21st CENTURY
实用规划教材

21世纪全国高职高专
电子信息系列实用规划教材

微机原理及接口技术

主　编　王用伦　张开成
副主编　屈芳升　张小义

北京大学出版社
PEKING UNIVERSITY PRESS

21 世纪全国高职高专电子信息系列实用规划教材

微机原理及接口技术

主　编　王用伦　张开成
副主编　屈芳升　张小义
参　编　亢娟娜　陈昕志　李　纯

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 简 介

本书是根据高等职业技术院校的教学要求编写的。在介绍计算机基础知识的基础上，以 Intel 8086 微处理器为重点，以 Intel 80X86 微处理器组成的 PC 系列微机为背景，结合微机发展的新知识、新技术，系统地介绍了微机的组成原理、接口技术和典型应用。

全书共分 11 章，首先介绍了计算机基础知识，然后分别介绍了微型计算机结构、指令系统、汇编语言程序设计、存储器结构、微机常用总线系统、中断系统等微机组成原理。并在此理论基础上，对并行接口技术，串行接口技术，中断和定时计数、显示器、键盘接口技术进行了介绍。为了帮助读者理解和掌握所学知识，每章后都有思考与练习题。同时，还结合理论知识给出了实训，帮助读者提高实际动手能力。

本书可作为高职高专教育相关专业微机原理与接口技术课程的教材，也可以作为本科生学习微机原理与接口技术的参考书，还可以作为从事微机软、硬件工作的工程技术人员的参考书。

图书在版编目(CIP)数据

微机原理及接口技术/王用伦，张开成主编. —北京：北京大学出版社，2009.4

(21 世纪全国高职高专电子信息系列实用规划教材)

ISBN 978-7-301-12385-0

Ⅰ. 微… Ⅱ. ①王…②张… 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—高等学校：技术学校—教材 Ⅳ.TP36

中国版本图书馆 CIP 数据核字(2007)第 083347 号

书　　名：微机原理及接口技术

著作责任者：王用伦　张开成　主编

策 划 编 辑：赖　青

责 任 编 辑：魏红梅

标 准 书 号：ISBN 978-7-301-12385-0/TM · 0009

出 版 者：北京大学出版社

地　　址：北京市海淀区成府路 205 号　100871

网　　址：http://www.pup.cn　http://www.pup6.com

电　　话：邮购部 62752015　发行部 62750672　编辑部 62750667　出版部 62754962

电 子 邮 箱：pup_6@163.com

印 刷 者：世界知识印刷厂

发 行 者：北京大学出版社

经 销 者：新华书店

787 毫米×1092 毫米　16 开本　18.75 印张　433 千字

2009 年 4 月第 1 版　2009 年 4 月第 1 次印刷

定　　价：29.00 元

前　言

本书是根据高等职业技术院校的教学要求编写的，突出了“基本理论够用为度，应用为重点”的高职教育特点，结合微机发展的新技术、新知识，注重学生动手能力的培养。

本书以 Intel 8086 微处理器为重点，以 Intel 80X86 微处理器组成的 PC 系列微机为背景，结合微机发展的新知识、新技术，系统地介绍了微机的组成原理、接口技术和典型应用。全书共分为 11 章。第 1 章介绍了微型计算机基础知识；第 2 章以 Intel 8086 微处理器为重点，对 Intel 80X86 微处理器及微机结构进行了介绍；第 3 章介绍了 8086 指令系统；第 4 章介绍了汇编语言程序设计；第 5 章介绍了微机的存储器结构，其中包括一些存储新技术；第 6 章重点介绍了目前常用的总线技术；第 7 章介绍了微型计算机中断及定时/计数技术；第 8 章介绍了微型计算机并行接口技术；第 9 章介绍了微型计算机串行接口技术；第 10 章介绍了微型计算机显示器及键盘接口技术；第 11 章为综合实训，通过软件仿真和温度控制实训项目，提高学生对微机原理和接口技术的实际应用能力。本书的前半部分主要介绍微机的组成原理，后半部分主要介绍微机的接口技术。接口技术部分主要围绕各种可编程芯片及其应用进行介绍，既对硬件接口的具体连接进行讲解，又结合应用进行了软件编程。为了帮助读者理解和掌握所学知识，每章后面都有思考与练习题。

