



普通高等教育电气信息类规划教材



免费电子教案下载

www.cmpedu.com

微机原理 及接口技术

蔡启仲 蓝

等编著



机械工业出版社
CHINA MACHINE PRESS



普通高等教育电气信息类规划教材

微机原理及接口技术

蔡启仲 蓝红莉 庄俊华 代宣军 等编著



机械工业出版社

本书系统地介绍了微型计算机的基本组成、工作原理、接口技术,注重软硬件分析,主要内容包括:微型计算机基础、80X86 及 ARM 微处理器 8086/8088 指令系统、汇编语言程序设计、存储器、输入/输出接口技术、中断系统、D/A 和 A/D 转换器、微机中的 DMA 系统、总线技术以及附录。

本书注重基本知识的介绍,结构清晰,重点突出,在内容安排上,体现循序渐进、由浅入深的特点,便于课堂讲授和自学,为学习后续课程“单片机原理及应用”、“嵌入式系统原理及应用”等打下坚实的基础。

本书可作为高等学校电气信息类专业本科生的教材,也可作为各类成人教育的教材,还可供有关技术人员学习与参考。

本书配套授课电子课件,需要的教师可登录 www.cmpedu.com 免费注册、审核通过后下载,或联系编辑索取(QQ: 241151483, 电话: 010-88379753)。

图书在版编目(CIP)数据

微机原理及接口技术 / 蔡启仲等编著. —北京:机械工业出版社, 2013.5
普通高等教育电气信息类规划教材

ISBN 978-7-111-42233-4

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

中国版本图书馆 CIP 数据核字 (2013) 第 080424 号

机械工业出版社 (北京市百万庄大街 22 号 邮政编码 100037)

责任编辑: 时 静 韩 静

责任印制: 乔 宁

北京宝昌彩色印刷有限公司印刷

2013 年 5 月·第 1 版第 1 次印刷

184mm×260mm·18.25 印张·452 千字

0001-3500册

标准书号: ISBN 978-7-111-42233-4

定价: 39.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

电话服务

网络服务

社服务中心: (010) 88361066

教材网: <http://www.cmpedu.com>

销售一部: (010) 68326294

机工官网: <http://www.cmpbook.com>

销售二部: (010) 88379649

机工官博: <http://weibo.com/cmp1952>

读者购书热线: (010) 88379203

封面无防伪标均为盗版

前 言

20 世纪以来, 计算机应用已广泛深入到国民经济的各个领域, 计算机应用技术已成为电气信息类专业大学生必须掌握的基本理论知识与基本技能。“微机原理及接口技术”是一门必修的计算机基础课程, 它为学习后续课程“单片机原理及应用”、“嵌入式系统原理及应用”等打下坚实的理论基础。

本书以 Intel 8086 微处理器为基础, 系统地介绍 16 位微型计算机的基本结构、工作原理, 目的不在于掌握 Intel 8086 微处理器及其组成的微型计算机, 而在于掌握微型计算机的基本组成结构、汇编语言的编程技术、微型计算机的接口技术。根据这个编写思路, 本书充分考虑内容的选取与组织, 并兼顾后续课程学习的需求, 较好地体现了电气信息类专业的特点和要求。本书的主要内容包括三个部分: 微型计算机的基础与硬件部分, 微型计算机的软件部分, 微型计算机的接口技术与接口芯片部分。具体内容包括: 微型计算机基础、80X86 及 ARM 微处理器、总线技术; 8086/8088 指令系统、汇编语言程序设计; 存储器、输入/输出接口技术、中断系统、微机中的 DMA 系统、D/A 和 A/D 转换器以及附录。

本书的编写是采用集体讨论、分工撰写、交叉修改的方式进行的, 北京建筑工程学院陈志新负责组织全书的内容结构的策划, 广西科技大学蔡启仲担任主编, 广西科技大学蓝红莉和北京建筑工程学院庄俊华担任副主编。蓝红莉负责撰写第 3 章、第 4 章, 协助主编统稿工作; 庄俊华负责撰写第 1 章、第 6 章以及第 8 章; 桂林理工大学代宣军负责撰写第 2 章和第 10 章; 广西科技大学李克俭负责撰写第 5 章; 广西科技大学胡波负责撰写第 7 章; 河池学院宋华宁负责撰写第 9 章; 蔡启仲负责统稿工作和附录的编制。

在本书撰写过程中, 得到了广西科技大学的大力协助, 同时, 机械工业出版社对本书的编写和出版自始至终给予了非常宝贵的指导和支持, 谨在此一并致谢。本书编写也参考了若干出版物, 在此向有关作者表示感谢。

