

高职高专电气系列教材

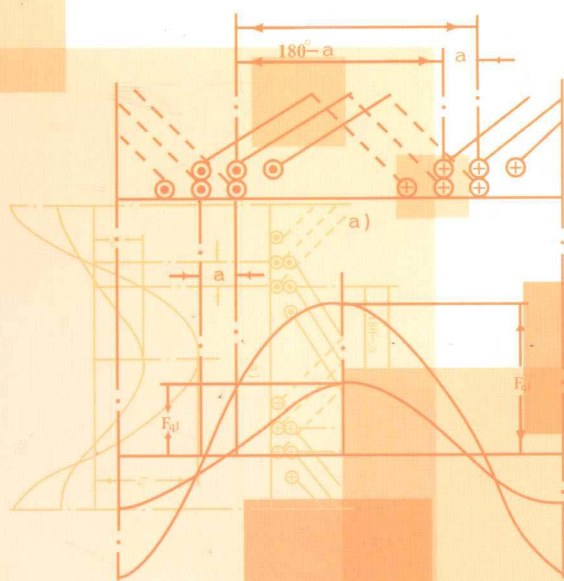
GAOZHI GAOZHUAN DIANQI XILIE JIAOCAI

单片机原理及其应用

DanPianJi YuanLi JiQi YingYong

■ 主 编 佟云峰

■ 副主编 胡文金 冯维杰



重庆大学出版社

单片机原理及其应用

主 编 佟云峰
副主编 胡文金 冯维杰

重 庆 大 学 出 版 社

内 容 简 介

本书以 MCS—51 系列单片机为主,较为详尽地介绍了单片机的组成、工作原理和应用技术。全书共分 7 章,内容包括:单片机的基本概念和发展、单片机的内部结构和外部端口功能、单片机指令系统及汇编程序设计、单片机的 C51 程序设计、系统扩展与接口技术、单片机的应用实例。本教材主要以高职高专院校电类专业学生为讲授对象,力求语言精练、准确,概念清晰,原理与应用紧密结合,例题与实例典型而实用,便于讲授或自学。本书也可作为各类工程技术人员学习单片机技术的参考书使用。

图书在版编目(CIP)数据

单片机原理及其应用/佟云峰主编. —重庆:重庆大学出版社,2004. 6

(高职高专电气类系列教材)

ISBN 7-5624-3070-5

I. 单... II. 佟... III. 单片微型计算机—高等学校:技术学校—教材 IV. TP368. 1

中国版本图书馆 CIP 数据核字(2004)第 012314 号

单片机原理及其应用

主 编 佟云峰

副主编 胡文金 冯维杰

责任编辑:谭 敏 版式设计:谭 敏

责任校对:何建云 责任印制:张立全

*

重庆大学出版社出版发行

出版人:张鸽盛

社址:重庆市沙坪坝正街 174 号重庆大学(A 区)内

邮编:400030

电话:(023) 65102378 65105781

传真:(023) 65103686 65105565

网址:<http://www.cqup.com.cn>

邮箱:fxk@cqup.com.cn (市场营销部)

全国新华书店经销

重庆华林天美彩色报刊印务有限公司印刷

*

开本:787×1092 1/16 印张:17.5 字数:437 千 插页:8 开 1 页

2004 年 6 月第 1 版 2004 年 6 月第 1 次印刷

印数:1—5 000

ISBN 7-5624-3070-5/TP·460 定价:24.00 元

本书如有印刷、装订等质量问题,本社负责调换

版权所有 翻印必究

前言

单片机(single chip microcontroller)以其集成度高、运算速度快、体积小、运行可靠、价格低廉等优势,在过程控制、数据采集、机电一体化、智能化仪器仪表、家用电器以及网络技术等方面得到广泛应用。特别是单片机嵌入式系统的开发与应用,标志着计算机发展史上又一个新的里程碑。作为计算机两大发展方向之一的单片机,以面向对象的实时控制为己任,嵌入到如家用电器、汽车、机器人、仪器仪表等设备中使其智能化。目前国内外各大电气公司,大的半导体厂商正不断地开发、使用单片机,使其无论在控制能力,减小体积,降低成本,还是开发环境的改善等方面,都得到了空前迅速的发展。各高校的许多专业纷纷开设单片机课程,单片机世界呈现出了一片繁荣景象。