本书在编写过程中，力求既注重微机基本知识与典型应用的结合，又注重基本知识与最新知识的联系，尽量把最新技术介绍给读者。同时，还结合每章的理论知识给出了具体的实训项目，提出了具体要求，帮助学生提高实际动手能力。最后，通过综合实训帮助学生建立起微机应用的整体概念，提高分析解决实际问题的能力。

本书建议授课的总学时为 50，各章内容及学时分配见下表。

章　节	授课学时	章　节	授课学时
第 1 章　微型计算机系统概述	4	第 6 章　总线	4
第 2 章　微型计算机结构	6	第 7 章　微型计算机中断系统及定时/计数器应用	4
第 3 章　指令系统	6	第 8 章　微型计算机 I/O 接口技术及应用	4
第 4 章　汇编语言程序设计	10	第 9 章　微型计算机串行接口技术及应用	4
第 5 章　存储器结构	4	第 10 章　显示器及键盘接口技术	4

实训项目可结合各学校的实训条件和学习内容，另外安排学时进行。

本书第 1 章、第 6 章由重庆航天职业技术学院的张开成编写，第 2 章、第 11 章由重庆航天职业技术学院的王用伦编写，第 3 章由甘肃畜牧工程职业技术学院的张小义编写，第 4 章由甘肃畜牧工程职业技术学院的亢娟娜编写，第 5 章、第 9 章由河南职业技术学院的陈昕志编写，第 7 章、第 8 章由河南职业技术学院的屈芳升编写，第 10 章由重庆航天职业技术学院的李纯编写。本书由王用伦、张开成任主编，由王用伦负责统稿。

本书参考了有关微机原理与接口技术方面的大量书刊资料，引用了部分参考文献的内

容，在此谨向这些书刊资料的作者表示衷心的感谢！

由于编者的认识和专业水平有限，加之时间比较仓促，书中不足之处在所难免，敬请广大读者批评指正。

编　者

2009 年 1 月

目　录

第 1 章　微型计算机系统概述

【教学提示】 本章主要介绍微型计算机的发展史，计算机中的数和编码，微型计算机的基本概念、体系结构以及工作过程。

【教学目标】 了解计算机的发展史，理解微型计算机的系统结构、硬件组成及基本工作过程，掌握计算机中的数和编码方法。

【教学重点】 微型计算机的系统结构、硬件组成及基本工作过程，计算机中的数和编码方法。

1.1　微型计算机的发展史

以半导体集成电路为中心的微电子技术的进步，使计算机向着微型化、高性能、低成本的方向迅猛发展。自 1946 年第一台电子计算机问世以来，计算机已经历了从电子管、晶体管、集成电路到大规模和超大规模集成电路的发展演变。计算机通常按其体积、性能和价格分为巨型机、大型机、中型机、小型机和微型机 5 类。由于微型机具有体积小、重量轻、功耗低、价格便宜、结构灵活等特点，因而得到了广泛的普及和应用。

微型计算机的发展史也就是微处理器的发展史，自从 20 世纪 70 年代初第一个微处理器诞生以来，微处理器的性能和集成度几乎每两年就提高一倍，而价格却降低了一个数量级。

1971年10月，美国Intel公司首先推出4004微处理器，采用PMOS技术，在4.2mm×3.2mm的硅片上集成了 2 250 个晶体管，可进行 4 位二进制的并行处理。这是实现 4 位并行运算的单片处理器，所有元件都集成在一片 MOS 大规模集成电路芯片上。以 4004 微处理器为基础，再配以相应的 RAM、ROM 和 I/O 接口芯片等，就构成了 MCS-4 微型计算机。

从 20 世纪 70 年代初至今仅 30 多年的时间，微处理器的发展就经历了以下几个时期。

第一代从 1971 年开始，是 4 位微处理器和低档 8 位微处理器的时期。典型产品有 Intel 的 4004(4 位)、4040(8 位)和 8008(8 位)，其集成度为 2 000 个晶体管/片，采用 PMOS 工艺，10μm 光刻技术，时钟频率仅 1MHz，平均执行指令时间约为 20μs。

