根本任务是培养生产、建设、管理和服务第一线需要的德、智、体、美全面发展的高等技术应用型专门人才，所培养的学生在掌握必要的基础理论和专业知识的基础上，应重点掌握从事本专业领域实际工作的基本知识和职业技能，因而与其对应的教材也必须有自己的体系和特色。

为了适应我国高职高专教育发展及其对教学改革和教材建设的需要，在教育部的指导下，我们在全国范围内组织并成立了“21世纪高职高专教育教材研究与编审委员会”（以下简称“教材研究与编审委员会”）。“教材研究与编审委员会”的成员单位皆为教学改革成效较大、办学特色鲜明、办学实力强的高等专科学校、高等职业学校、成人高等学校及高等院校主办的二级职业技术学院，其中一些学校是国家重点建设的示范性职业技术学院。

为了保证规划教材的出版质量，“教材研究与编审委员会”在全国范围内选聘“21世纪高职高专规划教材编审委员会”（以下简称“教材编审委员会”）成员和征集教材，并要求“教材编审委员会”成员和规划教材的编著者必须是从事高职高专教学第一线的优秀教师或生产第一线的专家。“教材编审委员会”组织各专业的专家、教授对所征集的教材进行评选，对列选教材进行审定。

目前，“教材研究与编审委员会”计划用2~3年的时间出版各类高职高专教材200种，范围覆盖计算机应用、电子电气、财会与管理、商务英语等专业的主要课程。此次规划教材全部按教育部制定的“高职高专教育基础课程教学基本要求”编写，其中部分教材是教育部《新世纪高职高专教育人才培养模式和教学内容体系改革与建设项目计划》的研究成果。此次规划教材编写按照突出应用性、实践性和针对性的原则编写并重组系列课程教材结构，力求反映高职高专课程和教学内容体系改革方向；反映当前教学的新内容，突出基础理论知识的应用和实践技能的培养；适应“实践的要求和岗位的需要”，不依照“学科”体系，即贴近岗位群，淡化学科；在兼顾理论和实践内容的同时，避免“全”而“深”的面面俱到，基础理论以应用为目的，以必要、够用为度；尽量体现新知识、新技术、新工艺、新方法，以利于学生综合素质的形成和科学思维方式与创新能力的培养。

此外，为了使规划教材更具广泛性、科学性、先进性和代表性，我们希望全国从事高职高专教育的院校能够积极加入到“教材研究与编审委员会”中来，推荐“教材编审委员会”成员和有特色、有创新的教材。同时，希望将教学实践中的意见与建议及时反馈给我们，以便对已出版的教材不断修订、完善，不断提高教材质量，完善教材体系，为社会奉献更多更新的与高职高专教育配套的高质量教材。

此次所有规划教材由全国重点大学出版社——清华大学出版社与北方交通大学出版社联合出版。适合于各类高等专科学校、高等职业学校、成人高等学校及高等院校主办的二级职业技术学院使用。

21世纪高职高专教育教材研究与编审委员会
2003年9月

前 言

自1981年IBM公司的通用微型计算机IBM PC/XT问世以来,在短短的20多年间,微型计算机一直以令人目不暇接的态势飞速发展。微型计算机的速度越来越快、性能越来越高、应用范围越来越广、对社会和人类文明的发展影响越来越大。因此,可以说学习微型计算机的基础知识,掌握微型计算机的基本使用技术,已成为现代社会人才应具备的基本素质之一。

微型计算机是以微处理器为核心,配以大规模集成电路存储器、输入输出接口电路及系统总线所组成的计算机。微型计算机的产生与发展是与组成微型计算机的核心部件——微处理器的产生与发展紧密相关的。每当一种新型的微处理器出现时,就会带动微型计算机中其他部件的相应发展。例如,微型计算机体系结构的进一步优化,存储器存储容量不断增大,存取速度不断提高,外围设备性能不断改进及新的设备的出现等,都是与微处理器的发展相适应的。