编 者

目 录

前言

第 1 章 微型计算机基础	1
1.1 计算机中的数制和编码	1
1.1.1 计算机中的数制	1
1.1.2 无符号数的表示及运算	1
1.1.3 带符号数的表示	5
1.1.4 补码的运算	7
1.1.5 计算机中的定点数和浮点数	10
1.1.6 计算机中的常用编码	12
1.2 微型计算机的分类与组成	14
1.2.1 微型计算机系统的分类	14
1.2.2 微型计算机系统的组成	15
1.3 微型计算机的基本工作原理	17
习题	20
第 2 章 80X86 及 ARM 微处理器	21
2.1 Intel 系列微处理器概述	21
2.2 8086/8088 微处理器	21
2.2.1 8086 CPU 引脚及其功能	22
2.2.2 8086 CPU 的基本结构	24
2.2.3 8086 CPU 的存储器组织	27
2.2.4 8086 CPU 的 I/O 组织	30
2.2.5 8086 CPU 最大模式和最小模式下的基本配置	31
2.2.6 8086 CPU 的内部时序	35
2.2.7 8086 与 8088 CPU 的主要区别	40
2.3 ARM 微处理器	41
2.3.1 S3C44B0X CPU 的引脚及其功能	41
2.3.2 S3C44B0X CPU 的基本结构	46
2.3.3 S3C44B0X CPU 的工作方式	49
2.3.4 S3C44B0X CPU 的内部寄存器	50
习题	52
第 3 章 8086/8088 指令系统	54
3.1 8086/8088 CPU 的寻址方式	54
3.1.1 寻址方式概述	54
3.1.2 与数据有关的寻址方式	54

3.2	8086/8088 的指令系统	59
3.2.1	数据传送指令	59
3.2.2	算术运算指令	64
3.2.3	逻辑指令	69
3.2.4	串操作指令	72
3.2.5	程序控制指令	76
3.2.6	处理器控制指令	81
	习题	82
第 4 章	汇编语言程序设计	84
4.1	汇编语言程序设计概述	84
4.1.1	汇编语句的种类和格式	84
4.1.2	汇编语言程序设计一般步骤	84
4.1.3	汇编语言程序的基本结构	85
4.2	汇编语言的表达式	85
4.2.1	数值表达式	85
4.2.2	地址表达式	87
4.3	伪指令	89
4.3.1	数据定义伪指令	89
4.3.2	符号定义伪指令	90
4.3.3	其他功能伪指令	90
4.3.4	汇编语言程序结构	92
4.4	宏指令	96
4.4.1	宏定义	97
4.4.2	宏调用	97
4.4.3	宏扩展	97
4.4.4	宏定义与宏调用中的参数	98
4.5	系统功能调用	98
4.5.1	概述	98
4.5.2	常用的输入/输出系统功能调用	99
4.6	汇编语言程序设计	102
4.6.1	顺序结构程序设计	102
4.6.2	分支结构程序设计	104
4.6.3	循环结构程序设计	107
4.6.4	子程序设计	111
4.7	实用程序设计举例	117
4.8	汇编语言程序的开发过程	121
4.8.1	上机开发过程	121
4.8.2	常用开发软件介绍	122
	习题	127

第 5 章 存储器	130
5.1 存储器的分类	130
5.1.1 按存储器在计算机系统中的作用分类	130
5.1.2 按存取方式分类	130
5.1.3 按存储介质分类	131
5.2 存储器的主要指标	132
5.3 读/写存储器 RAM	133
5.3.1 RAM 基本存储电路	133
5.3.2 RAM 的结构及译码方式	134
5.3.3 典型 RAM 芯片	137
5.4 只读存储器 ROM	141
5.4.1 掩膜式的 ROM	141
5.4.2 可编程的 ROM	142
5.4.3 可擦除可编程的 ROM	143
5.4.4 电可擦除可编程的 ROM	146
5.4.5 闪速存储器 Flash Memory	148
5.5 高速缓冲存储器 Cache	150
5.5.1 Cache 的工作原理	151
5.5.2 Cache 的组织结构	151
5.6 存储器芯片的扩展方法	152
5.7 微处理器 8086/8088 与存储器的连接	154
5.7.1 CPU 与存储器的连接要求	154
5.7.2 CPU 与静态随机存取存储器的连接实例	157
习题	158
第 6 章 输入/输出接口技术	160
6.1 计算机接口概述	160
6.1.1 接口的功能及组成	160
6.1.2 I/O 端口的编址方式	162
6.2 微机与外设之间的数据传送方式	164
6.2.1 无条件传送方式	164
6.2.2 查询传送方式	165
6.2.3 中断传送方式	165
6.2.4 直接存储器存取 (DMA) 方式	166
6.2.5 I/O 处理器控制方式	166
6.3 可编程定时/计数器接口芯片 8253	166
6.3.1 定时/计数概述	166
6.3.2 8253 的内部结构和外部引脚	167
6.3.3 8253 的控制字	169
6.3.4 8253 的初始化编程	170