目前以及今后相当长的一段时间内,在单片机应用领域中,8位单片机将仍然占据主流地位。因此,本教材以 MCS-51 单片机为主线讲述单片机的原理及应用技术,既符合教学特点的典型性,又不失内容的先进性。现阶段流行的各公司开发的品种繁杂的单片机,大多以 51 单片机为内核。因此,只要以一种机型学好单片机的基本原理和应用方法,就可达到“一通则百通”的效果。

本书由佟云峰主编,并编写了第3章、第7章的7.3.2和附录4;胡文金编写第4章的4.4、第6章、第7章的7.1、7.2、7.3.3和附录2;冯维杰编写第5章、第7章的7.3.1;吴政江编写第1章、附录1、附录3;秦培林编写第4章除4.4外其余部分;张帆编写第2章;昆明微控公司的邓鲁提供了有价值的资料,在此表示感谢。

限于水平,错误难免,敬请读者批评指正。

编 者

2004年1月21日

目 录

第1章 单片机概述	1
1.1 单片机的数学基础	1
1.1.1 数的进制及其相互转换	1
1.1.2 带符号数的表示方法	4
1.1.3 溢出的判别方法	6
1.1.4 ASCII 码和 BCD 码	8
1.2 单片机基础	10
1.2.1 计算机的经典组成	10
1.2.2 单片机的概念	11
1.2.3 单片机的应用范围	13
1.2.4 单片机的发展	14
1.2.5 单片机系统	15
1.2.6 单片机与嵌入式系统	18
1.3 常用单片机系列介绍	19
1.3.1 MCS—51 系列	19
1.3.2 MC68 系列	20
1.3.3 PIC16 系列	22
1.3.4 MSP430 系列	23
1.3.5 AVR 系列	25
本章小结	27
习题1	27
第2章 MCS—51 单片机的基本结构	29
2.1 单片机的内部结构	29
2.1.1 MCS—51 的组成框图	29
2.1.2 CPU 的结构	30
2.2 CPU 的时序和引脚功能	32

2.2.1	MCS—51 系列单片机的引脚功能描述	32
2.2.2	时钟电路及工作时序	33
2.2.3	复位电路及单片机的工作方式	35
2.3	存储器组织	37
2.3.1	存储器的划分	37
2.3.2	程序存储器	37
2.3.3	片内数据存储器 and 特殊功能寄存器(SFR)	38
2.3.4	片外数据存储器	45
2.4	MCS—51 的端口结构	45
2.4.1	端口功能	45
2.4.2	端口的内部结构与操作	46
2.4.3	读—修改—写操作	48
	本章小结	49
	习题 2	49
第 3 章	MCS—51 单片机的指令系统	50
3.1	指令格式和寻址方式	50
3.1.1	指令格式及符号约定	50
3.1.2	MCS—51 单片机的寻址方式	51
3.2	MCS—51 的指令系统	56
3.2.1	指令的分类	56
3.2.2	数据传送类指令	57
3.2.3	算术运算类指令	64
3.2.4	逻辑运算类指令	69
3.2.5	控制转移类指令	73
3.2.6	位操作类指令	79
3.3	汇编语言程序设计	83
3.3.1	伪指令	83
3.3.2	汇编语言程序的基本结构	84
3.3.3	汇编语言程序设计实例	89
	本章小结	92
	习题 3	92
第 4 章	MCS—51 的功能部件及应用	95
4.1	MCS—51 的中断系统及其应用	95
4.1.1	中断的概念	95
4.1.2	MCS—51 的中断源	99
4.1.3	MCS—51 对中断的控制	100

4.1.4 中断系统应用举例	104
4.2 定时器/计数器及其应用	105
4.2.1 定时器/计数器的结构	105
4.2.2 定时器/计数器的初始化	106
4.2.3 定时器/计数器的应用举例	109
4.3 串行口及应用	111
4.3.1 串行通信基础	111
4.3.2 MCS—51 串行口的结构	113
4.3.3 串行口的控制	113
4.3.4 串行通信实例	117
*4.4 其他集成功能部件的应用	119
4.4.1 MCS—51 的集成功能部件	119
4.4.2 I ² C 接口及其应用	120
4.4.3 WDT 及其应用	121
4.4.4 SPI 接口及其应用	122
4.4.5 集成 E ² PROM 的使用	123
4.4.6 集成 ADC 及其应用	125
4.4.7 集成 DAC 及其应用	128
本章小结	129
习题4	129
 第5章 单片机系统扩展及接口技术	130
5.1 存储器扩展技术	130
5.1.1 单片机系统的三总线	130
5.1.2 常用的存储器芯片	132
5.1.3 程序存储器的扩展	139
5.1.4 外部数据存储器的扩展	142
5.1.5 多片存储器芯片的扩展	144
5.2 并行 I/O 接口的扩展技术	146
5.2.1 可编程并行 I/O 接口芯片 8255A	146
5.2.2 8155 与单片机的接口	157
5.3 人机接口技术	163
5.3.1 LED 显示器与单片机的接口	163
5.3.2 液晶显示器(LCD)与单片机的接口	169
5.3.3 键盘接口	170
5.3.4 8279 芯片应用	176
5.4 模拟量输入/输出接口技术	186
5.4.1 D/A 芯片及其接口设计	187