本书是根据“计算机应用技术”专业“微型计算机原理与接口技术”课程教学大纲编写的。全书共9章。第1~6章介绍微型计算机基础、微处理器、指令与指令系统、汇编语言程序设计、总线、存储器等有关微型计算机原理的基础知识;第7~9章介绍微型计算机数据传送方法,串、并行通信及接口,常用的控制接口,磁盘、光盘接口,人机交互接口等有关微型计算机接口的知识及实用技术。各章有学习要点提示、例题解析和练习题及答案。附录A为本书的实验指导书,附录B为模拟试题及答案。全书简明易懂,突出概念和实用技术。

本书可作为高等职业技术教育、高等教育自学考试、中等专科教育“微型计算机原理与接口技术”课程的教材,亦可供工程技术人员参考。

本书由李兰友、万振凯、李静东、叶华、杨晓光、尹慧、马秀琴等编写,在编写过程中得到边奠英教授的大量指导和帮助,北方交通大学出版社卢先河总编给予了许多指导,在此深致谢忱。由于编者水平和经验有限,不当或错误之处敬希读者指正。

编 者
2003年10月

目 录

第 1 章 微型计算机基础	1
1.1 计算机中数的表示和运算	1
1.1.1 计算机中的数制及转换	1
1.1.2 数的表示	5
1.1.3 数的运算	7
1.2 基本逻辑电路	8
1.2.1 基本逻辑门电路	8
1.2.2 译码器、触发器和寄存器	10
1.3 微型计算机	11
1.3.1 电子数字计算机概述	11
1.3.2 微型计算机组成与配置	13
1.4 例题解析	16
练习题	17
第 2 章 微处理器	22
2.1 微处理器的原理与组成	22
2.1.1 微处理器的基本结构	22
2.1.2 微处理器的工作原理	23
2.1.3 微处理器的特点与分类	23
2.2 8086/8088 微处理器	24
2.2.1 8086/8088 主要特征	24
2.2.2 8086 CPU 内部结构	24
2.3 80286/80386/80486 微处理器	27
2.3.1 80286 微处理器	27
2.3.2 80386 微处理器	27
2.3.3 80486 微处理器	28
2.4 Pentium 微处理器	28
2.4.1 Pentium 特征	29
2.4.2 Pentium 的内部结构	29
2.4.3 Pentium 的工作模式	31
2.4.4 Pentium 处理器的发展	31
2.5 例题解析	32
练习题	33
第 3 章 80X86 指令系统	37
3.1 指令和指令系统	37
3.1.1 概述	37
3.1.2 指令格式	38

3.1.3	指令周期	40
3.1.4	指令的流水线和并行控制	40
3.2	寻址方式	42
3.3	80X86 指令系统	45
3.3.1	数据传送指令	45
3.3.2	算术运算指令	48
3.3.3	逻辑运算和移位指令	52
3.3.4	串操作指令	55
3.3.5	程序控制指令	56
3.3.6	处理器控制指令	58
3.3.7	80X86 扩展指令	59
3.4	例题解析	60
	练习题	61
第 4 章	汇编语言程序设计	66
4.1	汇编语言的基本概念	66
4.1.1	机器语言和汇编语言	66
4.1.2	基本语法	66
4.2	汇编语言语句	67
4.2.1	语句的种类和格式	67
4.2.2	伪指令	70
4.2.3	宏指令语句	75
4.3	程序设计的基本方法	76
4.3.1	程序设计的基本步骤	76
4.3.2	程序设计的基本技术	78
4.3.3	子程序	82
4.4	DOS 调用和 BIOS 调用	84
4.4.1	概述	84
4.4.2	DOS 系统功能调用	85
4.4.3	BIOS 中断调用	87
4.5	例题解析	87
	练习题	88
第 5 章	总线技术	93
5.1	总线	93
5.1.1	概述	93
5.1.2	PC 机总线结构	94
5.2	IBM PC 机的总线结构与功能	95
5.2.1	IBM PC 机的总线结构	95
5.2.2	系统总线工作原理	98
5.2.3	ISA 总线及其扩充	100
5.3	PCI 总线	102