第二代包括 1974—1978 年 Intel 公司推出的 8080/8085、Zilog 公司推出的 Z80、Motorola 公司的 MC6800/6802 等。这一代高档 8 位微处理器的特点是采用 NMOS 工艺，集成度达到 8 400 个晶体管/片以上，时钟频率为 2～4MHz，平均执行指令时间约为 1～2μs。这时的微处理器设计和生产技术已相当成熟，配套的各种器件也很齐备。以后的微处理器在提高集成度、提高功能和速度、增加外围电路的功能和种类上发展很快。

第三代以 16 位微处理器的出现为代表，时间为 1978 年。典型产品有 Intel 的 8086、

Zilog 的 Z8000 和 Motorola 的 MC68000 等。它们采用了 HMOS 工艺，集成度为 20 000～60 000 个晶体管/片，时钟频率为 4～8MHz，平均执行指令时间为 0.5μs。

第四代(1980—1992 年)以 32 位微型机为代表。典型产品有 Intel 的 80386/80486、Motorola 的 MC68020、Zilog 的 Z80000 等。这一代微处理器的时钟频率为 16～100MHz，集成度为 10^5 个晶体管/片。

1993 年 Intel 公司推出的 Pentium 微处理器，宣布了第五代微处理器的诞生。这种 Pentium 微处理器芯片采用了全新的体系结构，芯片的集成度高达 5.0×10^6～9.3×10^6 个晶体管/片，时钟频率达 150～300MHz。

1995 年 Pentium II 问世，1999 年 Pentium III诞生。目前市场上的主流产品已由 Pentium 4 取代了 Pentium III。随着集成电路工艺和计算机科学技术的迅猛发展，更高性能的微处理器将不断推出，微型计算机的发展速度不可估量，其应用也越来越广泛。

1.2　计算机中的数和编码

1.2.1　计算机中的数制

1. 常用进位计数制

1) 十进制

十进制数是人们最熟悉的，但机器实现起来困难。

2) 二进制

由于二进制数每位只需两个状态“0”或“1”，机器实现容易，因而二进制数是数字系统唯一认识的代码，但其书写太长。

3) 八进制

2^3=8，因而 3 位二进制数可用一位八进制数表示。

4) 十六进制

2^4=16，因而 4 位二进制数可用一位十六进制数表示。

在计算机应用系统中，二进制主要用于机器内部的数据处理，八进制和十六进制主要用于书写程序，十进制主要用于输出运算的最终结果。

为了使书写的进位计数制数便于区分，十进制数用 D 表示或省略，二进制数用 B 表示，八进制数用 Q 表示，十六进制数用 H 表示。

2. 数制转换

1) 非十进制数转换成十进制数

不同数制之间的转换方法有若干种，把非十进制数转换成十进制数采用按权展开相加法。具体步骤是：首先把非十进制数写成按权展开的多项式，然后按十进制数的计数规则求和。

2) 十进制数转换成二进制数

既有整数部分又有小数部分的十进制数转换成二进制数，首先要把整数部分和小数部分分开转换，再把两者的转换结果相加。具体方法介绍如下。

(1) 整数转换。

方法：连续除以 2 取余数，直到商为 0，并按照和运算过程相反的顺序把各个余数排列起来，即为二进制整数。

(2) 小数转换。

方法：连续乘以 2 取整数，并按照和运算过程相同的顺序把各个整数排列起来，即为二进制小数。

3) 二进制数转换成八进制数或十六进制数

二进制数转换成八进制数(或十六进制数)时，其整数部分和小数部分可以同时进行转换。方法是：以二进制数的小数点为起点，分别向左、向右将每 3 位(或 4 位)分成一组。对于小数部分，最低位一组不足 3 位(或 4 位)时，必须在有效位右边补 0，使其足位。然后，把每一组二进制数转换成八进制数(或十六进制数)并保持原顺序。对于整数部分，最高位一组不足位时，可在有效位的左边补 0，也可以不补。