5.4.2 A/D 芯片及其接口设计	192
5.5 开关电路及驱动电路接口	202
5.5.1 开关电路接口	202
5.5.2 光电耦合器驱动接口	204
本章小结	206
习题 5	208
第 6 章 单片机的 C51 程序设计	209
6.1 C51 数据类型及存储类型	209
6.1.1 C51 的数据类型	209
6.1.2 C51 的数据存储类型	210
6.1.3 C51 对单片机主要资源的定义	211
6.2 C51 的基本运算	213
6.2.1 C51 的算数运算	213
6.2.2 C51 的关系运算	214
6.2.3 C51 的逻辑运算	214
6.2.4 C51 的位运算	215
6.2.5 C51 的赋值运算	215
6.3 C51 的构造数据类型	216
6.3.1 数组	216
6.3.2 指针	217
6.3.3 结构	218
6.4 C51 流程控制语句	218
6.4.1 选择控制语句	219
6.4.2 循环语句	220
6.4.3 C51 的中断控制	222
6.5 C51 函数	224
6.5.1 函数的分类与定义	224
6.5.2 函数的调用	224
6.6 C51 应用编程实例	225
本章小结	231
习题 6	231
第 7 章 单片机应用系统设计与开发	232
7.1 单片机应用系统设计概述	232
7.1.1 设计要求与设计步骤	232
7.1.2 需求分析与总体方案设计	233
7.1.3 硬件设计和软件设计	234

7.2 单片机系统的可靠性设计	235
7.2.1 单片机系统的抗干扰设计	236
7.2.2 单片机系统的可靠性设计原则	236
7.3 单片机应用系统设计实例	237
7.3.1 交通信号灯控制系统	237
7.3.2 校园作息时间表控制系统	242
7.3.3 环境温度监测系统	249
本章小结	256
习题 7	256
附录	257
附录 1 MCS—51 系列单片机指令表	257
附录 2 C51 常用库函数	263
附录 3 ASCII(美国标准信息交换码)表	266
附录 4 常用集成芯片引脚图	267
参考文献	270

第 1 章

单片机概述

本章基本要求:

单片机是现代电子智能仪器仪表及嵌入式系统的主要组成部分,应用非常广泛,是现代工程技术人员必须掌握的知识之一。本章要求掌握数的进制及其相互转换、带符号数的表示方法、溢出的判别方法、ASCII 码和 BCD 码等单片机的数学基础知识;掌握单片机的概念、特点、应用范围、发展历程等基础知识;了解常用单片机系列,为后续章节的学习打下基础。

1.1 单片机的数学基础

1.1.1 数的进制及其相互转换

(1) 数的几种常用进制

数制是人们利用符号来计数的方法,数制有很多种,人们熟悉的是十进制。但由于数在机器中是以器件的物理状态来表示的,因此一个具有两种稳定状态且能相互转换的器件,就可以用来表示一位二进制数。二进制数的表示是最简单而且是最可靠的,另外,二进制的运算规则也是最简单的。因此,迄今为止,所有计算机都是以二进制进行算术运算和逻辑运算的。但是在使用二进制编写程序时既繁琐又容易出错,所以人们在编写程序时又经常用到十进制、十六进制或八进制。下面分别予以介绍。

任何一种数制都有两个要素,即基数和权。基数为数制中所使用的数码的个数。当基数为 R 时,该数制可使用的数码为 $0 \sim (R-1)$ 。例如在二进制中基数为 2,可使用 0 和 1 两个数码。在进行运算时按逢 R 进一,借 1 当 R 的规则进行。权是数制中某一数位上单位数的大小,它是一个指数,底是基数 R ,幂是数码的位置号,数码的位置号从 0 开始。将一个数中某一位的数码与该位的权相乘,即为该位数码的数值。

