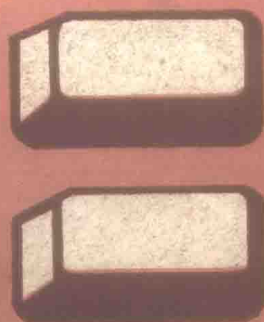
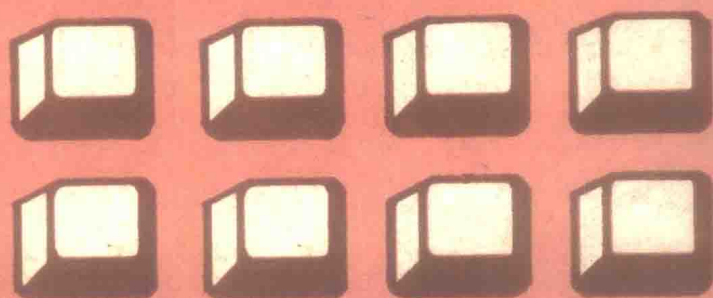


微型计算机技术

(高校工科非电专业适用)

黄义源 任仕纯 编



高等教育出版社

微 型 计 算 机 技 术

(高校工科非电类专业适用)

黄义源 任仕纯 编

高 等 教 育 出 版 社

这是一本国家教委高校工科电工学课程教学指导小组 1986~1990 年教材规划中计划编写的、以 Z80 汇编语言程序设计和接口技术为主要内容的简明教材,主要适用于高校工科机械类或其他各非电类专业,也可供各专业工程技术人员参考。

全书共分 9 章: 数字电路基础、数字逻辑部件、微机的硬件结构、CPU 的指令系统、汇编语言程序设计、接口技术、单板机及其应用、单片计算机、16 位微处理器 8086 等。

本书有明显的特色: 易学。讲授全书大约只需 50 学时, 其中微机部分约 32 学时。书中文字流畅, 重点突出, 难点分散, 思路清晰; 从实用角度出发, 删繁就简, 联系实际; 并将数字电路与微机相结合。本书数字电路内容符合国家教委批准的《电子技术(电工学 II)课程教学基本要求》。

本书责任编辑 王锦惠

微型计算机技术

(高校工科非电类专业适用)

黄义源 任仕纯 编

*

高等教育出版社出版

新华书店北京发行所发行

商务印书馆上海印刷厂印装

开本 787×1092 1/16 印张 19.5 字数 485 000

1988 年 10 月第 1 版 1988 年 10 月第 1 次印刷

印数 0001—10,620

ISBN 7-04-001094-1/TP·31

定价 3.90 元

前 言

随着微型计算机技术的飞速发展,微型计算机的应用已深入到各个领域,促进众多专业的发展。微型计算机现已成为工科各专业共同的基础知识。为了适应这种形势的需要,我们为本校非电专业学生编写了《微型计算机技术》教材,并从1984年开始试用,本书就是在这本教材多次试用的基础上修改而成的。

本书可作为高等学校工科非电专业的教材,也可作为广大工程技术人员学习和应用微型计算机的参考书。

考虑到非电专业的实际情况,全书力求突出两个特点:

1. 便于学习。在内容的处理上,一是将数字电路与微型计算机有机的结合起来。数字电路是微型计算机的硬件基础;微型计算机是数字电路的典型应用。将两者紧密结合起来,不仅可以减少重复,使得内容精简、结构紧凑,而且由数字电路过渡到微型计算机,思路清晰,循序渐进,对某些内容的学习容易达到事半功倍的效果。二是在讲述CPU、PIO、OTC等典型芯片时,从应用的观点出发,内部结构仅作最简要的说明,力戒“技术说明书”式地介绍,尽量分散难点,便于非电类专业的学生理解,便于非电类专业的技术人员自学。

2. 面向应用。非电专业学生学习微型计算机,主要是用它来研究和解决本专业的问题,推动本专业科学技术的发展。微型计算机的应用,主要是两个方面的问题,一是程序编制,二是接口技术。本书围绕这两方面组织微型计算机教学内容,并编入了大量的例题和习题。

本书数字电路部分是依据国家教委批准、1987年秋季始在全国试行的《高等工业学校电子技术(电工学II)课程教学基本要求》编写的,又按照微型计算机的需要,作了适当的扩充。

全书按70学时编写(含实验15~20学时),其中数字电路部分约占25学时,微机部分占45学时。前面六章为基础部分,是本书的重点。七、八、九章可根据学时多少和需要选学。

本书一、二、三章由任仕纯执笔,其余各章由黄义源编写。清华大学梁毓厚副教授审阅了书稿,提出了许多宝贵的意见。在本书编写过程中,湖南大学杨贻馨教授给予了大量的帮助和指导,在此一并表示深切的感谢。

