全国计算机等级考试四级教程

——计算机组成与接口

（2017年版）

教育部考试中心

高等教育出版社

全国计算机等级考试四级教程

——计算机组成与接口

（2017年版）

Quanguo Jisuanji Dengji Kaoshi Siji Jiaocheng

——Jisuanji Zucheng yu Jiekou

教育部考试中心

郑雪峰　赵海春　郑榕　编著

高等教育出版社·北京

内容简介

本书是根据教育部考试中心颁布的《全国计算机等级考试四级计算机组成与接口考试大纲（2013年版）》的要求编写的。教材内容紧扣考试大纲要求，力求做到简明扼要，在介绍基本概念的同时，把四级计算机组成与接口考试所要求的硬件知识贯穿在教材中。教材的知识包含三部分内容：计算机基本知识；汇编语言程序设计方法和计算机接口程序设计技术。前两部分内容是学好第三部分的基础。本教材的内容是计算机组成与接口的基础知识，主要包括微型计算机的基本构成、数据的表示方法、存储器的工作原理和CPU的连接方法、I/O接口芯片的工作原理和程序设计方法以及总线类型等。深入理解和熟练掌握本教材内容，是成为一名优秀的硬件开发工程师所必需的。

本书除作为四级计算机组成与接口考试参考用书外，也可作为高等院校计算机专业“微机原理与应用”课程的教材，同时可作为电子信息、物联网、通信、自动化、测控专业“微机原理”课程的教材。

图书在版编目(CIP)数据

全国计算机等级考试四级教程:2017年版.计算机组成与接口/教育部考试中心编.--北京:高等教育出版社,2016.11

ISBN 978-7-04-046569-3

Ⅰ.①全… Ⅱ.①教… Ⅲ.①电子计算机-水平考试-教材②计算机组成原理-水平考试-教材③电子计算机-接口-水平考试-教材 Ⅳ.①TP3

中国版本图书馆CIP数据核字(2016)第235168号

策划编辑 何新权　　责任编辑 何新权　　封面设计 王 琰　　版式设计 范晓红

责任校对 吕红颖　　责任印制 尤 静

出版发行 高等教育出版社
社　　址 北京市西城区德外大街4号
邮政编码 100120
印　　刷 北京明月印务有限责任公司

开　　本 787mm×1092mm 1/16
印　　张 16.25
字　　数 390千字
购书热线 010-58581118

咨询电话 400-810-0598
网　　址 http://www.hep.edu.cn
　　http://www.hep.com.cn
网上订购 http://www.hepmall.com.cn
　　http://www.hepmall.com
　　http://www.hepmall.cn
版　　次 2016年11月第1版
印　　次 2016年11月第1次印刷
定　　价 34.00元

物 料 号 46569-00

前　言

随着计算机应用技术的普及以及工业信息化的发展，根据生产应用需求，利用廉价的电子元器件开发专用的嵌入式硬件产品已经成为目前广泛的市场需求。

本书针对希望掌握计算机硬件开发技术的读者，内容组织上不要求最新和最全，而以掌握开发技术为目标，只要具备了电路知识的读者都能轻松地阅读本书内容，熟悉了解硬件设计的方法和接口芯片程序设计的方法，并通过实践最终成为一名优秀的硬件开发工程师。

全书分成8章，第一章介绍计算机组成的基本知识、计算机中的数据表示方法以及CPU的基本结构。尽管CPU的内部技术不断发展，功能越来越强，然而从硬件开发角度出发，更需要关心的是硬件设计的基本原理。因而，本章对功能更强但原理复杂的Pentium CPU不作重点介绍，而是把重点放在8086 CPU编程结构的介绍上。第二章介绍存储器的基本类型和与CPU的连接方式。CPU和存储器是构成嵌入式系统硬件的基本单元，只要具备这些基本单元，硬件系统就可以工作了。第三章介绍汇编语言程序设计。由于目前在硬件系统开发中汇编语言的应用越来越少，大量的开发软件已用C语言来编写，所以本章只是重点介绍8086 CPU的指令系统，而对Pentium CPU中的在8086 CPU指令系统上所增加的指令不作介绍。本章是学习第四、五章的基础。第四章介绍中断技术。中断技术是硬件系统实时响应外部现场事件的入口点，也是内部任务转换的触发事件。第五章至第八章介绍CPU与外部不同设备的数据交换方式，可以根据所设计硬件系统的具体要求，有选择地掌握这四章中的部分内容。