1) 十进制 (Decimal)

十进制是以 10 为基数,逢十进一、借一当十的计数体制。计数符号共有十个,分别为:0、1、2、3、4、5、6、7、8、9。计数规则是逢十进一,借一当十。十进制数常用下标 D 或 10 表示。

加权系数表示:

$$[M]_D = \sum_{i=-\infty}^{\infty} K_i \times 10^i$$

K_i 为第 i 位的系数可取 0 ~ 9 十个数字符号中的任一个; 10^i 为第 i 位的权。显然各位的权是 10 的幂。

例 1.1 $(42.51)_{10} = 4 \times 10^1 + 2 \times 10^0 + 5 \times 10^{-1} + 1 \times 10^{-2}$

2) 二进制(Binary)

二进制是以 2 为基数,逢二进一、借一当二的计数体制。计数符号共有两个,分别为:0、1。计数规则是逢二进一、借一当二。二进制数常用下标 B 或 2 表示。

运算规则: $0+0=0$ $0+1=1+0=1$ $1+1=10$ (读“壹零”) $0 \times 0=0$

$$1 \times 0 = 0 \times 1 = 0 \quad 1 \times 1 = 1$$

加权系数表示:

$$[M]_B = \sum_{i=-\infty}^{\infty} K_i \times 2^i$$

K_i 为第 i 位的系数可取 0 或 1; 2^i 为第 i 位的权,即各位的权是 2 的幂。

例 1.2 $[1101.01]_B = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$

下面再介绍有关二进制的几个概念:

位:一位二进制信息,只能是 0 或 1,也叫比特(bit)。

字节:计算机中将 8 位二进制数称为一个字节,也叫拜特(Byte)。为了表示大容量存储器的需要,人们还定义了千字节(KB)、兆字节(MB)、吉字节(GB)三个单位。它们的关系为:

$$1 \text{ KB} = 2^{10} \text{ Byte} = 1024 \text{ Byte}$$

$$1 \text{ MB} = 2^{10} \text{ KB} = 1024 \text{ KB} = 2^{20} \text{ Byte}$$

$$1 \text{ GB} = 2^{10} \text{ MB} = 1024 \text{ MB} = 2^{30} \text{ Byte}$$

字:计算机一次能处理的二进制数称为一个字,也叫沃德(Word),字是计算机中参加运算的基本单位。由于目前微型计算机通常是 16 位的,因此通常认为一个字为 16 位二进制数,即 $1 \text{ Word} = 2 \text{ Byte}$ 。

3) 八进制

八进制是以 8 为基数,逢八进一、借一当八的计数体制。计数符号共有 8 个,分别为:0、1、2、3、4、5、6、7。计数规则是逢八进一,借一当八。八进制数常用下标 O 或 8 表示。

加权系数表示:

$$[M]_O = \sum_{i=-\infty}^{\infty} K_i \times 8^i$$

K_i 为第 i 位的系数可取 0 ~ 7 八个数字符号中的任一个; 8^i 为第 i 位的权。显然各位的权是 8 的幂。

例 1.3 $[236]_O = 2 \times 8^2 + 3 \times 8^1 + 6 \times 8^0$

对八进制数有一个重要特点,那就是每位八进制数可用三位二进制数表示。例如:

$$(6)_8 = (110)_2。$$

4) 十六进制(Hexadecimal)

十六进制是以 16 为基数,逢十六进一、借一当十六的计数体制。计数符号共有 16 个,分别为:0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。计数规则是逢十六进一,借一当十六。十六进

制数常用下标 H 或 16 表示。

加权系数表示：

$$[M]_H = \sum_{i=-\infty}^{\infty} K_i \times 16^i$$

K_i 为第 i 位的系数可取 $0 \sim F$ 十六个数字符号中的任一个； 16^i 为第 i 位的权。显然各位的权是 16 的幂。

例 1.4 $[4E6]_H = 4 \times 16^2 + 14 \times 16^1 + 6 \times 16^0$

对十六进制数有一个重要特点,那就是每位十六进制数可用四位二进制数表示。例如：
 $(E)_{16} = (1110)_2$ 。

(2) 不同进制数之间的相互转换

1) 任意进制数转为十进制数

方法:按权展开求和。

例 1.5 $[1101.01]_B = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$
 $= 8 + 4 + 0 + 1 + 0 + 0.25 = (13.25)_D$

$[236]_8 = 2 \times 8^2 + 3 \times 8^1 + 6 \times 8^0 = 128 + 24 + 6 = (158)_{10}$