6.3.5	8253 的工作方式	171
6.3.6	8253 应用举例	175
6.4	可编程并行接口芯片 8255A	178
6.4.1	8255A 的功能	178
6.4.2	8255A 的内部结构及外部引脚	178
6.4.3	8255A 的控制字与状态字	179
6.4.4	8255A 的初始化编程	180
6.4.5	8255A 的工作方式	181
6.4.6	8255A 应用举例	184
6.5	串行通信	186
6.5.1	串行通信基础	186
6.5.2	串行通信工作方式	190
6.5.3	RS-232C 标准	190
6.5.4	长距离串行通信	194
6.6	可编程串行接口芯片 8251A	195
6.6.1	8251A 的主要功能	195
6.6.2	8251A 的内部结构及外部引脚	195
6.6.3	8251A 的工作方式及控制字	198
6.6.4	8251A 应用举例	201
	习题	202
第 7 章	中断系统	206
7.1	中断系统的基本概念	206
7.1.1	中断概念	206
7.1.2	中断处理的过程	206
7.2	8086 CPU 的中断系统	208
7.2.1	8086 CPU 的中断源	208
7.2.2	8086 CPU 中断处理过程	210
7.2.3	中断向量与中断类型号	212
7.3	中断控制器 8259A	212
7.3.1	8259A 的外部引脚和内部结构	212
7.3.2	8259A 的中断工作方式	215
7.3.3	8259A 的编程	218
7.3.4	8259A 应用举例	222
	习题	226
第 8 章	微机中的 DMA 系统	227
8.1	DMA 系统概述	227
8.1.1	DMA 方式的基本工作过程	227
8.1.2	DMA 方式的操作步骤	228
8.2	可编程 DMA 控制器 8237	229

8.2.1	8237 的结构、引脚及功能	229
8.2.2	8237 的内部寄存器	235
8.2.3	8237 内部的接口地址分配	238
8.2.4	8237 的应用	239
习题		240
第 9 章 数/模 (D/A) 和模/数 (A/D) 转换器		241
9.1	D/A 和 A/D 转换器在微机控制系统中的作用	241
9.2	D/A 转换器接口	242
9.2.1	D/A 转换的基本原理	242
9.2.2	D/A 转换器的主要性能指标	243
9.2.3	D/A 转换器 DAC0832	244
9.2.4	D/A 转换器 DAC1210	248
9.3	A/D 转换器	251
9.3.1	A/D 转换的基本原理	251
9.3.2	A/D 转换器的主要性能指标	253
9.3.3	A/D 转换器 ADC0809	253
9.3.4	A/D 转换器 AD574A	257
习题		262
第 10 章 总线技术		263
10.1	总线技术概述	263
10.1.1	总线基本概念及标准	263
10.1.2	面向总线结构的优点	265
10.1.3	总线控制原理	265
10.2	系统总线	266
10.2.1	ISA 总线	267
10.2.2	EISA 总线	269
10.2.3	VESA 总线	269
10.2.4	PCI 总线	270
10.3	外部通信总线	272
10.3.1	EPP 并行接口	273
10.3.2	USB 外设总线	273
习题		276
附录		277
附录 A 8086/8088 指令系统表		277
附录 B ASCII 码字符表		283
参考文献		284

第1章 微型计算机基础

本章介绍计算机中的数制与编码、二进制数运算方法、十进制 BCD 码的基本概念、微型计算机的分类与组成及基本工作原理，这是学习本门课程后续章节内容的必要准备。

1.1 计算机中的数制和编码

1.1.1 计算机中的数制

数制是计算机重要的基础知识之一，计算机最基本的功能是进行数据的加工和处理，无论其表现形式是文本、字符、图形，还是声音、图像，都必须以数的形式加以储存。要以数的形式储存就涉及数制的问题，对于机器来说，记数越简单，相应的电路就越简单，实现起来越容易，所以，计算机都采用二进制来进行记数。但是，由于二进制数在书写过程中过于烦琐且易出错，而且在汇编语言和一些高级语言中，需使用十六进制、八进制和十进制数，所以，在这里对数制及它们之间的转换进行介绍。