由于我们的学识水平有限,书中难免有错误和不妥之处,殷切期望读者批评指正。

编 者

1986年12月于湖南大学机械电子学教研室

目 录

第一章 数字逻辑基础1

1-1 计算机中数的表示方法.....1

一、计数制1

二、2 进制2

三、16 进制4

四、有符号数的表示方法5

五、2-10 进制编码9

六、字符代码10

思考练习题11

1-2 逻辑代数和逻辑门12

一、基本逻辑关系及逻辑门12

二、逻辑代数的基本公式17

三、逻辑门的组合应用19

四、门电路的控制作用22

五、正逻辑、负逻辑和混合逻辑23

六、集成逻辑门24

思考练习题26

1-3 触发器27

一、RS 触发器27

二、JK 触发器30

三、D 触发器33

四、触发器应用举例34

思考练习题36

习题一37

第二章 数字逻辑部件40

2-1 寄存器40

一、数码寄存器40

二、移位寄存器41

思考练习题44

2-2 计数器44

一、2 进制计数器44

二、10 进制计数器49

三、应用举例——药片计数器51

思考练习题52

2-3 译码器和数据选择器52

一、2 进制译码器52

二、显示器和显示译码器54

三、数据选择器和数据分配器56

思考练习题58

2-4 运算器58

一、半加器58

二、全加器59

三、4 位 2 进制加法器60

思考练习题61

2-5 定时电路61

一、多谐振荡器61

二、555 集成定时器62

三、单稳态触发器65

思考练习题67

习题二67

第三章 微型计算机的硬件结构69

3-1 总线69

一、三态门69

二、总线71

思考练习题72

3-2 微处理器72

一、概述72

二、寄存器组73

三、运算器74

四、控制器76

思考练习题78

3-3 存储器79

一、概述79

二 随机存储器(RAM)79

三、只读存储器(ROM)84

思考练习题87

3-4 简单的微型机硬件系统87

一、CPU 和存储器的连接87

二、I/O 接口及其与 CPU 的连接88

三、操作90

思考练习题	90	思考练习题	138
习题三	90	5-4 堆栈和子程序	138
第四章 CPU 的指令系统	92	一、堆栈	138
4-1 指令格式和寻址方式	92	二、子程序的调用和返回	140
一、机器码和助记符	92	三、子程序应用举例	141
二、指令的格式	93	四、辅助寄存器和交换指令	146
三、寻址方式	94	思考练习题	147
思考练习题	96	5-5 提高程序的质量	147
4-2 数据传送指令	96	一、深入分析指令系统选择最合适的指令	147
一、8位传送指令	97	二、合理地使用寄存器	148
二、16位传送指令	99	三、选用快速的算法	149
三、数据块传送指令	100	四、精心设计循环结构	149
思考练习题	101	五、尽量采用子程序结构	150
4-3 数据操作指令	102	习题五	150
一、8位算术逻辑指令	102	第六章 接口技术	154
二、16位算术指令	107	6-1 输入输出(I/O)方式	154
三、通用算术指令	108	一、输入输出接口的编址方式	154
四、移位循环指令	108	二、Z80 CPU 的输入输出指令	154
五、位操作指令	110	三、输入输出方式	156
思考练习题	111	思考练习题	157
4-4 程序控制指令	112	6-2 中断	157
一、无条件转移指令	112	一、中断的控制及中断指令	158
二、条件转移指令	112	二、中断的优先权排队与多重中断	159
三、循环控制转移指令	114	三、中断的响应	161
思考练习题	115	思考练习题	164
习题四	115	6-3 并行 I/O 接口电路——PIC	164
第五章 汇编语言程序设计	117	一、PIO 的结构和功能	164
5-1 汇编语言概述	117	二、直接输入输出	167
一、机器语言、汇编语言和高级语言	117	三、中断控制方式的应用	170
二、汇编语言语句结构	118	四、联络控制工作方式	177
三、Z80 常用的伪指令	118	思考练习题	177
四、汇编方法	120	6-4 计数定时电路——CTC	178
思考练习题	122	一、CTC 的结构和引脚功能	178
5-2 简单程序的设计	122	二、CTC 用作计数器	180
一、概述	122	三、CTC 用作定时器	182
二、顺序结构程序	122	思考练习题	185
思考练习题	126	6-5 数模和模数转换	185
5-3 分支和循环	126	一、数模转换器(DAC)	185
一、分支程序	126	二、模数转换器(ADC)	189
二、循环程序	130		