本书第一章、第二章和第八章由赵海春执笔，第六章和第七章由郑榕执笔，其余章节由郑雪峰执笔并负责全书的修改和最终定稿。

北京工业大学蒋宗礼教授在百忙之中审阅了全书，并提出了许多宝贵的意见，在此表示衷心的感谢。

由于编写仓促，疏误之处殷切希望得到读者的批评指正。

编者

目　录

第一章　计算机系统概述

导入语

本章主要介绍了计算机系统的基本结构、计算机硬件的主要技术指标、指令格式、指令的寻址方式、指令流水线技术以及8086 CPU的功能结构和时序。

考核内容

了解冯·诺依曼结构和哈佛结构计算机的差异以及计算机硬件的主要技术指标。掌握指令寻址方式、流水线技术和8086 CPU的功能结构和时序。

1.1　计算机的基本组成

1.1.1　硬件系统

计算机是由硬件系统和软件系统两大部分组成的。硬件系统是指计算机中那些看得见、摸得着的物理实体，主要包括中央处理器(Central Processing Unit，CPU)、存储器、输入/输出(Input/Output，I/O)设备等。

1. 冯·诺依曼结构计算机的特点

1945年，美籍匈牙利数学家冯·诺依曼(John von Neumann)在研究EDVAC(Electronic Discrete Variable Automatic Computer)机时提出了“存储程序”的概念，其基本思想是：将编好的程序和要处理的数据事先存放在存储器中，然后启动计算机工作，计算机应能在不需要人干预的情况下自动完成逐条指令取出和执行。以此概念为基础所设计的各类计算机称为冯·诺依曼结构计算机，也称普林斯顿结构计算机。其特点可归结如下：