$[C2]_H = 12 \times 16^1 + 2 \times 16^0 = 192 + 2 = (194)_{10}$

2) 十进制数转为二进制数

方法:对整数部分,连续除 2 取余反排列,直到商为 0;对小数部分,连续乘 2 取整正排列,直到乘积的小数部分为 0 或满足误差要求。

例 1.6 $[338.38]_D = [101010010.01100]_B$ $[25.706]_D = [11001.10111]_B$ (保留 5 位小数)。

2 25 ...余1	整数最低位	0.706	
2 12 ...余0		× 2	
2 6 ...余0		1.412	... 取整1 小数最高位
2 3 ...余1		0.412	
2 1 ...余1	整数最高位	× 2	
0		0.824	... 取整0
		× 2	
		1.648	... 取整1
		0.648	
		× 2	
		1.296	... 取整1
		0.296	
		× 2	
		0.592	... 取整0 小数最低位

由于最后舍弃的数 0.592 大于 0.5,因此按“四舍五入”原则,小数最低位 0 加 1。

推广:十进制数转为任意进制数。整数部分,连续除基数取余反排列,直到商为 0;小数部分,连续乘基数取整正排列,直到乘积的小数部分为 0 或满足误差要求。

3) 八进制数与二进制数之间的相互转换

二进制转为八进制:对整数部分,从最低位开始三位三位地分组,不足三位的前面补零;对小数部分,则从最高位开始三位三位地分组,不足三位的后面补0。然后每组以其对应的八进制数代替,排列顺序不变。

八进制转为二进制:将每位八进制数写成对应的三位二进制数,再按原来的顺序排列起来即可。

例 1.7 $[11110100010]_2 = [3642]_8$ $[6403]_8 = [110100000011]_2$

4) 十六进制数与二进制数之间的相互转换

方法:跟八进制数与二进制数之间的相互转换相似,只是按四位分组即可。

例 1.8 $[11110100010]_2 = [7A2]_{16}$ $[B59]_{16} = [101101011001]_2$

5) 八进制数与十六进制数之间的相互转换

方法:通过二进制数作中间变量进行变换。

例 1.9 $[B59]_{16} = [101101011001]_2 = [5531]_8$

$[6403]_8 = [110100000011]_2 = [D03]_{16}$

1.1.2 带符号数的表示方法

(1) 机器数与真值

前面提到的二进制数,没有涉及符号问题,是一种无符号数。但在实际应用中,一个数显然还有正、负之分,那么符号在计算机中是怎么表示的呢?计算机中采用二进制数码,对于数的符号“+”或“-”也用二进制数码表示。规定用二进制数码的最高位表示符号。并规定:用数码“0”表示正数的符号“+”;用数码“1”表示负数的符号“-”。

例 1.10 $X_1 = +010001B$; $X_2 = -010001B$, 在 8 位机分别表示为 $X_1 = 00010001B$; $X_2 = 10010001B$ 。

一个数在机器中的表示形式称为机器数,而原来的实际数本身称为机器数的真值。在计算机中常用的机器数有原码、反码、补码三种形式。当真值为 X 时,其原码、反码、补码分别用 $[X]_{\text{原}}$ 、 $[X]_{\text{反}}$ 、 $[X]_{\text{补}}$ 表示。

(2) 原码(true form)

符号位用“0”表示正数,“1”表示负数,其余各位表示真值除符号外的尾数本身,这种表示方法称为原码表示法。即用 0、1 分别代替真值中的“+”、“-”即得原码。以八位机为例(下同)。

1) 对于正数: $[X]_{\text{原}} = X$

例 1.11 若 $X_1 = +1101001B$, $X_2 = +101101B$, 则 $[X_1]_{\text{原}} = 01101001B$, $[X_2]_{\text{原}} = 00101101B$ (不足 8 位应在符号位后补“0”)。

2) 对于负数: $[X]_{\text{原}} = 2^{8-1} - X$

例 1.12 若 $X_1 = -1101001B$, $X_2 = -101101B$, 则

$[X_1]_{\text{原}} = 11101001B = 10000000B + 1101001B = 2^{8-1} - (-1101001B) = 2^{8-1} - X_1$

$[X_2]_{\text{原}} = 10101101B = 10000000B + 101101B = 2^{8-1} - (-101101B) = 2^{8-1} - X_2$