5.3.1	PCI 总线系统的结构	102
5.3.2	PCI 总线的特点	102
5.3.3	PCI 性能	103
5.4	外部总线	103
5.5	例题解析	106
	练习题	107
第 6 章	存储器	109
6.1	微型计算机存储器基础	109
6.2	半导体存储器	110
6.2.1	随机存储器 RAM	110
6.2.2	只读存储器 ROM	113
6.3	Cache 存储器	114
6.4	虚拟存储器	115
6.5	8086 系统存储器组织	116
6.5.1	主存储器工作原理	116
6.5.2	8086 CPU 与静态 RAM 的连接	117
6.5.3	8086 CPU 与 DRAM 的连接	119
6.6	例题解析	120
	练习题	121
第 7 章	输入输出与接口技术	125
7.1	接口的基本概念	125
7.1.1	接口功能及结构	125
7.1.2	I/O 接口硬件分类与发展	127
7.2	接口编址方式与地址编码方法	127
7.2.1	接口编址方式	127
7.2.2	I/O 接口地址编码方法	128
7.3	CPU 与外设之间的数据传送方式	130
7.3.1	程序查询方式	130
7.3.2	中断方式	131
7.3.3	DMA 方式	133
7.4	串行、并行接口技术	133
7.4.1	串行接口	133
7.4.2	并行通信接口	136
7.5	模拟接口技术	137
7.5.1	D/A 接口技术	137
7.5.2	A/D 接口技术	140
7.6	例题解析	143
	练习题	144
第 8 章	接口芯片	147
8.1	可编程串行通信接口 8251A	147

8.1.1	8251A 引脚和内部结构	147
8.1.2	8251A 应用实例	149
8.2	可编程并行接口 8255A	150
8.2.1	8255A 引脚与结构	150
8.2.2	8255A 应用实例	153
8.3	可编程计数器/定时器 8253A	154
8.3.1	8253A 引脚与内部结构	155
8.3.2	8253A 工作方式	157
8.4	DMA 控制器 8237A	158
8.4.1	8237A 的引脚与内部结构	159
8.4.2	8237A 的应用实例	162
8.5	中断控制器 8259A	163
8.5.1	8259A 引脚与内部结构	163
8.5.2	8259A 应用实例	165
8.6	例题解析	165
	练习题	167
第 9 章	外围设备及接口	170
9.1	磁盘、光盘接口	170
9.1.1	软盘存储器	170
9.1.2	硬盘存储器	172
9.1.3	光盘存储器	175
9.1.4	闪存和 USB 接口	177
9.2	LED 显示器接口技术	178
9.2.1	LED 显示基本原理	178
9.2.2	LED 显示器与显示方式	179
9.2.3	LED 应用实例	180
9.3	键盘接口技术	182
9.4	CRT 显示器接口技术	185
9.4.1	显示器概述	186
9.4.2	CRT 显示器接口	186
9.5	打印机接口技术	188
9.5.1	打印机的基本工作原理	188
9.5.2	主机与打印机的接口	189
9.5.3	打印机适配器	191
9.5.4	打印机输出字符的 BIOS 调用方法	194
9.6	例题解析	194
	练习题	195
附录 A	实验指导书	197
附录 B	模拟试题	208

第 1 章 微型计算机基础

本章要点：

- ☑ 计算机中数的表示和运算
- ☑ 基本逻辑门电路的逻辑表达式及真值表
- ☑ 基本逻辑部件工作原理及简单应用
- ☑ 微处理器、微型机、微型计算机系统的区别与联系
- ☑ 微型计算机系统的组成及系统配置
- ☑ CPU、ROM、RAM、I/O 等名词术语的意义及功能

1.1 计算机中数的表示和运算

1.1.1 计算机中的数制及转换

计算机的基本功能是对数进行计算和各种处理，而数在计算机中是以电子器件的物理状态表示的。为了使表达方便可靠，均采用二进制数字系统。二进制数的“0”、“1”对应于电子器件的“0”、“1”两种物理状态，不便于我们书写和记忆。因此在计算机中对数的表示形式除采用二进制外，还广泛采用十进制、八进制或十六进制。

在计算机中除了数的运算和处理外，还大量地传送和显示各种字符、汉字等信息。用一串数码表示不同的事物，则这一串数码称为代码。代码中每位数码“0”和“1”已失去了其数量大小的含义。为便于查找和记忆，在编制代码时总是要遵循一定的规则，这些规则叫做码制（code system）。而二进制计数系统中信息的最小部分称为位（bit，意为 binary 字头）。所以二进制的 1 位包含的信息是 1 比特。在计算机中作为整体传送和参加运算的一组二进制代码，称做字（word）。表明一个字的二进制代码的位数叫做计算机的字长（word length）。计算机位数愈多，它能表达的数值就愈大，能表达的数值的有效位数也就愈多，计算的精度也愈高。在计算机中定义 8 位二进制作为一个信息单位，叫做一个字节（byte）；16 位二进制叫做两字节。