1.1.2 无符号数的表示及运算

1. 无符号数的表示法

(1) 十进制数的表示法

十进制计数法的特点是：

- ① 以 10 为底，逢 10 进位。
- ② 需要 10 个数字符号 0, 1, 2, ..., 9。

一个十进制数 N_D 可以表示为如下形式：

$$N_D = \sum_{i=-m}^{n-1} D_i \times 10^i \quad (1-1)$$

式中， m 表示小数位的位数， n 表示整数位的位数， D_i 为十进制数字符号 0~9。

例如： $246.8D = 2 \times 10^2 + 4 \times 10^1 + 6 \times 10^0 + 8 \times 10^{-1}$

上式中的后缀 D 表示十进制数 (Decimal)，但 D 可以省略。

(2) 二进制数的表示法

二进制计数法的特点是：

- ① 以 2 为底，逢 2 进位。
- ② 需要 2 个数字符号 0, 1。

一个二进制数可以表示为如下形式：

$$N_B = \sum_{i=-m}^{n-1} B_i \times 2^i \quad (1-2)$$

例如： $1010.1B = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1}$

上式中后缀 B 表示二进制数(Binary)。

(3) 十六进制数的表示法

十六进制计数法的特点是：

① 以 16 为底，逢 16 进位。

② 需要 16 个数字符号 0, 1, 2, ..., 9, A, B, C, D, E, F。其中 A~F 依次表示 10~15。

一个十六进制数可表示为如下形式：

$$N_H = \sum_{i=-m}^{n-1} H_i \times 16^i \quad (1-3)$$

例如：C4BF.2BH = $12 \times 16^3 + 4 \times 16^2 + 11 \times 16^1 + 15 \times 16^0 + 2 \times 16^{-1} + 11 \times 16^{-2}$ 。

上式中后缀 H 表示十六进制数(Hexadecimal)。

2. 数制转换

(1) 任意进制数转换为十进制数

二进制、十六进制、乃至任意进制的数转换为十进制数，按式 (1-2)、式 (1-3) 等展开求和即可。

(2) 十进制数转换为二进制数

① 十进制整数转换为二进制整数。任何一个十进制数转换为二进制数后，都可以表示成为式 (1-2) 的形式。问题的核心在于求出 n 及 B_i 。

下面通过一个简单的例子分析一下转换的方法。例如：

$$13D = 1101B = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$\begin{matrix} \uparrow & \uparrow & \uparrow & \uparrow \\ B_3 & B_2 & B_1 & B_0 \end{matrix}$

上式也可以表示为

$$\begin{aligned} 13D = 1101B &= (1 \times 2^2 + 1 \times 2) \times 2 + 0 \times 2^1 + 1 \times 2^0 \\ &= [(1 \times 2 + 1) \times 2 + 0] \times 2 + 1 \end{aligned}$$

$\begin{matrix} \uparrow & \uparrow & \uparrow & \uparrow \\ B_3 & B_2 & B_1 & B_0 \end{matrix}$

可见，要确定 13D 对应的二进制数，只需从右到左分别确定 B_0 、 B_1 、 B_2 和 B_3 即可。显然，从上式可以归纳出以下转换方法，用 2 连续去除十进制数，直至商等于零为止。逆序排列余数便得到与该十进制相应的二进制数各位的数值。过程如下：

$$\begin{array}{rcl} 2 \overline{)13} & & \\ 2 \overline{)6} & \cdots \cdots 1 & (\text{商6余1}) \rightarrow B_0 \\ 2 \overline{)3} & \cdots \cdots 0 & (\text{商3余0}) \rightarrow B_1 \\ 2 \overline{)1} & \cdots \cdots 1 & (\text{商1余1}) \rightarrow B_2 \\ 0 & \cdots \cdots 1 & (\text{商0余1}) \rightarrow B_3 \end{array}$$

所以 $13D = 1101B$ 。

用与此类似的方法可以完成十进制数至十六进制数的转换，不同的是用 16 连续去除而已。

② 十进制小数转换为二进制小数。根据式 (1-2)，则有：

$$0.8125D = B_{-1} \times 2^{-1} + B_{-2} \times 2^{-2} + B_{-3} \times 2^{-3} + B_{-4} \times 2^{-4}$$

$$= 2^{-1} \times \{B_{-1} + 2^{-1} \times [B_{-2} + 2^{-1} \times (B_{-3} + 2^{-1} \times B_{-4})]\}$$