3) 对于 0

在计算机中,0 可认为它是 +0,也可认为它是 -0,故 0 在原码中有两种表示法。对八位

机: $[+0]_{\text{原}} = 00000000\text{B}$, $[-0]_{\text{原}} = 10000000\text{B}$ 。

字长为 n 位的原码表示法的一般规律:

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

(3) 反码(one's complement)

1) 对于正数, 其反码表示法与原码相同, 即 $[X]_{\text{反}} = [X]_{\text{原}} = X$ 。

例 1.13 若 $X_1 = +1101001\text{B}$, $X_2 = +101101\text{B}$, 则 $[X_1]_{\text{反}} = [X_1]_{\text{原}} = 01101001\text{B}$, $[X_2]_{\text{反}} = [X_2]_{\text{原}} = 00101101\text{B}$ (不足 8 位应在符号位后补“0”)。

2) 对于负数, 反码等于其原码符号位不变, 其余各位按位取反 (即“1”换成“0”, “0”换成“1”)。也可按以下公式计算: $[X]_{\text{反}} = 2^8 - 1 + X$ 。

例 1.14 若 $X = -1101001\text{B}$, 则 $[X]_{\text{原}} = 11101001\text{B}$, $[X]_{\text{反}} = 10010110\text{B} = 2^8 - 1 + (-1101001\text{B}) = 2^8 - 1 - 1101001\text{B}$ 。

3) 对于 0, 反码有 $[+0]_{\text{反}}$ 和 $[-0]_{\text{反}}$ 两种表示法。对于 8 位机: $[+0]_{\text{反}} = 00000000\text{B}$, $[-0]_{\text{反}} = 11111111\text{B}$ 。

字长为 n 位的反码表示法的一般规律:

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

(4) 补码(two's complement)

补码表示法可以把负数转换为正数, 使减法转换为加法, 从而使正负数的加减运算转换为单纯的正数相加的运算。因此, 计算机中一般采用补码表示法。

1) 对于正数, 其补码就是该正数本身, 即 $[X]_{\text{补}} = X$ 。

例 1.15 若 $X = +1101001\text{B}$, 则 $[X]_{\text{补}} = 01101001\text{B}$ 。

2) 对于负数, 其补码等于其反码加 1。即 $[X]_{\text{补}} = [X]_{\text{反}} + 1 = 2^n - 1 + X + 1 = 2^n + X$ (对 8 位机 $n=8$)。

例 1.16 若 $X = -1101001\text{B}$, 则 $[X]_{\text{原}} = 11101001\text{B}$, $[X]_{\text{反}} = 10010110\text{B}$, $[X]_{\text{补}} = 10010110\text{B} + 1 = 10010111\text{B} = 2^8 + X = 2^8 + (-1101001\text{B}) = 2^8 - 1101001\text{B}$ 。

3) 对于 0, $[+0]_{\text{补}} = [-0]_{\text{补}} = 00000000\text{B}$, 即 0 的补码只有一种表示法。

字长为 n 位的补码表示法的一般规律:

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

综上所述, 对正数有 $[X]_{\text{原}} = [X]_{\text{反}} = [X]_{\text{补}} = X$; 对负数, 用“1”代替负号“-”就得原码, 再对原码除符号位(最高位)外其余各位按位取反即得反码, 最后对反码加 1 就得补码。

(5) 已知机器数求真值

1) 先求原码。对正数(符号位为 0), 原码、反码、补码相同, 无须转换; 对负数(符号位为 1), 反码的数值位按位取反, 可转换为原码, 补码的数值位按位取反后末位加 1, 可转换为原码。

2) 由原码求真值。用“+”、“-”代替原码的符号位 (“0”换为“+”, “1”换为“-”) 即可。

例 1.17 若 $[X]_{\text{补}} = 10011010\text{B}$, 求 X ?

解 因符号位为 1, 所以 X 为负数。则 $[X]_{\text{原}} = 11100101\text{B} + 1 = 11100110\text{B}$, $X = -1100110\text{B}$

1.1.3 溢出的判别方法

(1) 计算机中带符号数的加减法运算

在微型计算机中, 原码表示的数易于识别, 但做加减法运算时比较复杂, 符号位和数值位需要分别处理。首先做两个数绝对值的减法, 用绝对值大的数减去绝对值小的数, 然后用绝对值大的数的符号作为结果的符号。采用补码做加减法运算时, 符号位与数值位同时参与运算, 减法也转换为加法运算, 符号位无须单独处理。