为了用数字量表示物理量的大小，仅仅使用 1 位数码来表达往往是不够用的，因此必须使用进位计数的方法组成多位数码，这种把多位数码中每 1 位的构成方法和从低位向高位进位的规则称为进位计数制，简称数制。在微型计算机中，常见和常使用的数制有十进制、二进制、八进制及十六进制等。为了区别所使用的数制，在几种数制混用的场合，常用下标或数制代号标注。例如 235_{10} 或 $(235)_D$ 表示十进制数 235。如采用代码标注，二进制用“B”表示（Binary 字头）；八进制用“Q”表示（Octal 字头，为避免将“O”与零相混，将“O”改写“Q”）；十六进制用“H”表示（Hexadecimal 字头）。考虑到十进制数极为常用，习惯上可以将下标“D”省略。

1. 十进制

十进制是我们日常生活中最常用的数制。特征如下：

- ◆ 使用 10 个不同的数码符号 0, 1, 2, 3, 4, 5, 6, 7, 8, 9。
- ◆ 基数为 10。
- ◆ 每一个数码符号根据它在数中所处的位置 (即数位), 按逢十进一决定其实际数值。

任意一个十进制正数 D , 可以写成如下形式:

$$(D)_{10} = D_{n-1} \times 10^{n-1} + D_{n-2} \times 10^{n-2} + \dots + D_1 \times 10^1 + D_0 \times 10^0 + D_{-1} \times 10^{-1} + D_{-2} \times 10^{-2} + \dots + D_{-m} \times 10^{-m}$$

例如, $(123.45)_{10}$ 以小数点为界, 往左是个位、十位、百位, 往右是十分位、百分位。所以可以表示为:

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

在计算机中, 一般用十进制作数据的输入、输出。

2. 二进制

在二进制计数制中, 基数是 2, 计数的原则是“逢二进一”。特征如下:

- ◆ 使用两个不同的数码符号 0 和 1。
- ◆ 基数为 2。
- ◆ 每一个数码符号根据它在数中所处的位置 (即数位), 按逢二进一决定其实际数值。

任意一个二进制正数 B , 可以写成如下形式:

$$(B)_2 = B_{n-1} \times 2^{n-1} + B_{n-2} \times 2^{n-2} + \dots + B_1 \times 2^1 + B_0 \times 2^0 + B_{-1} \times 2^{-1} + B_{-2} \times 2^{-2} + \dots + B_{-m} \times 2^{-m}$$

例如: $(11011.101)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = (27.625)_{10}$ 。

把十进制整数转换成二进制整数通常采用的方法是“除以 2 取余数”。即将十进制数除以 2, 得到一个商数和余数; 再将其商数除以 2, 又得到一个商数和余数; 按此方法继续下去, 直到商数等于零为止。每次所得的余数 (0 或 1) 就是对应二进制数的各位数字, 并且第一次得到的余数是二进制数的最低位, 最后一次得到的余数是二进制数的最高位。

[例 1.1] 将十进制数 123 转换为等值的二进制数。

根据上述原则, 将十进制数 123 转换成二进制数的过程如下。

2 123	余数	
2 61	1	二进制数的最低位
2 30	1	
2 15	0	
2 7	1	
2 3	1	
2 1	1	
0	1	二进制数的最高位

因此, 十进制数 123 转换成二进制数是 1111011。

把十进制小数转换成二进制小数所采用的规则是“乘以 2 取整数”。方法是用 2 乘以十进制纯小数, 将其结果的整数部分去掉; 再用 2 乘以余下的纯小数部分, 再去掉其结果的整数部分; 如此继续下去, 直到余下的纯小数为 0 或达到所要求的精度为止。最后将每次得到的整数部分 (0 或 1) 按先后顺序从左到右排列, 即得到所对应的二进制小数。