(1) 计算机硬件由运算器、存储器、控制器、输入设备和输出设备五大部件组成。

(2) 指令和数据在计算机中均以二进制数表示，并存储在同一个存储器内，它们在形式上没有差别，可按其存储地址进行寻址访问。

(3) 指令由操作码和地址码组成，操作码用来表示操作的性质，地址码用来表示操作数在存储器中的位置。

(4) 指令在存储器内按顺序存放。通常，指令按顺序执行，特定条件下，可根据运算结果或设定的条件改变执行顺序。

(5) 机器以运算器为中心，输入/输出设备与存储器间的数据传送通过运算器完成。

原始的冯·诺依曼结构计算机以运算器为中心，现已转变为以存储器为中心，如图1.1所示。

在该结构中，指令和数据共享同一总线，致使信息流的传输成为影响计算机系统性能的瓶

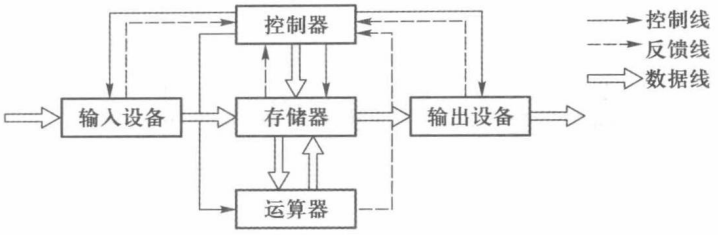

图 1.1 以存储器为中心的计算机结构

颈,限制了计算机数据处理速度的提高。

使用冯·诺依曼结构的中央处理器有很多,如 Intel 公司的 80X86、ARM 公司的 ARM7、MIPS 公司的 MIPS 等。

2. 哈佛结构计算机的特点

冯·诺依曼结构采用一种传统的存储器设计思想。该结构在面对高速、实时处理时,不可避免地会造成总线拥挤。为此,哈佛大学提出了与冯·诺依曼结构完全不同的另一种存储器设计思想,习惯上称之为哈佛结构。哈佛结构的指令和数据是完全分开的,存储器分为两部分,一部分是程序存储器,用于存放指令;另一部分是数据存储器,用于存放数据。程序存储器和数据存储器是两个独立的存储器,每个存储器独立编址、独立访问。哈佛结构至少有两组总线:程序存储器(PM)的数据总线和地址总线,数据存储器(DM)的数据总线和地址总线,如图 1.2 所示。这种分离的程序总线和数据总线允许在一个机器周期内同时获取指令字(来自程序存储器)和操作数(来自数据存储器),所以哈佛结构的中央处理器通常具有较高的执行效率。由于指令和数据是分开组织和存储的,所以本条指令执行时可以预先读取下一条指令,同时,还可以使指令和数据有不同的数据宽度,如 Microchip 公司的 PIC16 芯片的指令宽度为 14 位,而数据宽度为 8 位。使用哈佛结构的中央处理器有很多,例如,Motorola 公司的 MC68 系列,Zilog 公司的 Z8 系列,ARM 公司的 ARM9、ARM10 和 ARM11。大多数单片机和数字信号处理(DSP)系统都使用哈佛结构。

目前,许多现代微型计算机中的高速缓冲存储器(Cache)采用哈佛结构,将一级 Cache 分为指令 Cache 和数据 Cache 两个部分,而主存储器采用冯·诺依曼结构,由指令和数据合用同一个主存。这样,将冯·诺依曼结构和哈佛结构结合起来,不仅可以提高主存储器利用率,而且可以提高程序执行的效率,减少指令执行的时钟周期数。

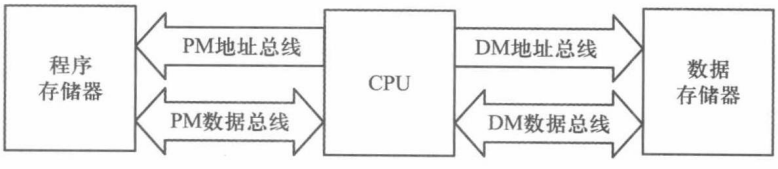

图 1.2 哈佛结构的存储器设计

1.1.2 软件系统

计算机软件通常是指计算机所配置的各类程序和文件，它们是存放在内存或外存中的二进制编码信息。软件是相对于硬件而言的，它们是不能直接触摸的。软件一般可分为两大部分：系统软件和应用软件。

1. 系统软件

系统软件是用于管理、控制和维护计算机系统资源（硬件和软件）的程序集合。系统软件主要包括以下四类。

1）操作系统

操作系统是最重要的系统软件，它是管理计算机硬件和软件资源、控制程序运行、改善人机交互并为应用软件提供支持的一种软件。通常，操作系统包括五大功能：处理机管理、存储管理、文件管理、设备管理及作业管理。

2）语言处理程序

语言处理程序一般是由汇编程序、编译程序、解释程序和相应的操作程序等组成。它是为用户设计的编程服务软件，其作用是将用户的源程序翻译成计算机能识别的目标程序，例如 C 语言编译器等。

3）数据库管理系统

数据库管理系统（DataBase Management System）是一种操纵和管理数据库的大型软件，用于建立、使用和维护数据库，简称 DBMS，例如 Oracle 数据库管理系统等。

4）服务支持软件

服务支持软件是帮助用户使用和维护计算机的软件，为系统提供许多功能，包括各种调试程序、诊断程序、硬件维护程序和网络管理程序等。

2. 应用软件

应用软件是为了某一专用目的而开发的软件。它包括商品化的通用软件和专用软件两种。

1.1.3 计算机系统层次结构

计算机系统以硬件为基础，通过各种软件来扩充系统功能，形成了一个由硬件和软件组成的综合复杂体。从系统结构的角度，可将计算机系统划分为如图 1.3 所示的分层虚拟结构。

第 0 级是硬联逻辑级。这是计算机的内核，由门、触发器等逻辑电路组成。

第 1 级是微程序级。这一级的机器语言是微指令集，程序员用微指令编写的微程序一般是由硬件直接执行的。

第 0 级和第 1 级是机器的核心部分，某些机器不用微程序解释执行指令，直接用硬件逻辑实现机器指令系统，则这两级合而为一。有些机器用毫微程序实现微程序，则第 0 级还可划分为两级。