由上式可以看出，十进制小数转换为二进制小数的方法是，连续用 2 去乘十进制小数，直至乘积的小数部分等于“0”。顺序排列每次乘积的整数部分，便得到二进制小数各位的系数 B_{-1} , B_{-2} , B_{-3} , ...。若乘积的小数部分永不为“0”，则根据精度的要求截取一定的位数即可。0.8125D 的转换过程如下：

$0.8125D \times 2 = 1.625$	得出 $B_{-1}=1$
$0.625D \times 2 = 1.25$	得出 $B_{-2}=1$
$0.25D \times 2 = 0.5$	得出 $B_{-3}=0$
$0.50D \times 2 = 1.0$	得出 $B_{-4}=1$

所以， $0.8125D = 0.1101B$ 。

(3) 二进制数与十六进制数之间的转换

因为 $2^4=16$ ，故二进制数转换为十六进制数只需以小数点为起点，向两端每 4 位二进制数用 1 位十六进制数表示即可。例如：

$$1101110.01011B = \underline{01101110.01011000}B = 6E.58H$$

二进制数书写冗长易错，因此一般用十六进制表示一个数，这样比较简洁、方便。

3. 二进制数的运算

(1) 二进制数的算术运算

一个数字系统只要能进行加法和减法运算，就可以利用加法和减法进行乘法、除法及其他数值运算。

① 加法。运算规则为 $0+0=0$, $0+1=1+0=1$, $1+1=10$ (逢 2 进 1)。例如： $1110+1010$ ，从最低位开始将被加数与加数逐位相加，则有：

$$\begin{array}{r} 1110 \\ + 1010 \\ \hline 11000 \end{array}$$

② 减法。运算法则为 $0-0=1-1=0$, $1-0=1$, $0-1=10$ (0 减 1 不够减，从高位借 1 作为 2，相当于 $2-1=1$)。例如： $1110-1001$ ，从最低位开始将被减数与减数逐位相减，则有：

$$\begin{array}{r} 1110 \\ - 1001 \\ \hline 0101 \end{array}$$

③ 乘法。运算规则为 $0 \times 0=0$, $0 \times 1=1 \times 0=0$, $1 \times 1=1$ 。

例如： 1011×1101 ，用乘数各位分别乘被乘数后求和，则有：

$$\begin{array}{r} 1011 \\ \times 1101 \\ \hline 1011 \\ 0000 \\ 1011 \\ 1011 \\ \hline 10001111 \end{array}$$

④ 除法。运算法则为 $0 \div 1 = 0$, $1 \div 1 = 1$ 。例如: $11100111 \div 1011$, 其中 11100111 为被除数, 1011 为除数。则有:

$$\begin{array}{r}
 10101 \\
 1011 \overline{) 11100111} \\
 \underline{- 1011} \\
 00110 \\
 \underline{- 0000} \\
 1101 \\
 \underline{- 1011} \\
 00101 \\
 \underline{- 0000} \\
 01011 \\
 \underline{- 1011} \\
 0
 \end{array}$$

可见, 商为 10101

(2) 逻辑运算

二进制数的逻辑运算包括“与”运算、“或”运算、“非”运算和“异或”运算。

① “与”运算。“与”运算通常用符号“ $\wedge$ ”(或“ $\&$ ”)表示。运算规则为 $0 \wedge 0 = 0$, $1 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 1 = 1$ 。例如: $10110101 \wedge 10001001$, 按位相与, 则有:

$$\begin{array}{r}
 10110101 \\
 \wedge \quad 10001001 \\
 \hline
 10000001
 \end{array}$$

可见, 能用“与”运算保留一些位, 而屏蔽掉另一些位。例如, 将 10101010 的低 4 位保留, 而高 4 位屏蔽掉, 则只需将 10101010 和 00001111 相“与”即可。

② “或”运算。“或”运算通常用符号“ $\vee$ ”(或“ $|$ ”)表示。运算规则为 $0 \vee 0 = 0$, $0 \vee 1 = 1$, $1 \vee 1 = 1$ 。例如: $11110011 \vee 10011111$, 按位相或, 则有:

$$\begin{array}{r}
 11110011 \\
 \vee \quad 10011111 \\
 \hline
 11111111
 \end{array}$$

可见, 能用“或”运算使一些位不变, 而使另一些位置 1。例如: 将 01010101 的高 3 位不变, 而低 5 位置 1, 则只需将 01010101 和 00011111 相“或”即可。