[例 1.2] 将十进制小数 0.6875 转换成二进制小数。

根据上述原则，将十进制小数 0.6875 转换成二进制小数的过程如下。

0.6875	整数	
× 2		
1.3750	1	二进制数的最高位
0.3750		
× 2		
0.7500	0	
0.7500		
× 2		
1.5000	1	
0.5000		
× 2		
1.0000	1	二进制数的最低位
1.0000		

因此，十进制小数 0.6875 转换成二进制小数为 0.1011。

注意：有些十进制小数不一定能完全准确地转换成二进制小数。在这种情况下，可以根据精度要求只转换到小数点后某一位为止。

在计算机中，数的存储、运算、传输都使用二进制。

3. 八进制

在八进制计数制中，基数是 8，计数的原则是“逢八进一”。特征如下：

- ◆ 使用 8 个不同的数码符号 0, 1, 2, 3, 4, 5, 6, 7。
- ◆ 基数为 8。
- ◆ 每一个数码符号根据它在数中所处的位置（即数位），按逢八进一来决定其实际数值。

任意一个八进制正数 S，可表示为：

$$(S)_8 = S_{n-1} \times 8^{n-1} + S_{n-2} \times 8^{n-2} + \dots + S_1 \times 8^1 + S_0 \times 8^0 + S_{-1} \times 8^{-1} + \dots + S_{-m} \times 8^{-m}$$

例如：

$$(472.64)_8 = 4 \times 8^2 + 7 \times 8^1 + 2 \times 8^0 + 6 \times 8^{-1} + 4 \times 8^{-2} = (314.8125)_{10}$$

将十进制整数转换成八进制整数的方法是“除以 8 取余数”。

[例 1.3] 将十进制数 59 转换成八进制数。

根据上述原则，将十进制数 59 转换成八进制数的过程如下。

8 59	余数	
8 7	3 最低位
0	7 最高位

因此，十进制数 59 转换成八进制数是 73。

将十进制数转换成八进制数的方法是：分别按“除以 8 取余数”和“乘以 8 取整数”对十进制数的整数部分与小数部分进行转换，然后再组合起来，即可得到相应的八进制数。

[例 1.4] 将十进制小数 0.6875 转换成八进制小数。

根据上述原则，将十进制小数 0.6875 转换成八进制小数的过程如下。

0.6875	×	8	
5.5000			5
0.5000			……去掉整数部分后
×	8		
4.0000			4
0.0000			……去掉整数部分后余下的纯小数为 0, 转换结束

所以, 十进制小数 0.6875 转换成八进制小数为 0.54。

将八进制数转换成二进制数的方法是: 把八进制数中的每一位数都用相应的三位二进制数来代替。

[例 1.5] 将八进制数 253.74 转换成二进制数。

根据上述原则, 将八进制数 253.74 转换成二进制数的过程如下。

2	5	3	·	7	4
↓	↓	↓		↓	↓
10	101	011	·	111	100

因此, 八进制数 253.74 转换成二进制数是 10101011.111100。该二进制数的整数部分中最左边的 0 和小数部分中最右边的 0 均可省略, 因此其二进制数是 10101011.1111。

4. 十六进制

在十六进制计数制中, 基数是 16, 计数的原则是“逢十六进一”。特征如下:

- ◆ 使用 16 个不同的数码符号, 它们是 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F。
- ◆ 基数为 16。
- ◆ 每一个数码符号根据它在数中的位置 (即数位), 按逢十六进一决定其实际数值。

对任意一个十六进制正数 H, 可表示为:

$$(H)_{16} = H_{n-1} \times 16^{n-1} + H_{n-2} \times 16^{n-2} + \cdots + H_1 \times 16^1 + H_0 \times 16^0 + H_{-1} \times 16^{-1} + \cdots + H_{-m} \times 16^{-m}$$

例如:

$$(3AB.28)_{16} = 3 \times 16^2 + A \times 16^1 + B \times 16^0 + 2 \times 16^{-1} + 8 \times 16^{-2} = (939.15625)_{10}$$

将十进制整数转换成十六进制整数的规则是“除 16 取余”。十六进制数计数的原则是“逢十六进一”。在十六进制数中, 用 A 表示 10; B 表示 11; C 表示 12; D 表示 13; E 表示 14; F 表示 15。