第 2 级是传统机器级。这一级的机器语言是机器的指令集，程序员用机器指令编写的程序可以由微程序进行

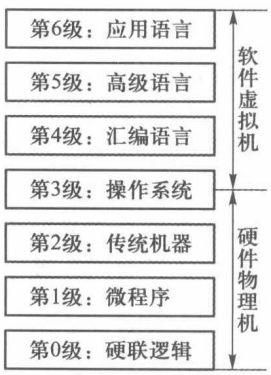

图 1.3 计算机系统的分层虚拟结构

解释。

第3级是操作系统级。这一级的机器语言中的多数指令是传统机器的指令(例如算术运算、逻辑运算等指令),此外,这一级还提供操作系统级指令(例如打开文件、读/写文件、关闭文件等指令)。因此,用这一级语言编写的程序中,与第2级指令相同的指令直接由微程序实现;而操作系统级指令部分由操作系统进行解释。操作系统是运行在第2级上的解释程序。

第4级是汇编语言级。这一级的机器语言是汇编语言,完成汇编语言翻译的程序称为汇编程序。

第5级是高级语言级。这一级的机器语言就是各种高级语言,用这些高级语言所编写的程序一般由编译程序来完成编译工作,只有个别高级语言是用解释的方法实现的。

第6级是应用语言级。这一级是为了使计算机满足某种用途而专门设计的,因此这一级的语言就是各种面向问题的应用语言。

把计算机系统划分成多级分层虚拟结构,有利于正确理解计算机系统的工作过程,明确软件、硬件在计算机系统中的地位和作用。

1.2 计算机硬件的主要技术指标

1.2.1 机器字长

机器字长是指CPU一次能传送或处理的二进制数据的位数,它反映了CPU中定点运算数据通路、定点运算器和CPU内通用寄存器的宽度。因为在一次运算中,操作数和运算结果通过数据总线在寄存器和运算部件之间传送。通常,机器字长越长,数的表示范围越大,计算的精度也越高。机器字长也会影响机器的运算速度。若CPU字长较短,而运算数据的位数较多,那么需要经过两次或多次运算才能完成,导致影响机器的运算速度。

1.2.2 主存容量

主存容量是指主存所能存储的全部信息量,通常表示为:

存储容量=存储单元个数 × 每个单元的位数

对于字节编址的计算机,由于一个字节(Byte,B)已被定义为8位二进制位,所以,存储容量就以字节数来表示。

1.2.3 运算速度

计算机的运算速度与许多因素有关,如机器的主频、所执行的操作类型、主存的存取速度等。

执行时间(响应时间、延迟时间)和吞吐率(执行速度)是衡量计算机性能的基本指标。早期用完成一次定点加法运算所需的时间来衡量运算速度。后来采用吉普森(Gipson)法,它综合考虑每条指令的执行时间以及它们在全部操作中所占的百分比,即

$$T_M = \sum_{i=1}^{n} f_i t_i$$

其中,T_M 为机器运行速度,f_i 为第 i 种指令占全部操作的百分比数,t_i 为第 i 种指令的执行

时间。

除了时间衡量标准之外，机器的运算速度还普遍采用单位时间内执行指令的平均条数来衡量。用 MIPS(Million Instructions Per Second，每秒百万条指令)作为计量单位(例如，某计算机每秒能执行 100 万条指令，则记作 1 MIPS)；或用 MFLOPS(Million Floating Point Operations Per Second，每秒百万次浮点运算)等来衡量运算速度。

1.3 数据信息的表示

1.3.1 常用记数制及其相互转换

日常生活中广泛使用十进制数，而计算机内部则以二进制数作为数据表示的基础，在此基础上，人们有时也采用八进制、十六进制等表示，以便于阅读。

1. 进位记数制

进位记数制又称为数制，即按进位制的原则进行计数。数制由两大要素组成：基数 R 和各数位的权 W。基数 R 决定了数制中各数位上允许出现的数码个数，基数为 R 的数制称为 R 进位制。权 W 则表明该数位上的数码所表示的单位数值的大小。