思考练习题	193	二、数据操作	237
习题六	193	三、程序控制功能	240
第七章 单板机及其应用	196	8-4 输入、输出和中断	242
7-1 单板机的组成	196	一、I/O 接口	242
一、硬件	196	二、I/O 指令	243
二、单板微计算机的监控程序	200	三、中断	243
三、键盘和显示接口	202	四、定时器/计数器	244
7-2 顺序控制	206	8-5 程序举例	246
一、时间顺序控制	207	第九章 16 位微处理器 8086	250
二、过程顺序控制	211	9-1 概述	250
三、系统的简化	213	9-2 8086 CPU	250
7-3 数据采集	214	一、8086 的结构特点	250
7-4 步进电机的开环控制	217	二、8086 的寄存器结构	252
一、驱动电源	217	三、存贮器的寻址	253
二、脉冲分配	218	四、I/O 结构	254
三、举例	220	五、中断结构	254
7-5 直流伺服电机的脉宽调速	222	9-3 指令系统介绍	255
一、脉宽调速的基本原理	222	一、数据传送指令	255
二、开环的脉宽调速系统	223	二、算术运算指令	256
三、闭环脉宽调速系统的硬件结构	227	三、逻辑运算和移位循环指令	257
四、闭环系统的软件设计	229	四、字符串操作指令	258
第八章 单片计算机	232	五、程序控制指令	258
8-1 概述	232	六、处理器控制指令	259
8-2 单片机的结构	233	附录一 Z80 指令机器码表	261
一、CPU	233	附录二 Z80 指令系统	270
二、存贮器	235	附录三 MCS-48 系列单片机指令表	278
8-3 MCS-48 单片机的功能	237	部分习题答案	281
一、数据传送	237		

第一章 数字逻辑基础

本章主要介绍数字逻辑电路的一些基本知识,如各种数制之间的相互转换,码制,基本逻辑关系以及相应的数学基础——逻辑代数,基本的逻辑器件——门电路和触发器等,为学习微型计算机打下基础。

1-1 计算机中数的表示方法

计算机的主要功能是对数进行处理,数的表示方法是首先要解决的问题。

一、计数制

10进制是日常生活中最常用的计数制,它用0~9共10个数字符号来表示数的大小。这些数字符号也叫数码。数制所用数码的个数称为基数。例如10进制有10个数码,基数为10,计数时逢10进1。除10进制外,常用到的数制还有2进制、8进制、12进制、16进制等。基数小于10的计数制,用10进制的相应字符作为它的字符,如2进制的字符为0~1,8进制的字符为0~7。基数大于10的计数制,采用10进制字符加上大写的英文字母(从首字符开始),作为它的字符。如16进制的字符,除用0~9的十个数字外,还用A~F的六个字母。表1-1列出了这几种数制的基数和字符。

表1-1 计数制

数制	基数	字 符
2进制	2	0 1
8进制	8	0 1 2 3 4 5 6 7
10进制	10	0 1 2 3 4 5 6 7 8 9
12进制	12	0 1 2 3 4 5 6 7 8 9 A B
16进制	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

一个数常常是由多个数码组合成的。数码在数中的位置不同,其值也不同。例如10进制数123.4,其中数码1位于百位,其值为一百;2位于十位,其值为二十;3位于个位,其值为三;4在小数点后一位,即十分之一位,其值为十分之四。123.4这个数可以展开成下面的多项式:

$$123.4 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1}$$

式中, 10^2 、 10^1 、 10^0 、 10^{-1} 等称为该位的“权”,每一位上的数码与该位“权”的乘积,就是该位数值的大小。上式称为按权展开式。

其它进制的数,也可按权展开,其一般形式为:

$$N = d_{n-1}b^{n-1} + d_{n-2}b^{n-2} + \dots + d_{-m}b^{-m} = \sum_{i=n-1}^{-m} d_i b^i \quad (1-1)$$

式中: d_i ——第 i 位的数码;

b ——基数;

b^i ——权;

n ——整数的总位数;

m ——小数的总位数。

例如: 2 进制数 $11011 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

8 进制数 $256 = 2 \times 8^2 + 5 \times 8^1 + 6 \times 8^0$

16 进制数 $3AF4 = 3 \times 16^3 + A \times 16^2 + F \times 16^1 + 4 \times 16^0$

在计算机中, 主要采用 2 进制。因此, 下面将重点介绍 2 进制。

二、2 进制

1. 计算机中为什么采用 2 进制

(1) 2 进制只有 0, 1 两种状态, 很容易用电子元件来实现。例如可设开关的闭合为 1, 断开为 0; 电压的高为 1, 低为 0; 脉冲的有为 1, 无为 0 等等。而且这种简单的状态工作可靠, 抗干扰能力强。

(2) 2 进制数运算简单

2 进制数只有两个数码, 运算规则较 10 进制简单得多。

加法: 进位规则是逢 2 进 1, 即 $1+1=10$

$$\begin{array}{r} 1101 \cdots \cdots \text{被加数} \\ + 1001 \cdots \cdots \text{加数} \\ \hline 10110 \cdots \cdots \text{和} \end{array}$$