[例 1.6] 将十进制数 89 转换成十六进制数。

根据上述原则, 将十进制数 89 转换成十六进制数的过程如下。

16 89	余数	
16 5	9	……最低位
0	5	……最高位

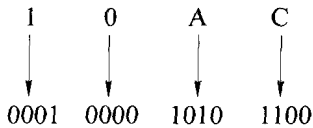
因此, 十进制数 89 转换成十六进制数是 59。

将十六进制数转换成二进制数的方法是: 把十六进制数中的每一位数都用相应的四位二

进制数来代替。

[例 1.7] 将十六进制数 10AC 转换成二进制数。

根据上述原则，将十六进制数 10AC 转换成二进制数的过程如下。



经转换后得到的二进制数是 0001000010101100。该二进制数中最左边的 3 个 0 可以省略，因此十六进制数 10AC 转换成二进制数是 1000010101100。

1.1.2 数的表示

1. 机器数的特点

数在计算机中的表示形式称为机器数，机器数所代表的实际数值称为真值。机器数有以下 3 个特点。

- ◆ 机器数表示的数值范围受计算机字长的限制。机器数的字长是由计算机的字长决定的，也就决定了机器数表示的范围。
- ◆ 机器数的符号位被数值化。因为机器数有正、负之分，统一规定机器数的正号用 0 表示，负号用 1 表示。在表示机器数的计算机字长中要拿出一位作为符号位。例如，在字长为 8 位的微机中，用最高位表示符号位，此时，机器数的表示范围为： $\{ (11111111)_2 = (-127)_{10} \} \sim \{ (01111111)_2 = (+127)_{10} \}$ 。
- ◆ 机器数的小数点处于约定的位置。计算机采用约定的方式表示小数点。小数点的位置可以有两种约定：一种是规定小数点的位置固定不变，这样表示的机器数称为定点数；另一种允许小数点可以浮动，这样表示的机器数称为浮点数。微型机中多使用定点数。

2. 带符号数的表示

在计算机中进行某些数的运算时，常常需要将数的正负表达出来，这些带有正负的数称为带符号数。前面我们所研究的二进制数，均未涉及数的符号问题，所以是一种无符号数。在微型计算机中，通常规定一个数的最高位为符号位，那么 $D_7 \sim D_0$ 则为数字位。对符号位规定 0 表示正，1 表示负。如：

$$X_1 = 01101101B = +109 \quad X_2 = 11101101B = -109$$

这种将符号位一起当做一个数，叫做机器数，此数的本身值叫做这个机器数的真值。所以， $X_1 = +1101101B$ 和 $X_2 = -1101101B$ 称为真值， $X_1 = 01101101B$ 和 $X_2 = 11101101B$ 就是机器数。