假设任意数值 N 用 R 进制数来表示，则用形式为 $n+k$ 个自左向右排列的符号来表示。

$$N=(D_{n-1}D_{n-2}\cdots D_0.D_{-1}D_{-2}\cdots D_{-k})_R$$

式中 $D_i(-k\leqslant i\leqslant n-1)$ 为该数制采用的基本符号，可取值 0，1，2，…，$R-1$，小数点处于 D_0 与 D_{-1} 之间，整数部分有 n 位，小数部分有 k 位，数值 N 的实际值为

$$N=\sum_{i=-k}^{n-1}(D_i\times R^i)$$

通常最左边的数位 D_{n-1} 的权最大，称为最高有效位；最右边的数位 D_{-k} 的权最小，称为最低有效位。

1) 十进制(Decimal)

十进制数共有 10 个数字符号，即 0～9，它的基数为 10，逢十进一。任意一个十进制数可以表示为

$$(N)_{10}=\sum_{i=-k}^{n-1}D_i\times 10^i$$

例如，十进制数 135.96 可以表示为：

$$135.96=1\times10^2+3\times10^1+5\times10^0+9\times10^{-1}+6\times10^{-2}$$

2) 二进制(Binary)

在二进制中使用的数字只有 0 和 1，基数为 2，逢二进一。任意一个二进制数可以表示为

$$(N)_2=\sum_{i=-k}^{n-1}D_i\times 2^i$$

例如，二进制数$(1100.1001)_2$ 可以表示为

$$(1100.1001)_2=1\times2^3+1\times2^2+0\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+0\times2^{-3}+1\times2^{-4}$$

3) 八进制(Octal)

八进制数共有 8 个数字符号，即 0～7，基数为 8，逢八进一。任意一个八进制数可以表示为

$$(N)_8 = \sum_{i=-k}^{n-1} D_i \times 8^i$$

4）十六进制（Hexadecimal）

十六进制共有 16 个数字符号，即 0～9 和 A、B、C、D、E、F（依次表示 10～15），基数为 16，逢十六进一。任意一个十六进制数可以表示为

$$(N)_{16} = \sum_{i=-k}^{n-1} D_i \times 16^i$$

2. 数制之间的转换

1）非十进制转化为十进制

非十进制数转化为十进制数的方法是将非十进制数按权展开，然后求和。

【例 1.1】 将非十进制数转化为十进制。

（1）$(100.101)_2 = 1\times2^2+0\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

（2）$(123.56)_8 = 1\times8^2+2\times8^1+3\times8^0+5\times8^{-1}+6\times8^{-2}$

（3）$(AB)_{16} = 10\times16^1+11\times16^0$

2）十进制转化为非十进制

十进制数转化为非十进制数的方法是将十进制数的整数部分和小数部分分别转换，最后将结果写到一起。

（1）十进制整数转化为非十进制整数的方法

设 N 为 n 位十进制整数，则 N 转化为非十进制（R 进制）的形式为

$$\begin{aligned}(N)_{10} &= D_0\times R^0+D_1\times R^1+D_2\times R^2+\cdots+D_{n-1}\times R^{n-1}\\ &= D_0+R(D_1+D_2\times R^1+\cdots+D_{n-1}\times R^{n-2})\end{aligned}$$

所以，

$$D_0 = \mathrm{mod}\ ((N)_{10}, R)$$

同理，

$$\begin{aligned}D_1 &= \mathrm{mod}((N)_{10}, R^2)\\ D_2 &= \mathrm{mod}\ ((N)_{10}, R^3)\\ &\cdots\cdots\\ D_{n-1} &= \mathrm{mod}((N)_{10}, R^n)\end{aligned}$$

其中，mod( ) 为取余函数。

由此得到，十进制整数转化为非十进制（R 进制）整数的方法一般是采用除 R 取余数法。其规则为：将十进制数除以 R，所得余数即为对应 R 进制数最低位的值。然后将上次所得的商除以 R，所得余数即为 R 进制数次低位的值，如此进行下去，直到商等于 0 为止，最后得出的余数是所求 R 进制数最高位的值。

（2）十进制小数转化为非十进制小数的方法

假如 M 为 k 位十进制小数，则 M 转化为非十进制（R 进制）的形式为

$$(M)_{10} = D_{-1}\times R^{-1}+D_{-2}\times R^{-2}+D_{-3}\times R^{-3}+\cdots+D_{-k}\times R^{-k}$$

两边同时乘以 R，得

$$(M)_{10}\times R=D_{-1}\times R^{0}+D_{-2}\times R^{-1}+D_{-3}\times R^{-2}+\cdots+D_{-k}\times R^{-k+1}$$