③ “非”运算。“非”运算通常用符号“-”表示。运算规则为 $\bar{0} = 1$, $\bar{1} = 0$ 。

例如: 10110111, 按位取反, 结果是 01001000。

④ “异或”运算。“异或”运算通常用符号“ $\oplus$ ”表示。运算规则为 $0 \oplus 0 = 0$, $1 \oplus 0 = 1$, $0 \oplus 1 = 1$, $1 \oplus 1 = 0$ 。例如: $10100101 \oplus 11001100$, 按位异或, 则有:

$$\begin{array}{r}
 10100101 \\
 \oplus \quad 11001100 \\
 \hline
 01101001
 \end{array}$$

可见, 能用“异或”运算使某些数清零。例如: 将 10100110 清零, 只需将 10100110 和本身“异或”即可。

还可以用“异或”运算比较两个数是否相等。

例如： $11101110 \oplus 10111011 \neq 0$ ，所以 11101110 和 10111011 不相等。

1.1.3 带符号数的表示

计算机中使用的数据分为无符号数和有符号数两种。上面几节阐述的二进制数，是一种无符号数的表示，例如：8 位无符号数表示的范围为 00H~FFH(0~255)。但是在机器中，有符号的数据有正、负之分，那么符号是怎样表示的呢？通常一个数的最高位为符号位，符号位为“0”表示正数，符号位为“1”表示负数，其余为数值。若字长为 8 位，即 D_7 为符号位， $D_6 \sim D_0$ 为数值位。符号位 D_7 为“0”表示正数，为“1”表示负数。8 位有符号数 T_1 的表示如图 1-1 所示。

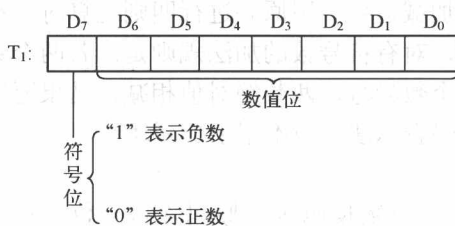


图 1-1 8 位有符号数的表示

若字长为 16 位， D_{15} 为符号位， $D_{14} \sim D_0$ 位为数值位，16 位有符号数 T_2 表示如图 1-2 所示。

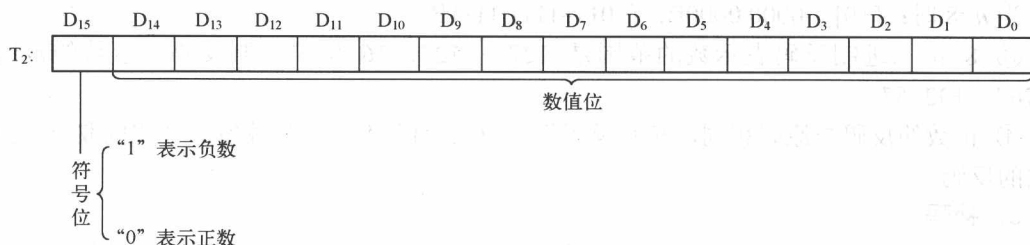


图 1-2 16 位有符号数的表示

字长是衡量计算机运算精度的指标，在计算机中完成一次运算和处理所对应的一组二进制数的位数称为计算机的字长。字长越长，计算处理的二进制有效位数越多，计算的精度越高。在机器中，用二进制数表示有符号数，用最高位表示符号，其余的为数值位，这样一组连同符号一起编码化的二进制数称为机器数，机器数所代表的数值的大小称为机器数的真值。

数据在计算机内采用符号化处理后，机器可表示并识别带符号的数据。为了改进运算方法，简化控制电路，人们研究出多种有符号数的编码方式，最常用的有三种表示方法，即原码、反码和补码。

数 X 的原码记作 $[X]_{\text{原}}$ ，反码记作 $[X]_{\text{反}}$ ，补码记作 $[X]_{\text{补}}$ 。

1. 原码

在机器数中最高位为符号位，符号位为“0”表示正数，符号位为“1”表示负数，其余

为该数的绝对值，这种表示方法就称为原码。即仅将符号位数字化表示为 0 或 1，数的绝对值与符号一起编码。

原码的特点：

① 最高位为符号位，正数为 0，负数为 1。

② 8 位二进制原码表示数的范围是 $-127 \sim +127$ ；16 位二进制原码表示数的范围是 $-32767 \sim +32767$ 。

③ 0 的原码有两种表示方法，即 +0 和 -0，设字长为 8 位：

$$[+0]_{\text{原}} = 0000\ 0000\text{B}; [-0]_{\text{原}} = 1000\ 0000\text{B}$$

采用原码表示法时，编码简单直观，与真值转换方便。但也带来一些麻烦，一是引起机器中零的表示不唯一，0 有二义性，给机器判零带来麻烦，必须在设计时约定好机器采用正零或负零；二是不便于进行加减运算。用原码进行四则运算时，符号位需单独处理，而且原码加减运算规则复杂。例如，对有符号数的加法规则是：若两个数同号，两数绝对值相加，结果冠以共同的符号；若两个数异号，两数绝对值相减，结果冠以绝对值大的数值的符号，而减法又有一套规则。故原码表示法一般不用于加减运算。