4) 八进制数或十六进制数转换成二进制数

八进制数(或十六进制数)转换成二进制数时，只需把八进制数(或十六进制数)的每一位数码分别转换成 3 位(或 4 位)的二进制数，并保持原排序即可。整数最高位一组左边的 0 及小数最低位一组右边的 0 可以省略。

1.2.2　符号数的表示

1. 机器数与真值

二进制数与十进制数一样有正有负。在计算机中，常把数的符号和数值部分一起编码后表示带符号数。常用的带符号数编码方法有原码、反码和补码表示法。这几种表示法都将数的符号数码化，通常正号用“0”表示，负号用“1”表示。为了区分一般书写时表示的数和机器中编码表示的数，称前者为真值，后者为机器数，即数值连同符号数码“0”或“1”一起作为一个数称为机器数，而它的数值连同符号“＋”或“-”称为机器数的真值。

为了表示方便，常把 8 位二进制数称为字节，把 16 位二进制数称为字，把 32 位二进制数称为双字。机器数应用字节、字或双字表示，因此只有 8 位、16 位或 32 位机器数的最高位才是符号位。

1) 原码

如上所述，数值部分用该数的绝对值，符号部分用“0”表示正数，用“1”表示负数，这样表示数的方法称为带符号数的原码表示法，所表示的数称原码。

$$y1=+127=+1111111B \qquad [y1]_{原}=01111111B$$

$$y2=-127=-1111111B \qquad [y2]_{原}=11111111B$$

其中最高位为符号，后面 7 位是数值。用原码表示时，+127 和-127 的数值部分相同而符号位相反。

8 位整数原码表示的数值范围为 0FFH～7FH，即-127～+127。原码数 00H 和 80H 的数值部分相同，符号位相反，它们分别为+0 和-0。16 位整数原码表示的数值范围为 0FFFFH～7FFFH，即-32767～+32767。原码数 0000H 和 8000H 的数值部分相同，符号位相反，它们分别为+0 和-0。

原码表示简单易懂，而且与真值的转换方便。但若是两个异号数相加，或两个同号数

相减，就要做减法。为简化计算机的结构，把减法运算转换为加法运算，因而引入了反码和补码。

2) 反码

正数的反码与原码相同；负数的反码为它的原码的数值部分按位取反，而符号位不变，即仍为“1”。

$$y1=+127=+1111111B \quad [y1]_{反}=01111111B$$
$$y2=-127=-1111111B \quad [y2]_{反}=10000000B$$

3) 补码

正数的补码与原码相同；负数的补码为它的原码的数值部分按位取反，最末位再加1，而符号位不变，即仍为“1”。

$$y1=+127=+1111111B \quad [y1]_{补}=01111111B$$
$$y2=-127=-1111111B \quad [y2]_{补}=10000001B$$