比较上面的两个表达式会发现，D_{-1}所对应的权变成了 R^0，即小数最高位的权。

将 D_{-1}从$(M)_{10}\times R$ 中去掉，然后两边再同时乘以 R，得

$$((M)_{10}\times R-D_{-1})\times R=D_{-2}\times R^{0}+D_{-3}\times R^{-1}+\cdots+D_{-k}\times R^{-k+2}$$

同理，D_{-2}所对应的权变成了 R^0。

重复上述过程，直至达到所要求的数据精度或小数部分为 0 时为止。

由此得到，十进制小数转化为非十进制小数的方法是乘 R 取整法。其规则为：将十进制数乘以 R，所得乘积的整数部分即为对应 R 进制小数最高位的值，然后对所余的小数部分乘以 R，所得乘积的整数部分为次高位的值，如此进行下去，直到乘积的小数部分为 0，或结果已满足所要求的数据精度为止。

所以，十进制数

$$(N.M)_{10}=(D_{n-1}D_{n-2}\cdots D_{0}.D_{-1}D_{-2}\cdots D_{-k})_{R}$$

3）二进制与八进制、十六进制的转换

3 位二进制数组成 1 位八进制数，将二进制数转换为八进制数时，从小数点开始，分别向两边把整数部分和小数部分每 3 位分为一组。若整数最高位的一组不足 3 位，则在其左边补 0；若小数最低位的一组不足 3 位，则在其右边补 0。然后将每组二进制数用对应的八进制数代替，即可得到转换结果。

将八进制转换为二进制的方法与上述过程相反，每位八进制数转换为 3 位二进制数。

将二进制数转换为十六进制数的方法与二进制转换为八进制的方法类似，从小数点开始，分别向两边每 4 位为一组。若整数最高位的一组不足 4 位，则在其左边补 0；若小数最低位的一组不足 4 位，则在其右边补 0。然后将每组二进制数用对应的十六进制数代替，即可得到转换结果。

十六进制转换为二进制只需将每位十六进制数转换为 4 位二进制数即可。

【例 1.2】 将$(11010.11001)_2$转换为八进制、十六进制数。

$$(11010.11001)_2=(\underline{011}\ \underline{010}.\underline{110}\ \underline{010})_2=(32.62)_8$$

$$(11010.11001)_2=(\underline{0001}\ \underline{1010}.\underline{1100}\ \underline{1000})_2=(1A.C8)_{16}$$

【例 1.3】 将$(28F.6D)_{16}$转换为二进制数。

$$(28F.6D)_{16}=(\underline{0010}\ \underline{1000}\ \underline{1111}.\underline{0110}\ \underline{1101})_2$$

1.3.2 真值和机器数

计算机中参与运算的数据可以分为两大类：无符号数和有符号数，通常暂存在寄存器中，通常寄存器的位数为机器字长。

所谓无符号数，即没有符号的数，在寄存器中的每一位均可存放数值。当存放有符号数时，则需要留出位置存放符号。对有符号数而言，符号的“正”、“负”机器是无法识别的，所以符号的“正”、“负”也需要数字化，一般用“0”表示正号，用“1”表示负号。并且规定将它放在有效数字的前面，即组成了有符号数。把符号“数字化”的数，称为机器数。而把带“+”或“-”符号的数称为真值。例如，真值为+0.1011，机器数为 0.1011；真值为-0.1011，机器数为 1.1011。

1.3.3 二—十进制编码

在计算机中,信息的概念是广泛的,除了数值信息外,还有大量的非数值信息,例如逻辑数据、十进制数字信息、文字信息等。这些信息在计算机中用二进制代码表示。

使用二进制数表示十进制数的方法,通常称之为二—十进制编码(BCD 码)。常用的 BCD 码分为有权码和无权码。通常计算机是用 4 位二进制数表示 1 位十进制数。4 位二进制数有 16 种状态,从中选择 10 种来表示十进制数 0 ~ 9 的 10 种状态。