2. 反码

正数的反码与原码相同，负数反码等于原码除符号位保持为“1”外，其余各位按位取反。

反码的特点：

① 反码表示法中，最高位仍为符号位，正数为 0，负数为 1。

② “0” 有两种表示方法：

当 $n=8$ 时： $[+0]_{\text{反}} = 0000\ 0000\text{B}$ ； $[-0]_{\text{反}} = 1111\ 1111\text{B}$

③ 8 位二进制反码表示数的范围是 $-127 \sim +127$ ；16 位二进制反码表示数的范围是 $-32767 \sim +32767$ 。

④ 正数的反码与原码相同，负数反码符号位为“1”不变，其数值部分按位取反得到该数据的反码。

3. 补码

正数的补码表示与原码相同，即最高位为符号位，其余位为数值位；负数的补码等于它的反码加 1，即在其反码的最低位加 1 就为该数的补码。

补码的特点：

① 补码表示法中，最高位仍为符号位，正数为 0，负数为 1。

② “0” 仅有一种表示方法，即 $[+0]_{\text{补}} = [-0]_{\text{补}}$ 。

③ 8 位二进制补码表示数的范围是 $-128 \sim +127$ 。16 位二进制补码表示数的范围是 $-32768 \sim +32767$ 。

④ 对于 8 位二进制数 $1000\ 0000\text{B}$ ，在补码中定义为 $[-128]$ ，在原码中定义为 $[-0]$ ，在反码中定义为 $[-127]$ ；16 位二进制数 $1000\ 0000\ 0000\ 0000\text{B}$ ，在 16 位的补码中定义为 $[-32768]$ 。

在微型计算机中，所有带符号的数据都是用补码表示。一个数据是带符号数还是不带符号数，事先是规范好的，是已知的。

8 位二进制数的原码、反码和补码见表 1-1。

表 1-1 原码、反码和补码

二进制数	无符号数	带符号数		
		原码	反码	补码
0000 0000	0	+0	+0	+0
0000 0001	1	+1	+1	+1
0000 0010	2	+2	+2	+2
...	...	...	...	...
0111 1110	126	+126	+126	+126
0111 1111	127	+127	+127	+127
1000 0000	128	-0	-127	-128
1000 0001	129	-1	-126	-127
...	...	...	...	...
1111 1101	253	-125	-2	-3
1111 1110	254	-126	-1	-2
1111 1111	255	-127	-0	-1

4. 数的补码与真值

(1) 正数补码与真值的关系

如前面所述，正数的原码 $[X]_{\text{原}}$ 等于正数的补码 $[X]_{\text{补}}$ ，即 $[X]_{\text{补}}=[X]_{\text{原}}$ ，所以正数补码的真值 $X=[X]_{\text{补}}$ 。

(2) 负数补码与真值的关系

负数补码求一次该负数补码的反码，然后+1，即可得到补码对应真值的绝对值，其符号位也参与求反运算。

即 $|X|=[\overline{X}]_{\text{补}}+1$ ；其中 $[\overline{X}]_{\text{补}}$ 表示对补码进行求反运算。

例 1-1 $[X]_{\text{补}}=1000\ 0010\text{B}$ ，求 $[X]_{\text{真}}$ 值。

解：因为该二进制补码符号位为“1”，该补码对应的真值是负数，则

$$|X|=[\overline{X}]_{\text{补}}+1=\overline{1000\ 0010}+1=0111\ 1101+1=0111\ 1110=126$$

由于 $[X]_{\text{补}}$ 的符号位为 1，则 $[X]_{\text{真}}=-0111\ 1110=-126$

例 1-2 求 8 位补码 0111 1111B 和 1000 0001B 的真值。

解：(1) 因为 $[X]_{\text{补}}=0111\ 1111\text{B}$ ，符号位为“0”，该补码对应的真值是正数，

则 $[X]_{\text{真}}=[X]_{\text{补}}=0111\ 1111\text{B}=+127\text{D}$

(2) 因为 1000 0001B 的符号位为“1”，该补码对应的真值是负数，其绝对值为

$$|X|=[\overline{X}]_{\text{补}}+1=\overline{1000\ 0001}+1=0111\ 1110+1=0111\ 1111\text{B}=127\text{D}$$