减法: 借位规则是“借 1 当 2”。

$$\begin{array}{r} 1.0.01 \cdots \cdots \text{被减数} \\ - 0.1.10 \cdots \cdots \text{减数} \\ \hline 0.0.11 \cdots \cdots \text{差} \end{array}$$

乘法: 规则是: $0 \times 0 = 0$, $1 \times 0 = 0$, $1 \times 1 = 1$

$$\begin{array}{r} 1011 \cdots \cdots \text{被乘数} \\ \times 101 \cdots \cdots \text{乘数} \\ \hline 1011 \\ 0000 \\ 1011 \\ \hline 110111 \cdots \cdots \text{积} \end{array}$$

除法: 除法是乘法的逆运算。

$$\begin{array}{r} 000111 \cdots \cdots \text{商} \\ \text{除数} \cdots 101 \overline{) 100011 \cdots \cdots \text{被除数}} \\ \underline{101} \\ 111 \\ \underline{101} \\ 101 \\ \underline{101} \\ 0 \end{array}$$

由上可见,在加、减、乘、除四则运算中,乘法实质上是做移位加法,除法则是做移位减法(本书第五章中将详细讨论)。后面将要讲到,减法也可改为做加法。因此,在计算机中只要用一种加法器,就可完成四则运算,从而使运算电路大为简化。

2. 2 进制到 10 进制的转换

如前所述,计算机中采用 2 进制,而人们习惯的是 10 进制,因此常需进行 2 进制和 10 进制的相互转换。

2 进制转换成 10 进制的基本方法,是对 2 进制的按权展开式求和,得到等值的 10 进制数。

[例 1-1] 将 2 进制数 1101_2 、 1011011_2 、 0.111_2 、 11.101_2 转换成等值的 10 进制数。

解: (1) $1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$$= 8 + 4 + 1$$

$$= 13_{10}$$

(2) $1011011_2 = 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^1 + 1$

$$= 64 + 16 + 8 + 2 + 1$$

$$= 91_{10}$$

(3) $0.111_2 = 2^{-1} + 2^{-2} + 2^{-3}$

$$= \frac{1}{2} + \frac{1}{4} + \frac{1}{8}$$

$$= 0.875_{10}$$

(4) $11.101_2 = 2^1 + 2^0 + 2^{-1} + 2^{-3}$

$$= 2 + 1 + 0.5 + 0.125$$

$$= 3.625_{10}$$

上述方法,对 N 进制都是适用的。

[例 1-2] 将 8 进制数 156 和 16 进制数 64 转换成等值的 10 进制数。

解: (1) $156_8 = 1 \times 8^2 + 5 \times 8^1 + 6 \times 8^0$

$$= 64 + 40 + 6$$

$$= 110_{10}$$

(2) $64_{16} = 6 \times 16^1 + 4 \times 16^0$

$$= 96 + 4$$

$$= 100_{10}$$

3. 10 进制到 2 进制的转换

基本方法是除 2 取余法:将 10 进制数不断用 2 除,所得余数从后往前读,得到等值的 2 进制数。

[例 1-3] 将 10 进制数 5、13、16 转换成等值的 2 进制数。

解: (1)

$$\begin{array}{rcl} 2 \overline{) 5} & \cdots \text{余} & 1 \\ 2 \overline{) 2} & \cdots \text{余} & 0 \\ 2 \overline{) 1} & \cdots \text{余} & 1 \\ \hline & & 0 \quad 1 \quad 0 \quad 1 \end{array}$$

即 $5_{10} = 101_2$

(2)

$$\begin{array}{r}
 2 \overline{) 13} \cdots \text{余 } 1 \\
 2 \overline{) 6} \cdots \text{余 } 0 \\
 2 \overline{) 3} \cdots \text{余 } 1 \\
 2 \overline{) 1} \cdots \text{余 } 1 \\
 0 \qquad 1 \ 1 \ 0 \ 1
 \end{array}$$

即 $13_{10} = 1101_2$

(3)

$$\begin{array}{r}
 2 \overline{) 16} \cdots \text{余 } 0 \\
 2 \overline{) 8} \cdots \text{余 } 0 \\
 2 \overline{) 4} \cdots \text{余 } 0 \\
 2 \overline{) 2} \cdots \text{余 } 0 \\
 2 \overline{) 1} \cdots \text{余 } 1 \\
 0 \qquad 1 \ 0 \ 0 \ 0 \ 0
 \end{array}$$