1) 有权码

所谓有权码就是表示十进制数的二进制码的每一位都有确定的权(即每一位的权是给定的)。表 1.1 列出了几种常用的有权 BCD 码。

表 1.1 4 位有权码

十进制数	8421 码	2421 码	5211 码	84-2-1 码	4311 码
0	0000	0000	0000	0000	0000
1	0001	0001	0001	0111	0001
2	0010	0010	0011	0110	0011
3	0011	0011	0101	0101	0100
4	0100	0100	0111	0100	1000
5	0101	1011	1000	1011	0111
6	0110	1100	1010	1010	1011
7	0111	1101	1100	1001	1100
8	1000	1110	1110	1000	1110
9	1001	1111	1111	1111	1111

表 1.1 中,8421 码是最常用的有权码,它是用 0000,0001,0010,…,1001 分别表示 0 ~ 9,4 位二进制编码的每位的权值从左至右分别是 8、4、2、1,因此称为 8421 码。其他编码的每位权值均与其编码名称中的数字相对应。例如,84-2-1 码,每位的权值从左至右分别是 8、4、-2、-1。

8421 码的算术运算需要修正,修正的规则是:如果两个 8421 码之和小于或等于 9 时,则不需要修正;当结果大于 9 时,要对结果加 6 修正,并向高位进位。进位可以是第一次相加或是修正产生的。

对于 2421 码、5211 码、4311 码,任何 2 个十进制数位,采用这三种编码的任何一种编码,它们相加之和等于或大于 10 时,其结果的最高位都向左产生进位,小于 10 时则不产生进位。这一特点有利于实现“逢十进一”的计数和加法规则。

2) 无权码

十进制无权码是指表示每个十进制数位的 4 位二进制码的每一位没有确定的权。常用的无权码有余 3 码(Excess-3 Code)和格雷码(Gray Code),如表 1.2 所示。余 3 码是 8421 码每个编码都加上 0011 形成的,运算规则是:当 2 个余 3 码相加不产生进位时,结果要减去 0011,产生进

位时，进位信号送入高位，本位加 0011。

表 1.2 4 位无权码

十进制数	余 3 码	格雷码(1)	格雷码(2)	格雷码(3)
0	0011	0000	0000	0000
1	0100	0001	0001	0100
2	0101	0011	0011	0110
3	0110	0010	0010	0010
4	0111	0110	0110	1010
5	1000	0111	1110	1011
6	1001	0101	1010	0011
7	1010	0100	1000	0001
8	1011	1100	1100	1001
9	1100	1000	0100	1000

格雷码也称为循环码，其任何相邻的两组代码中仅有一个二进制数位不同，其余各位均相同，因而又称作单位距离码。因此，当从一个编码变到下一个相邻编码时，只有一位发生变化，出现错误的概率小，也有利于得到更好的译码波形。

1.3.4 ASCII 码

在计算机中使用的英文字母、数字、标点符号及一些特殊符号等统称为“字符”(Character)。所有字符的集合称作“字符集”。字符不能直接在计算机内部进行处理，因而也必须对其进行数字化编码，字符集中的每一个字符都有一个唯一的代码(二进制编码 0/1 序列)，构成了该字符集的编码表。

用二进制代码表示字符的编码方式有很多，其中，美国国家标准信息交换码(American Standard Code for Information Interchange，ASCII)是目前国际上使用最广泛的计算机字符编码。ASCII 字符编码如表 1.3 所示。

表 1.3 ASCII 字符编码表

$b_3b_2b_1b_0$ \ $b_6b_5b_4$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	、	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u

续表

$b_3b_2b_1b_0$ \ $b_6b_5b_4$	000	001	010	011	100	101	110	111
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

ASCII 码的编码规则：每个字符用 7 位二进制数（$b_6b_5b_4b_3b_2b_1b_0$）来表示，7 位二进制共有 128（2^7）种状态，可表示 128 个字符，7 位编码的取值范围为 0000000 ~ 1111111。在计算机内，每个字符的 ASCII 码用 1 B（8 位）来存放，字节的最高位为校验位，通常用“0”来填充，后 7 位（$b_6b_5b_4b_3b_2b_1b_0$）为编码值。

ASCII 码表中的常用字符包括：

（1）数字“0”~“9”。对应的 ASCII 码值为 0110000B ~ 0111001B，用十六进制数表示为 30H ~ 39H。

（2）字母。包括大、小写的英文字母各 26 个。字母“A”~“Z”的 ASCII 码值为 41H ~ 5AH，字母“a”~“z”的 ASCII 码值为 61H ~ 7AH。