1) 补码加法运算

补码加法运算的规则是: $[X + Y]_{\text{补}} = [X]_{\text{补}} + [Y]_{\text{补}}$

例 1.18 已知 $X = 1100011\text{B}$, $Y = -0011\text{B}$, 求 $X + Y = ?$

解 $[X]_{\text{补}} = 01100011\text{B}$, $[Y]_{\text{补}} = 11111001\text{B}$

$$\begin{array}{r} [X]_{\text{补}} = 01100011\text{B} \\ +) [Y]_{\text{补}} = 11111001\text{B} \\ \hline \end{array}$$

模溢出 $\rightarrow \boxed{1} 01011100\text{B}$

所以 $[X + Y]_{\text{补}} = [X]_{\text{补}} + [Y]_{\text{补}} = 01011100\text{B}$, $X + Y = +1011100\text{B}$, 最高位的进位自动丢失 (称为溢出)。

2) 补码减法运算

在微型计算机中减法运算也通过补码转换为加法运算, 减法运算的规则是:

$[X - Y]_{\text{补}} = [X + (-Y)]_{\text{补}} = [X]_{\text{补}} + [-Y]_{\text{补}}$, 其中 $[-Y]_{\text{补}}$ 可由 $-Y$ 求出, 也可以由 $[Y]_{\text{补}}$ 求出。把 $[Y]_{\text{补}}$ 的符号位与数值位一起取反, 末位加 1, 结果就等于 $[-Y]_{\text{补}}$ 。

例 1.19 已知 $X = 1000111\text{B}$, $Y = 1001\text{B}$, 求 $X - Y = ?$

解 $[X]_{\text{补}} = 01000111\text{B}$, $[Y]_{\text{补}} = 00001001\text{B}$, $[-Y]_{\text{补}} = 11110111\text{B}$

$$\begin{array}{r} [X]_{\text{补}} = 01000111\text{B} \\ +) [-Y]_{\text{补}} = 11110111\text{B} \\ \hline \end{array}$$

模溢出 $\rightarrow \boxed{1} 00111110\text{B}$

$[X - Y]_{\text{补}} = 00111110\text{B}$, $X - Y = +0111110\text{B}$

(2) 溢出的判别方法

1) 溢出的概念

在计算机内部表示数据与人工表示数据的情况不同, 人工表示数据时, 数据的值可以为任意大小, 而在计算机内只能用有限位数来表示数据。所以计算机中所能表示的数有一定的范围, 对于绝对值太大而超过一定值的数, 计算机无法表示, 这时会造成数据的最高位丢失, 数据产生错误, 这种情况称为上溢出。出现上溢出时, 应停止运算, 进行错误处理。对于绝对值太小的数, 在计算机中同样也表示不出来, 此时计算机将这个数作为 0 处理, 数据产生误差, 这种情况称为下溢出。由于下溢出所带来的误差很小, 在允许范围之内, 不做错误处理。因此在以

后提到的溢出指的是上溢出。

2) 溢出的判断

当两个数作加减法运算时,如何判断运算结果是否有溢出呢?常用的方法有补码和变形补码两种方法。

①补码判断法

两个用补码表示的数作加减法运算时,如果是同号相减或异号相加,只能使数据的绝对值越来越小,运算结果不可能产生溢出;如果是同号相加或异号相减,则运算结果可能会出现溢出。此时,可以把运算结果的符号与参与运算的数据符号相比较,如果出现正数加正数得负数或负数加负数得正数的情况,则可以断定运算结果出现了溢出。

例 1.20 已知 $X = 1110010B$, $Y = 1001101B$, 求 $X + Y = ?$

解 $[X]_{补} = 01110010B$, $[Y]_{补} = 01001101B$

$$\begin{array}{r} [X]_{补} = 01110010B \\ +) [Y]_{补} = 01001101B \\ \hline \end{array}$$

$$10111111B$$

由运算结果可以看出,两个正数相加,结果为负数,可以断定是溢出造成的。出现溢出时运算结果是错误的,不再使用。

例 1.21 已知 $X = -1100111B$, $Y = 1001100B$, 求 $X - Y = ?$

解 $[X]_{补} = 10011001B$, $[Y]_{补} = 01001100B$, $[-Y]_{补} = 10110100B$

$$\begin{array}{r} [X]_{补} = 10011001B \\ +) [-Y]_{补} = 10110100B \\ \hline \end{array}$$