计算机中为了运算的方便，对带符号数有 3 种表示方法：原码、反码、补码。

(1) 原码

一个二进制数，最高位表示数的符号，其他位表达数值，这样的二进制数表示法叫原码表示法。符号位规定用 0 表示正数，用 1 表示负数。特征如下：

- ◆ 原码形式与二进制数的原来表示方法基本一样。
- ◆ 原码中增加了专门表示数的正、负的符号位，也就是用 0 表示正号，用 1 表示负号。

- ◆ 最左边一位的 0 和 1 不代表具体数值，而分别表示“+”和“-”。

例如， $X = 108$ 则 $[X]_{\text{原}} = 01101100\text{B}$

$X = -108$ 则 $[X]_{\text{原}} = 11101100\text{B}$

在计算机中，原码表示数简单明了，但是如果两个符号不同的数进行运算，处理起来非常不方便，因此还需要引进反码和补码。

(2) 反码

用反码表示带符号数规定：正数的反码与原码相同，负数的反码则为除符号位外的其余各位按位取反构成。其特点可以总结为：

- ◆ 正数的原码和反码完全一样。
- ◆ 负数的反码是由其原码的数值部分求反（即由 0 变为 1，1 变为 0）而得到的。

例如：

$[+8]_{\text{原}} = 00001000\text{B}$ $[+8]_{\text{反}} = 00001000\text{B}$

$[-8]_{\text{原}} = 10001000\text{B}$ $[-8]_{\text{反}} = 11110111\text{B}$

负数的反码还可由绝对值相同的正数，连同符号位在内全部变反求得。

(3) 补码

补码是计算机中带符号数的实用表示方法。规定如下：

- ◆ 正数的补码与原码和反码是一样的。
- ◆ 负数的补码可由其反码的末位加 1。即负数的补码是对其原码除符号位外各数值位求反并在末位加 1 而得到的。

[例 1.8] 求十进制数 (-123) 的补码。

首先将十进制数转换为二进制数原码。 $(-123)_{10} = (11111011)_{\text{原}}$ ，最左边是符号位。由原码求补码的原则是：正数的原码与补码相同；负数则除符号位外，其余各位求反，再在末位加 1。依此原则，可求得 $(-123)_{10}$ 的补码为 $(10000101)_{\text{补}}$ 。

3. 数中小数点的表示

(1) 数的定点表示

$$N = 2^P \times S$$

式中 S 称为数 N 的尾数， P 称为数 N 的阶码， 2 称为阶码的底。这里 P 和 S 都是用二进制表示的数。尾数 S 表示数 N 的全部有效数字，阶码 P 指明了小数点的位置。如果阶码 P 的值固定，这种表示方法为定点表示法，这种数称为定点数。

假如 $P=0$ ， S 又是纯整数，则小数点的位置定在 S 之后：

符号位	尾数
-----	----

·小数点

如果 $P=0$ ， S 是纯小数，则小数点的位置定在 S 之前：

符号位	尾数
-----	----

·小数点

小数点的位置只是约定的位置，实际上并不表示出来。应该指出，现代计算机都是能处理多种数值类型的计算机。

(2) 数的浮点表示

数的浮点表示就是一个数的小数点的位置不是固定的，而是可以浮动的。换句话说，如果阶码 P 是可变的，称这种表示方法为数的浮点表示法。用浮点表示法表示的数称为浮点数。

此时，在 $N=2^P \times S$ 式中，阶码 P 为可变的整数，且可正、可负，尾数 S 也可正、可负。

通常用一位二进制数 P_1 表示阶码的符号位。 $P_1=0$ 表示阶码为正， $P_1=1$ 表示阶码为负。同样用一位二进制数 S_1 表示尾数的符号位。 $S_1=0$ 表示尾数为正， $S_1=1$ 表示尾数为负。

浮点表示法首先要规定字的格式。例如，字长为 32 位，可用左边的 8 位作阶码，其中最左边一位为阶码的符号位；用右边的 24 位作为尾数，其中左边第一位为尾数的符号位。

格式如下：

P_1	阶码 P	S_1	尾数 S
-------	--------	-------	--------

大多数计算机中，都把尾数定为纯小数，即把小数点定在尾数 S 之前。这并不是把小数点位置固定了，实际的小数点位置是由阶码决定的。

4. 负数的表示

负数的表示常用“2 的补码形式”，即在最左边的数字前加负号，表示形式如下。

1	1	1	1	1
$-2^4 = -16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

上面表示的数字是： $1 \times (-2^4) + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = -16 + 8 + 4 + 2 + 1 = -1$

1.1.3 数的运算

1. 定点数运算

数值运算的核心是加、减、乘、除四则运算。计算机中的数又有定点和浮点表示形式，因此运算方法和规则又有所不同。减法运算可以通过补码转换成加法运算，因此计算机中可以只设置加法器。当采用补码运算时，为了在运算过程中保持结果的正确符号，可采用双符号位，双符号位补码又称为变形补码。

定点补码加减法的运算规则如下。

- ◆ 操作数均为补码表示。
- ◆ 符号位一起参加运算。
- ◆ 加法：做 $[X]_{\text{补}} + [Y]_{\text{补}}$ 。
- ◆ 减法：做 $[X]_{\text{补}} + [Y]_{\text{补}}$ 。
- ◆ 运算过程中，符号位向前的进位为模，舍弃。
- ◆ 运算结果仍为补码。

[例 1.9] 用变形补码计算 $[X]_{\text{补}} + [Y]_{\text{补}}$ ，并指出是否溢出，若溢出说明是正溢还是负溢。

$$\begin{array}{ll}
 X = +010110 & Y = +100101 \\
 [X]_{\text{补}} + [Y]_{\text{补}} = 00010110 + 00100101 = 00111011 & \text{未溢出} \\
 X = -001101 & Y = -010011
 \end{array}$$