（3）通用字符。如“+”、“-”、“;”、“,”、“/”和“,”等 32 个。

（4）控制字符。包括空格 SP（20H）、回车 CR（0DH）、换行 LF（0AH）等 34 个。

1.3.5 定点数的表示

在计算机中根据小数点的位置是否固定可以分为定点数和浮点数两种数据格式。在定点数中小数点的位置固定不变，通常把小数点固定在数位的最前面或末尾，所以定点数可以分为定点小数和定点整数两类。根据符号的有无，定点数又分为无符号数和有符号数两类。

1. 无符号数的表示

所谓无符号数是指没有符号的数，在寄存器中的每一位均可用来存放数值。当存放有符号数时，则需留出位置以存放符号。

2. 有符号数的表示

在计算机中，常采用机器数来表示数据，常用的有原码、反码、补码、移码等。

1）原码表示法

原码表示法是一种比较直观的表示方法，其符号位表示该数的符号，“+”号用“0”表示，“-”号用“1”表示，而数值部分用二进制数的绝对值来表示。

（1）定点小数的原码形式为 $x_0.x_1x_2\cdots x_n$，则原码的定义是

$$[x]_{原}=\begin{cases}x, & 0\leqslant x\leqslant 1\\ 1-x=1+|x|, & -1\leqslant x\leqslant 0\end{cases}$$

式中 x 是真值。

【例 1.4】 $x=+0.1101$，则 $[x]_{原}=0.1101$；$x=-0.1101$，则 $[x]_{原}=1-(-0.1101)=1+0.1101=1.1101$。

（2）定点整数的原码形式为 $x_0x_1x_2\cdots x_n$，则原码的定义是

$$x_{原}=\begin{cases}x, & 0\leqslant x\leqslant 2^n\\ 2^n-x=2^n+|x|, & -2^n\leqslant x\leqslant 0\end{cases}$$

其中 x 是真值，n 是整数位数。

【例 1.5】 $x=+1101$，则 $[x]_{原}=01101$；$x=-1101$，则 $[x]_{原}=2^4-(-1101)=2^4+1101=11101$。

原码表示法有以下两个特点：

① 零的表示有“+0”和“-0”之分，故有两种形式：

$[+0]_{原}=0.000\cdots0$

$[-0]_{原}=1.000\cdots0$

② 符号位 x_0 的取值由下式决定：

$$x_0=\begin{cases}0, & x>0\\ 1, & x<0\end{cases}$$

其中 x 是真值。

原码表示法简单明了，但用它进行加减运算时会带来许多麻烦。例如，当两数异号且要进行加法运算时，先要判断两数绝对值大小，然后将绝对值大的数减去绝对值小的数，结果的符号位以绝对值大的数为准。运算步骤既复杂又费时。如果能找到一个与负数等价的正数来代替该负数，就可把减法操作用加法代替。而机器数采用补码表示时，就能满足此要求。

2）补码表示法

由于计算机的运算受机器字长的限制，属于有模运算，所以，在计算机中可以使用补码进行计算。

（1）定点小数的补码形式为 $x_0.x_1x_2\cdots x_n$，则补码的定义是

$$[x]_{补}=\begin{cases}x, & 0\leqslant x\leqslant 1\\ 2+x, & -1\leqslant x<0\end{cases}$$

其中 x 是真值。

【例 1.6】 $x=0.1011$，则 $[x]_{补}=0.1011$；$x=-0.1011$，则 $[x]_{补}=10.0000+(-0.1011)=1.0101$。

$x=0$，则 $[+0.0000]_{补}=0.0000$；$[-0.0000]_{补}=10.0000+(-0.0000)=0.0000$。

显然，$[+0]_{补}=[-0]_{补}=0.000\cdots0$，即补码中的 0 只有一种表示形式。

对于小数，若 $x=-1$，则根据小数补码定义，有 $[x]_{补}=2+x=10.0000-1.0000=1.0000$。可见，-1 本不属于小数范围，但却有 $[-1]_{补}$ 存在，这是由于补码中的 0 只有一种表示形式，故它比