则 $[X]_{\text{真}}=-0111\ 1111\text{B}=-127\text{D}$

1.1.4 补码的运算

在计算机中，带有符号位的数都是用补码表示。在算数运算过程中，连同符号位一起参加运算，称为补码运算，其运算结果仍为补码。

1. 补码加法

补码加法的规则是：和的补码等于补码之和。设 $X+Y=Z$ ，则 $[X]_{\text{补}}+[Y]_{\text{补}}=[X+Y]_{\text{补}}=[Z]_{\text{补}}$ ，其中， X 、 Y 为正、负数均可。

例 1-3 已知： $[+51]_{\text{补}}=0011\ 0011\text{B}$ ， $[+66]_{\text{补}}=0100\ 0010\text{B}$

$[-51]_{\text{补}}=1100\ 1101\text{B}$ ， $[-66]_{\text{补}}=1011\ 1110\text{B}$

求： $[+66]_{\text{补}}+[+51]_{\text{补}}=?$ $[+66]_{\text{补}}+[-51]_{\text{补}}=?$ $[-66]_{\text{补}}+[-51]_{\text{补}}=?$

解：(1) $[+66]_{\text{补}}+[+51]_{\text{补}}=?$

二进制(补码)加法		十进制加法
	0100 0010 $[+66]_{\text{补}}$	+ 66
+)	0011 0011 $[+51]_{\text{补}}$	+ 51
	0111 0101 $[+117]_{\text{补}}$	+ 117

由于 $[+66]_{\text{补}}+[+51]_{\text{补}}=[(+66)+(+51)]_{\text{补}}=01110101\text{B}$ 结果为正，因此， $[(+66)+(+51)]_{\text{原}}=[(+66)+(+51)]_{\text{补}}=01110101\text{B}$ 其真值为 +117，计算结果正确。

(2) $[+66]_{\text{补}}+[-51]_{\text{补}}=?$

二进制(补码)加法		十进制加法
	0100 0010 $[+66]_{\text{补}}$	+ 66
+)	1100 1101 $[-51]_{\text{补}}$	- 51
自动丢失 $\leftarrow 1$	0000 1111 $[+15]_{\text{补}}$	+ 15

由于 $[+66]_{\text{补}}+[-51]_{\text{补}}=[(+66)+(-51)]_{\text{补}}=00001111\text{B}$ 的符号位为 0，结果为正，因此， $[(+66)+(-51)]_{\text{原}}=[(+66)+(-51)]_{\text{补}}=00001111\text{B}$ ，其真值为 +15，计算结果正确。

(3) $[-66]_{\text{补}}+[-51]_{\text{补}}=?$

二进制(补码)加法		十进制加法
	1011 1110 $[-66]_{\text{补}}$	- 66
+)	1100 1101 $[-51]_{\text{补}}$	- 51
自动丢失 $\leftarrow 1$	1000 1011 $[-117]_{\text{补}}$	- 117

由于 $[-66]_{\text{补}}+[-51]_{\text{补}}=10001011\text{B}=[(-66)+(-51)]_{\text{补}}$ 结果为负，因此， $[(-66)+(-51)]_{\text{原}}=[(-66)+(-51)]_{\text{补}}=11110101\text{B}$ ，其真值为 -117，计算结果正确。

可以看出，不论被加数、加数是正数还是负数，只要直接用它们的补码直接相加，当结果不超出补码所表示的范围时，计算结果便是正确的补码形式。但当计算结果超出补码表示的范围时，结果就不正确了，这种情况称为溢出。

2. 补码减法

补码减法的规则是：差的补码等于第一个数的补码和第二个数符号变性之后(原正数变为负数，原负数变为正数)的补码相加。设 $X-Y=Z$ ，也可写成 $X+(-Y)=Z$ ，

则 $[X]_{\text{补}}+[-Y]_{\text{补}}=[X-Y]_{\text{补}}=[Z]_{\text{补}}$ 。

当 $Y \geq 0$ 时： Y 应变为负数，求其补码，再与 $[X]_{\text{补}}$ 相加；

当 $Y < 0$ 时： Y 应变为正数，求其补码，再与 $[X]_{\text{补}}$ 相加。

例 1-4 已知： $[+51]_{\text{补}}=0011\ 0011\text{B}$ ， $[+66]_{\text{补}}=0100\ 0010\text{B}$ ， $[-51]_{\text{补}}=1100\ 1101\text{B}$ ， $[-66]_{\text{补}}=1011\ 1110\text{B}$

求： $[+66]_{\text{补}}-[-51]_{\text{补}}=?$ $[-66]_{\text{补}}-[-51]_{\text{补}}=?$