8位补码数表示的数值范围为80H～7FH，即-128～+127。16位补码表示的数值范围为8000H～7FFFH，即-32768～+32767。

2. 原码、反码、补码和真值的相互转换关系

前面已经讨论了符号数的真值表示和在机器中的表示，下面总结机器数和真值的相互转换关系。

正数的原码、反码、补码相同，即用正号“0”加上绝对值的数值位表示。

负数的原码、反码、补码和真值的相互转换关系，如图1.1所示。

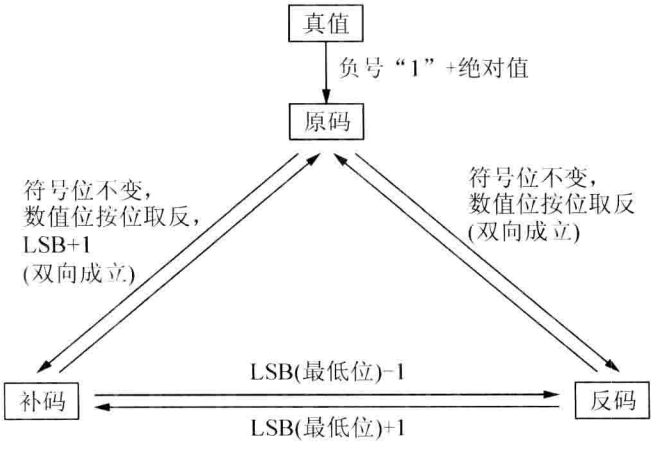

图1.1　负数的原码、反码、补码和真值的相互转换关系

下面以负数情况为例来进行分析，以说明符号数的机器表示和真值的相互转换关系。

1) 已知原码，求补码

【例1.1】 已知某数X的原码为10110100B，试求其补码。

【解】 由$[X]_{原}$=10110100B，知X为负数。求其补码表示时，符号位不变，数值部分按位取反，再在末位加1。

```
  1 0 1 1 0 1 0 0     原码
  ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
  1 1 0 0 1 0 1 1     符号位不变，数值位取反
+               1     +1
-----------------
  1 1 0 0 1 1 0 0     补码
```

∴ $[X]_{补}$=11001100B

2) 已知补码，求原码

【例 1.2】 已知某数 X 的补码为 11101110B，试求其原码。

【解】 由$[X]_{补}$=11101110B，知 X 为负数。求其原码表示时，符号位不变，数值部分按位求反后，再在末位加 1。

```
  1 1 1 0 1 1 1 0     补码
  ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
  1 0 0 1 0 0 0 1     符号位不变，数值位取反
+               1     +1
-----------------
  1 0 0 1 0 0 1 0     原码
```

∴ $[X]_{原码}$=10010010B

3. 求补

已知$[X]_{补}$，求$[-X]_{补}$的过程称为求补。所谓求补，就是将[X]补的所有位(包括符号位)一起逐位取反，然后在末位加 1，即可得到-X 的补码，亦即[-X]补。不管 X 是正数还是负数，都应按此方法操作。

【例 1.3】 试求+97、-97 的补码。

【解】 +97=1100001，于是$[+97]_{补}$=01100001B

求$[-97]_{补}$的方法是：

```
  0 1 1 0 0 0 0 1     [+97]补
  ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
  1 0 0 1 1 1 1 0     逐位取反
+               1     +1
-----------------
  1 0 0 1 1 1 1 1     [-97]补
```

∴ $[+97]_{补}$=01100001B，$[-97]_{补}$=10011111B

4. 已知补码，求对应的十进制数

【例 1.4】 已知某数 X 的补码为 10101011B，试求其对应的十进制数。

【解】 由$[X]_{补}$=10101011B，知 X 为负数，故符号位不变，数值部分按“求反加 1”法得到$[X]_{原}$。

$[X]_{原}$=11010101B，X=-1010101B，X=-85

1.2.3 二进制数的运算

1. 二进制数的加减运算

计算机把机器数均当作无符号数进行运算，即符号位也参与运算。运算的结果要根据

运算结果的符号，运算有无进位(借位)和溢出等来判别。计算机中设置有这些标志位，标志位的值由运算结果自动设定。

1) 无符号数的运算

无符号数实际上是指参加运算的数均为正数，且全部数位都用于表示数值。n 位无符号二进制数能表示的数值范围为 0～(2^n-1)。

两个无符号数相加，由于两个加数均为正数，因此其和也是正数。当和超过其位数所允许表示的数值范围时，就向更高位进位。例如：

127+150=7FH+96H

```
  0 1 1 1 1 1 1 1
+ 1 0 0 1 0 1 1 0
-----------------
1 0 0 0 1 0 1 0 1=115H=256+16+5=277
↑
└── 进位
```

两个无符号数相减，被减数大于或等于减数，无借位，结果为正；被减数小于减数，有借位，结果为负。例如：

128-10=80H-0AH

```
  1 0 0 0 0 0 0 0
- 0 0 0 0 1 0 1 0
-----------------
  0 1 1 1 0 1 1 0=76H=112+6=118
```

反过来相减，即 10-128，运算过程如下：

```
  0 0 0 0 1 0 1 0
- 1 0 0 0 0 0 0 0
-----------------
1 1 0 0 0 1 0 1 0=-01110110=-118
↑
└── 借位
```

由此可见，对无符号数进行减法运算，其结果的符号用借位标志位来判别：若 CF=0，则无借位，结果一定为正；若 CF=1，则有借位，结果一定为负，对 8 位数值位求补便可求得它的绝对值。