即 $16_{10} = 10000_2$

上述方法, 如将基数 2 改为 N, 则适用于将 10 进制数转换成等值的 N 进制数。

将 10 进制小数化为等值的 2 进制小数, 稍微麻烦一点。通常采用乘 2 取整法, 即将 10 进制小数, 不断用 2 乘, 每次将整数取出, 即得相应的 2 进制小数。

[例 1-4] 将 10 进制小数 0.875、0.325 转换成等值的 2 进制小数。

解: (1)

$$\begin{array}{r}
 0.875 \\
 \times 2 \\
 \hline
 1.750 \\
 \downarrow \\
 1
 \end{array}
 \qquad
 \begin{array}{r}
 0.750 \\
 \times 2 \\
 \hline
 1.500 \\
 \downarrow \\
 1
 \end{array}
 \qquad
 \begin{array}{r}
 0.5 \\
 \times 2 \\
 \hline
 1.0 \\
 \downarrow \\
 1
 \end{array}$$

即 $0.875_{10} = 0.111_2$

(2)

$$\begin{array}{r}
 0.325 \\
 \times 2 \\
 \hline
 0.650 \\
 \downarrow \\
 0
 \end{array}
 \qquad
 \begin{array}{r}
 0.65 \\
 \times 2 \\
 \hline
 1.30 \\
 \downarrow \\
 1
 \end{array}
 \qquad
 \begin{array}{r}
 0.3 \\
 \times 2 \\
 \hline
 0.6 \\
 \downarrow \\
 0
 \end{array}
 \qquad
 \begin{array}{r}
 0.6 \\
 \times 2 \\
 \hline
 1.2 \\
 \downarrow \\
 1
 \end{array}$$

即 $0.325_{10} \approx 0.0101_2$

三、16 进制

2 进制的一个缺点是位数太多, 不易识别, 写起来太麻烦。为了识别方便, 在书写计算机

表 1-2 各种进制的对照表

10 进制数	2 进制数	16 进制数
0	0	0
1	1	1
2	1 0	2
3	1 1	3
4	1 0 0	4
5	1 0 1	5
6	1 1 0	6
7	1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
10	1 0 1 0	A
11	1 0 1 1	B
12	1 1 0 0	C
13	1 1 0 1	D
14	1 1 1 0	E
15	1 1 1 1	F
16	1 0 0 0 0	10
17	1 0 0 0 1	11
18	1 0 0 1 0	12
19	1 0 0 1 1	13
20	1 0 1 0 0	14

的程序时,常将它们写成16进制数。表1-2是2进制、16进制的对照表。

1. 16进制与2进制的转换

由表1-2可见,每位16进制数可由4位2进制数表示。因此将16进制数的各位用相应的4位2进制表示,即可转换成等值的2进制数。

[例1-5] 将16进制数13AF、5C06转换成等值的2进制数。

$$\text{解: (1) } 13AF_{16} = \underline{0001} \underline{0011} \underline{1010} \underline{1111} \\ = 1001110101111_2$$

$$(2) 5C06_{16} = \underline{0101} \underline{1100} \underline{0000} \underline{0110} \\ = 101110000000110_2$$

(有效数字前的0舍掉)

2进制转换为16进制时,将2进制数从右至左每4位一组,每组用等值的16进制表示。

[例1-6] 将2进制数10110101、1001101转换成等值的16进制数。

$$\text{解: (1) } 10110101_2 = \underline{1011} \underline{0101} \\ = B5_{16}$$

$$(2) 1001101_2 = \underline{0100} \underline{1101} \\ = 4D_{16}$$

2. 书写时的规定

在书写计算机程序时,一般不是用基数作下标来区别各种进制,而是用相应的英文字母作后缀来标识各种进制的数。

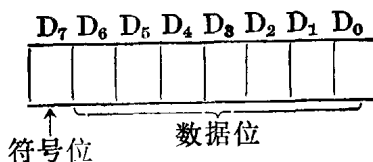
例如: B(Binary)——表示2进制数,可将 100_2 写成 $100B$;

D(Decimal)——表示10进制数,一般约定D可省略,即无后缀的数字为10进制数字 $100D=100$;

H(Hexadecimal)——表示16进制数,如 $100_{16}=100H$ 。

四、有符号数的表示方法

前面提到的2进制数,没有涉及数的正负问题。不考虑正负的数称为无符号数。工业过程控制中,表示各种开关状态的数据(如用1表示闭合,用0表示断开),属无符号数。算术运算中的数,自然会有正有负,这类数称为有符号数。为了在计算机中正确地表示有符号数,通常规定,数字的最高位为符号位,并用0表示正,用1表示负。在8位微型计算机中,一个数用8位2进制数表示,其标准格式如下:

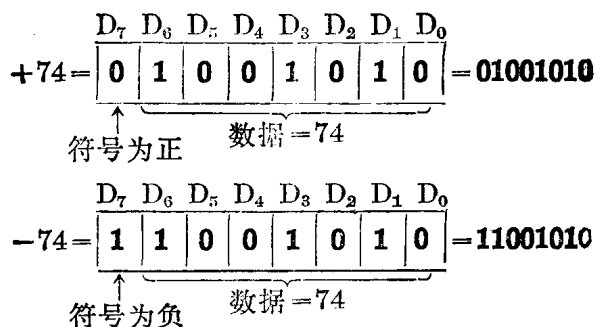


最高位 D_7 为符号位, $D_6 \sim D_0$ 为2进制数据位。

$$\text{例如: } +74 = +1001010$$

$$-74 = -1001010$$

在机器中表示为:



这种连正负号也数字化的数,称为机器数,是计算机所能识别的数。

机器数有原码、反码、补码三种表示方法,兹分述如下:

1. 原码

在符号位中用 0 表示正、用 1 表示负的 2 进制数,称为原码。

[例 1-7] 设 $x_1 = +0010101$, $x_2 = -0010101$, 试写出 x_1 和 x_2 的原码。

解: (1) $[x_1]_{\text{原}} = 0 \quad 0010101$
↑ 数值不变
↑ 用 0 表示正

(2) $[x_2]_{\text{原}} = 1 \quad 0010101$
↑ 数值不变
↑ 用 1 表示负

[例 1-8] 写出 +13、-13、+127、-127、+0、-0 的原码(用 8 位 2 进制数表示,下同)。

解: $[+13]_{\text{原}} = 0 \quad 0001101$
 $[-13]_{\text{原}} = 1 \quad 0001101$
 $[+127]_{\text{原}} = 0 \quad 1111111$
 $[-127]_{\text{原}} = 1 \quad 1111111$
 $[+0]_{\text{原}} = 0 \quad 0000000$
 $[-0]_{\text{原}} = 1 \quad 0000000$

由上面两个例子可以看出,用原码表示带符号数,其数值部分不变,易于识别,这是方便的一面;但在进行加减法运算时,比较复杂。例如,在进行加法运算时,机器首先要判断两数的符号是否相同,若相同,则做加法;若不同,则做减法。做减法时还要判断两数绝对值的大小,以便于大者减去小者。然后再根据绝对值大的数决定差值的符号。

引入反码和补码的概念,就可解决这个困难,使运算简化。

2. 反码

反码的定义是:

正数: 反码 = 原码

负数: 反码 = 原码的符号位不变而数值按位取反

所谓按位取反,就是将各位的 1 变成 0, 0 变成 1。

[例 1-9] 求 +13、-13 的反码。

解: $[+13]_{\text{反}} = [+13]_{\text{原}}$
 $= 0 \quad 0001101$

$$\begin{array}{rcl}
 [-13]_{\text{原}} & = & 1\ 0001101\text{B} \\
 & \downarrow & \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\
 [-13]_{\text{反}} & = & 1\ 1110010\text{B} \\
 & \uparrow & \uparrow \\
 & & \text{数值按位取反} \\
 & & \text{符号位不变}
 \end{array}$$

即: $[-13]_{\text{反}} = 11110010\text{B}$

[例 1-10] 求 $[+0]_{\text{反}}$ 、 $[-0]_{\text{反}}$ 、 $[[-0]_{\text{反}}]_{\text{反}}$ 。

解: $[+0]_{\text{反}} = [+0]_{\text{原}}$

$$= 0\ 0000000\text{B}$$

$$[-0]_{\text{原}} = 1\ 0000000 \quad \leftarrow \text{符号位不变, 数值按位取反}$$

$$[-0]_{\text{反}} = 1\ 1111111 \quad \leftarrow$$

$$[[-0]_{\text{反}}]_{\text{反}} = 1\ 0000000 \quad \leftarrow \text{符号位不变, 数值按位取反}$$

可见, $[[-0]_{\text{反}}]_{\text{反}} = [-0]_{\text{原}}$, 也就是说, 0 的反码的反码等于原码, 这个结论可以推广为:

$$[[x]_{\text{反}}]_{\text{反}} = [x]_{\text{原}} \quad (1-2)$$

其正确性, 读者可自行验证。

3. 补码

补码的定义是:

正数: 补码 = 原码

负数: 补码 = 反码 + 1

[例 1-11] 求 +13、-13、+127、-127 的补码。

解: 一般采用先由原码求反码, 再求补码的方法。

$$(1) [+13]_{\text{补}} = [+13]_{\text{原}}$$

$$= 0\ 0001101$$

$$(2) [-13]_{\text{原}} = 1\ 0001101 \quad \leftarrow \text{符号位不变, 数值按位取反}$$

$$[-13]_{\text{反}} = 1\ 1110010 \quad \leftarrow$$

$$[-13]_{\text{补}} = 1\ 1110011 \quad \leftarrow \text{加 1}$$

$$(3) [+127]_{\text{补}} = [+127]_{\text{原}}$$

$$= 0\ 1111111$$

$$(4) [-127]_{\text{原}} = 1\ 1111111 \quad \leftarrow \text{符号位不变, 数值按位取反}$$

$$[-127]_{\text{反}} = 1\ 0000000 \quad \leftarrow$$

$$[-127]_{\text{补}} = 1\ 0000001 \quad \leftarrow \text{加 1}$$

[例 1-12] 求 $[-3]_{\text{补}}$ 、 $[[-3]_{\text{补}}]_{\text{补}}$ 。

解: (1) $[-3]_{\text{原}} = 10000011 \quad \textcircled{1}$

$$[-3]_{\text{反}} = 11111100$$

$$[-3]_{\text{补}} = 11111101$$

(2) 设 $x = [-3]_{\text{补}} = 11111101$, 则

$$[x]_{\text{原}} = 11111101$$

$$[x]_{\text{反}} = 10000010$$

$$[x]_{\text{补}} = 10000011$$

$$\text{即 } [x]_{\text{补}} = [[-3]_{\text{补}}]_{\text{补}} = 10000011 \quad \textcircled{2}$$

比较 ①② 两式可知, $[[-3]_{\text{补}}]_{\text{补}} = [-3]_{\text{原}}$ 。即其补码的补码等于原码, 推广之, 有

$$[[x]_{\text{补}}]_{\text{补}} = [x]_{\text{原}} \quad (1-3)$$

结论的正确性,读者可自行验证。

上面简要地介绍了原码、反码和补码。目的在于解决带符号数的表示方法和计算问题。它们的特点是:

- (1) 正负符号数字化。规定数字的最高位为符号位,用 2 进制数 0 表示正、1 表示负。
- (2) 数值为正时,三种机器数相同,即

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

- (3) 数值为负时,三种机器数的符号位均为 1,但后面的数值部分各不相同:

原码的数值部分与原数值相同;

反码的数值部分,为原数值按位取反;

补码的数值部分,为原数值按位取反后加 1。

表 1-3 列出了 8 位机器数与无符号数、有符号数的原码、反码、补码的对照表。

表 1-3 8 位机器数对照表

8 位机器数		无符号数	有 符 号 数		
2 进制数	16 进制数		原 码	反 码	补 码
00000000	00	0	+0	+0	+0
00000001	01	1	+1	+1	+1
00000010	02	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮	⋮
01111100	7C	124	+124	+124	+124
01111101	7D	125	+125	+125	+125
01111110	7E	126	+126	+126	+126
01111111	7F	127	+127	+127	+127
10000000	80	128	-0	-127	-128
10000001	81	129	-1	-126	-127
10000010	82	130	-2	-125	-126
⋮	⋮	⋮	⋮	⋮	⋮
11111100	FC	252	-124	-3	-4
11111101	FD	253	-125	-2	-3
11111110	FE	254	-126	-1	-2
11111111	FF	255	-127	-0	-1

在计算机中,负数一律用补码表示。

利用补码,就可将减法运算变为加法运算,兹举例说明如下:

[例 1-13] $35-24=11$

解: 用减法:

$$\begin{array}{r} 00100011 \leftarrow 35 \\ - 00011000 \leftarrow 24 \\ \hline 00001011 \leftarrow 11 \end{array}$$

用加法:

$$\begin{array}{r} 00100011 \leftarrow 35 \\ + 11101000 \leftarrow [-24]_{\text{补}} \\ \hline 100001011 \leftarrow 11 \end{array}$$

↑
自然丢失

在 8 位机中, 最高位的进位, 已超出机器字长的范围, 所以自然丢失。减法运算与补码的加法运算, 结果相同。

[例 1-14] $18-14=4$

解:

$$\begin{array}{r} 00010010 \leftarrow 18 \\ + 11110010 \leftarrow [-14]_{\text{补}} \\ \hline 100000100 \leftarrow 4 \\ \uparrow \\ \text{自然丢失} \end{array}$$

[例 1-15] $(-5)+(-7)=-12$

解:

$$\begin{array}{r} 11111011 \leftarrow [-5]_{\text{补}} \\ + 11111001 \leftarrow [-7]_{\text{补}} \\ \hline 111110100 \leftarrow [-12]_{\text{补}} \\ \uparrow \\ \text{丢失} \end{array}$$