$$\text{模溢出} \rightarrow \boxed{1} 01001101B$$

由运算结果可以看出,两个负数相加,结果为正数,可以断定是溢出造成的。出现溢出时运算结果是错误的,不再使用。

例 1.22 已知 $X = 1100111B$, $Y = 1110011B$, 求 $X - Y = ?$

解 $[X]_{补} = 01100111B$, $[Y]_{补} = 01110011B$, $[-Y]_{补} = 10001101B$

$$\begin{array}{r} [X]_{补} = 01100111B \\ +) [-Y]_{补} = 10001101B \\ \hline \end{array}$$

$$11110100B$$

由于是同号相减,运算结果不可能产生溢出。因此 $[X + Y]_{补} = 11110100B$, $X + Y = -0001100B$

②变形补码判断法

变形补码是采用双符号位表示的补码,用 00 表示正数,用 11 表示负数。用变形补码判断运算结果是否有溢出时,只需要判断结果的双符号位是否相同即可,如果双符号位相同,运算结果没有溢出,否则运算结果有溢出。

例 1.23 已知 $X = -1100111B$, $Y = 1001100B$, 求 $X + Y = ?$ 和 $X - Y = ?$

解 $[X]_{变形补} = 110011001B$, $[Y]_{变形补} = 001001100B$, $[-Y]_{变形补} = 110110100B$

$$\begin{array}{r} [X]_{\text{变形补}} = 110011001\text{B} \\ +) [Y]_{\text{变形补}} = 001001100\text{B} \\ \hline \end{array}$$

$$111100101\text{B}$$

双符号位相同,结果无溢出, $X + Y = -0011011\text{B}$ 。

$$\begin{array}{r} [X]_{\text{变形补}} = 110011001\text{B} \\ +) [-Y]_{\text{变形补}} = 110110100\text{B} \\ \hline \end{array}$$

$$101001101\text{B}$$

双符号位不同,结果溢出。出现溢出时运算结果是错误的,不再使用。

1.1.4 ASCII 码和 BCD 码

(1) 二进制代码

数码符号不仅可以用于计数表示数值的大小,而且可以用于表示特定的对象。如电话号码、邮政编码、手机号码等就是用 0~9 这十个十进制数码符号的组合来表示特定的对象,可以称为十进制代码。同样,由 0 和 1 组成的二进制数码不仅可以表示数值的大小,而且可以用来表示特定的信息。这种具有特定含义的二进制数码称为二进制代码。建立这种代码与它表示的对象(如十进制数、字母、特定符号、逻辑值等)的一一对应关系过程称为编码;将代码所表示的特定信息翻译出来称为译码,分别由编码器、译码器来实现。

(2) 二一十进制码(BCD 码)

二一十进制码就是用四位二进制数来表示 0~9 这十个十进制符号,简称为 BCD 码。由于四位二进制数从 0000~1111 共有 16 种组合,而十进制只有十个数码符号,因此有很多种 BCD 码。如 8421 码、2421 码、5211 码、余 3 码等等。常用的是 8421BCD 码。

1) 8421 码

8421 码是用四位二进制数的前十种组合来表示 0~9 这十个十进制数。这种代码每一位的权都是固定不变的,属于恒权代码。它和四位二进制数一样,从高位到低位各位的权分别是 8、4、2、1,故称为 8421 码。其特点是每个代码的各位数值之和就是它所表示的十进制数。所以,它便于记忆,应用也比较普遍。

例 1.24 若 $X = (01001010)_2$, $Y = (00110111)_2$, 求 $X + Y$ 的 BCD 码?

$$\begin{aligned} \text{解 } X + Y &= (01001010)_2 + (00110111)_2 = (10000001)_2 = (129)_{10} \\ &= (000100101001)_{\text{BCD}} \end{aligned}$$

2) 2421 码和 5211 码

它们也属于恒权代码,从高位到低位各位的权分别是 2、4、2、1 和 5、2、1、1,故而得名。其中,2421 码又分为(A)和(B)两种代码,它们的编码状态不完全相同。在 2421(B)码中,0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 互为反码,即两码对应位的值相反。

3) 余 3 码

这种代码所组成的四位二进制数,正好比它代表的十进制数多 3,故称为余 3 码。两个余 3 码相加时,其和要比对应表示的十进制数之和多 6。因而两个十进制数之和等于 10 时,两个对应余 3 码之和相当于四位二进制的 16,刚好产生进位信号,不必进行修正。另外,余 3 码的 0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 也互为反码。余 3 码不能由各位二进制数的权来决定其代