2) 符号数的加法运算和溢出判断

(1) 补码的加减法运算。

符号数在引入补码表示法以后，补码的减法运算可以转换为加法运算来进行。其运算法则为：

$$[x \pm y]_{补}=[x]_{补}+[\pm y]_{补}$$

补码加法运算规则如下。

① 参加运算的数均为补码。

② 符号位和数值位一起参加运算，符号位向前的进位自然丢掉，若不发生溢出，留下的结果一定是正确的。

③ 运算结果为补码。

④ 若运算结果为负数的补码，若未发生溢出，应转换为原码，这样才能知道结果的真值。

【例 1.5】 用 8 位补码完成下列计算。

(1) 63−127　　(2) −67−15

【解】

(1) $[63]_{补}$：　0 0 1 1 1 1 1 1

$[-127]_{补}$：+ 1 0 0 0 0 0 0 1

[补]：　1 1 0 0 0 0 0 0

[原]：　1 1 0 0 0 0 0 0

∴ 63−127=−64

$[-67]_{补}$：　1 0 1 1 1 1 0 1

$[-15]_{补}$：+ 1 1 1 1 0 0 0 1

[1] 1 0 1 0 1 1 1 0[补]

1 1 0 1 0 0 1 0[原]

∴ −65−15=−82

(2) 补码加减法运算的溢出判断。

两个补码加数相加时，在什么情况下才有可能发生溢出呢？很显然，只有当它们都是正数或负数，补码相加时才有可能发生溢出。

【例 1.6】 两个带符号数 01000001B(+65)和 01000011B(+67)相加如下。

```
  0 1 0 0 0 0 0 1
+ 0 1 0 0 0 0 1 1
-----------------
  1 0 0 0 0 1 0 0
```

例 1.6 中是两个正数相加，但相加结果却是一个负数，显然这个结果是错误的，出现这种情况的原因就在于这两个加数相加的结果超过了 8 位二进制带符号数补码所能表示的范围(−128～+127)。

【例 1.7】 再来看两个负数补码 10001000B(−120)和 11101110B(−18)的相加情况。

```
  1 0 0 0 1 0 0 0
+ 1 1 1 0 1 1 1 0
-----------------
 [1]0 1 1 1 0 1 1 0
```

由于规定用 8 位二进制数来表示带符号数，故忽略作为进位位的第 9 位。按 8 位二进制数来解释这两个符号数的相加，其结果为一个正数，很明显，结果是错误的。错误的原因如上所述。

以上两种情况叫做补码运算的溢出。当两个同符号的数据相加时，如果相加的结果超过了微处理器所能表示的数值范围，就将发生溢出，其结果就是错误的。因此，在微处理机中设有专门的电路用以判断运算结果是否产生溢出，并以某种标志告诉人们这次运算的结果是否存在溢出。只要溢出没有发生，运算的结果总是正确的。这个在 8086 微处理器中是用 OF 来标志的。若运算结果发生溢出，则置 OF 为 1；否则 OF 为 0。

在机器中如何判溢出呢？如何置 OF 标志位呢？

设符号位向进位位的进位为 C_Y，数值部分向符号位的进位为 C_S，则有无溢出的判别式如下。

$$OF=C_Y \oplus C_S$$

式中：OF=1 表示有溢出；OF=0 表示无溢出。

【例 1.8】 再来看 105+50、-105-50 和-50-5 的运算结果有无溢出。

(1)　01101001　　(2)　10010111　　(3)　11001110
　　+ 00110010　　　+ 11001110　　　+ 11111011
　　10011011　　　101100101　　　111001001
　　C_Y=0，C_S=1　　C_Y=1，C_S=0　　C_Y=1，C_S=1
　　OF=0⊕1=1，有溢出。　　OF=1⊕0=1，有溢出。　　OF=1⊕1=0，无溢出。