在和中, 符号位为 1, 故结果是负数, 将结果的数值部分按位取反加 1, 求得原码为 10001100, 即为 -12。

[例 1-16] $13-14=-1$

解:

$$\begin{array}{r} 00001101 \leftarrow 13 \\ + 11110010 \leftarrow [-14]_{\text{补}} \\ \hline 11111111 \leftarrow [-1]_{\text{补}} \end{array}$$

可见, 引入补码的概念以后, 加减法运算都可用加法来实现; 数值的符号位也当作数值处理, 一道参加运算, 非常方便。因此, 在计算机中, 加减法一般都采用补码运算。

五、2-10 进制编码

如前所述, 计算机中采用 2 进制, 而人们却习惯用 10 进制。为了便于人机联系, 通常还采用一种 2-10 进制的编码方式。

2-10 进制的本质是 10 进制, 具有 10 进制的特点; 但每位 10 进制数用相应的 4 位 2 进制码表示, 又具有 2 进制的形式。实际上, 2-10 进制是一种用 2 进制编码的 10 进制数(Binary Coded Decimal), 简称 BCD 码。

2-10 进制编码的方法很多, 最常用的是 8421 码。

4 位 2 进制数能表示 0000~1111 共 16 种状态。按自然顺序舍去了后面六种状态(1010~1111), 只取前面 10 种状态(0000~1001), 用来表示 0~9 的 10 进制数, 自左至右每位 2 进制数字的权分别为 8、4、2、1, 这种码制称为 8421 码, 如表 1-4 所示。8421 码中, 被舍去的 6 种状态称为非法码。

表 1-4 2-10 进制编码

10 进制数	8421 码(BCD 码)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

由表 1-4 可以看出, 对于一位 10 进制数, 其结果与 2 进制完全相同。但如果是两位以上的 10 进制数, 就不相同了。例如 10 进制数的 12, 其 BOD 码为 8 位: 00010010, 而不是 2 进制的 1100。

BOD 码计数法, 比较直观, 易于识别。例如 5678 的 BOD 码为:

$$\begin{array}{r} 0101 \\ \hline 5 \end{array} \quad \begin{array}{r} 0110 \\ \hline 6 \end{array} \quad \begin{array}{r} 0111 \\ \hline 7 \end{array} \quad \begin{array}{r} 1000 \\ \hline 8 \end{array}$$

BOD 码既满足了计算机必须使用 2 进制的要求, 又保持了 10 进制的方便形式, 因此得到了广泛的应用。

在用 BOD 码进行计算时, 进位规则与 4 位 2 进制的进位规则不同。4 位 2 进制是逢 16 进 1, 8421 码是 10 进制, 逢 10 进 1。两数相加结果大于 9 时, 就出现非法码。例如:

$$34 + 9 = 43$$

十进制

$$\begin{array}{r} 34 \\ + 9 \\ \hline 43 \end{array}$$

8421 码

$$\begin{array}{r} 0011 \quad 0100 \\ + \quad \quad 1001 \\ \hline 0011 \quad 1101 \\ \text{十位} \quad \text{个位} \end{array}$$

可见, 个位数相加结果大于 9, 出现了非法码, 整个结果都错了, 因此要进行修正。4+9=13, 用 8421 码表示为 00010011, 原计算结果中的“1101”比“00010011”要小 6。因此为了得到正确的结果, 应进行加 6 修正。即

$$\begin{array}{r} 0011 \quad 1101 \leftarrow \text{原结果} \\ + \quad \quad 0110 \leftarrow \text{加 6 修正} \\ \hline 0100 \quad 0011 = 43 \leftarrow \text{最终结果} \end{array}$$

【例 1-17】用 8421 码求

$$48 + 69 = 117$$

解:

$$\begin{array}{r} 0100 \quad 1000 \\ + 0110 \quad 1001 \\ \hline 1011 \quad 0001 \\ + 0110 \quad 0110 \leftarrow \text{加 66 修正} \\ \hline 10001 \quad 0111 = 117 \end{array}$$

此题中, 个位数为 10001, 有进位, 须加 6 修正, 而十位数出现了非法码 1011, 也要作加 6 修正。

对减法运算则用减 6 修正, 规律与加法相同。

【例 1-18】用 8421 码求 $34 - 9 = 25$

$$\begin{array}{r} 0011 \quad 0100 \leftarrow 34 \\ - 0000 \quad 1001 \leftarrow 9 \\ \hline 0010 \quad 1011 = 25 \leftarrow \text{个位是非法码} \\ - \quad \quad 0110 \quad \leftarrow \text{减 6 修正} \\ \hline 0010 \quad 0101 = 25 \end{array}$$

在计算机中, 一般都有专门的指令来完成 BOD 码运算中的加 6、减 6 修正功能(称为 10 进制调整)。

六、字符代码

在计算机中, 除数字外, 还需处理各种字符, 如字母、运算符号、标点符号等等。这些字符