2. 二进制数的逻辑运算

1) 逻辑非

逻辑非亦称“求反”。对二进制数进行逻辑非运算，就是按位求它的反。

2) 逻辑与

对两个二进制数进行逻辑与，就是按位求它们的逻辑积。逻辑与又称逻辑“乘”，常用记号“∧”或“.”来表示。

3) 逻辑加

对两个二进制数进行逻辑加，就是按位求它们的“或”，所以逻辑加又称“逻辑或”，常用记号“∨”或“+”来表示。

4) 逻辑异或

对两个二进制数进行逻辑异或，就是按位求它们的模 2 和，所以逻辑异或又称“按位加”，常用“⊕”来表示。

注意：按位加与普通整数加法的区别是它仅按位相加，不产生进位。

1.2.4　二进制编码

计算机中除了处理数值信息外，还要处理大量的字符数据，如字母、符号、汉字、图像、语音等。在计算机中常用的字符编码有 BCD(Binary Coded Decimal)码、ASCII 码、格雷码、奇偶校验码等，这些编码在机器内部都是用二进制表示的，所以统称为二进制编码。

1. BCD 码

二进制编码的十进制码(BCD 码)有压缩和非压缩两种存储形式。压缩的 BCD 码用半个字节存放一位十进制数，即一个字节存放两位十进制数；而非压缩的 BCD 码则以一个字节存放一位十进制数。BCD 码在指令中是常用的一种编码。

所谓 BCD 码就是由 4 位二进制数表示一位十进制数，十进制数只有 10 个基数符号，而 4 位二进制数有 16 种取值组合，这样就出现了多种编码方案。下面仅介绍 8421 码、余 3 码和格雷码。这 3 种编码与十进制数的对应关系见表 1-1。

表 1-1　BCD 码与十进制数的对应关系

十进制数据	8421 码	余 3 码	格　雷　码
0	0000	0011	0000
1	0001	0100	0001
2	0010	0101	0011
3	0011	0110	0010
4	0100	0111	0110
5	0101	1000	0111
6	0110	1001	0101
7	0111	1010	0100
8	1000	1011	1100
9	1001	1100	1101

1) 8421 码

8421 码是一种最简单的二进制自然编码，它以 4 位二进制数的前 10 个代码分别对应十进制数的 10 个数码，1010～1111 为无效编码。这种编码按权求和，和就是对应的十进制数。这就是说，编码中的每位仍保留着二进制数所具有的位权，而且 4 位代码从左向右的位权依次为 8、4、2、1，8421 码就是由此而命名的。

如十进制数 72 的压缩 BCD 码用 8421 码可表示为 01110010；十进制数 54 对应的编码为 01010100。

2) 余 3 码

余 3 码是无权码，它具有良好的代码校验性。这种编码转换成十进制数后，每个代码的值比相应的十进制数多 3。例如，表 1-1 中的十进制数 4，它对应的二进制代码是 0111=7；十进制数 9 对应的二进制代码是 1100=12，它们都比相应的十进制数大 3。因此，称这种编码为余 3 码。

3) 格雷码

格雷码也是一种无权的 BCD 编码形式，其特点是相邻的两位代码仅有一个码元变化。这种编码的抗干扰能力强，常用于计算机控制机床的角编码器、轴位编码等。

2. ASCII 码

对各个字母和符号编制的代码叫字符代码。字符代码的种类繁多，目前在计算机和数字通信系统中被广泛采用的是 ASCII 码(American Standard Code for Information Interchange，美国信息交换标准代码)，其编码表见表 1-2。

在表 1-2 中读码时，先读列码 $B_7B_6B_5$，再读行码 $B_4B_3B_2B_1$ 即为某字符的 7 位 ASCII 码。例如字母 A 的列码是 100，行码是 0001，因此 A 的 7 位 ASCII 码是 1000001，即 41H。注意表中最左边一列的 A，B，…，F 是十六进制数的 6 个数码。

目前，ASCII 码有 7 位和 8 位两种字符编码形式。常用的是 7 位 ASCII 码，它包括 26 个大写和小写英文字母、10 个数字以及一些专用字符。7 位编码的 ASCII 码，实际也采用 8 位二进制(一个字节)，但最高位置 0 或用作校验，故最多可表示 128 个字符(即 2^7=128)。