高等学校教材

微型计算机原理

舒贞权 孙 漪 陈景琦 编

WXJSJ
WXJSJ
WXJSJ

西安交通大学出版社

简　介

本书从应用的角度出发，用软硬结合的方法，以Z80（8位微处理器）和Intel 8086/8088(16位微处理器)为例，系统的介绍了微型计算机的原理与应用。全书共九章。前八章讨论以8位微处理器为基础的微型计算机基本原理、硬件和软件，包括微型计算机的基础知识；Z80 CPU的结构、引脚和时序；详细地分析了Z80的指令系统；以较多的实例介绍汇编语言源程序的设计方法；存贮器芯片及其与CPU的连接；输入/输出基本原理及中断技术；从实际出发介绍了几种常用的输入/输出接口电路芯片和A/D、D/A转换器的连接方法；最后简要地讨论了微型计算机系统组成和监控程序及操作系统的概念。第九章较详细地讨论了目前最为流行的16位微处理器Intel 8086/8088的结构、指令系统和汇编语言。

本书系高等学校工科电子类专业“微型计算机原理”课程的教学用书，可作为其它有关专业的教材，也适用于广大科技人员作为培训教材和自学参考书。

(陕)新登字007号

微型计算机原理

舒贞权　孙　漪　陈景琦　编

责任编辑　陆　薇　赵晋平

*

西安交通大学出版社出版

邮政编码:710049

空军西安印刷厂印装

陕西省新华书店经销

*

开本787×1092 1/16　印张:22.125　字数:543千字

1989年1月第1版　1994年8月第5次印刷

印数: 35551—40550

ISBN7-5605-0128-1/TP·14　定价: 16.00元

前　言

本教材是根据高等学校工科电子类专业《无线电技术与信息系统》教材编审委员会制订的《微型计算机原理》课程教材编写纲要编写的。

无线电技术专业与电子工程专业（包括通信、雷达、电子对抗、电视、电子仪器与测量等）的学生学习本课程的目的，是为了解决微型计算机，包括微处理器及其支持电路在电子设备的自动化、智能化中的应用问题。因此，本教材在编写时着重从应用的角度出发进行分析，并充分注意到应用中的一些共性问题。在硬件方面着重介绍Z80和Intel 8086/8088微处理器结构、接口技术和接口电路；在软件方面着重介绍了应用微型计算机的基本工具—汇编语言（Z80汇编语言和Intel 8086/8088汇编语言）程序设计。

本教材以1984年6月经评审推荐的讲义为基础，在长期教学实践中，经两次修改后写成的，在修改过程中，注意到微机应用的发展趋势，对原稿作了较大幅度的删节，并增加了16位微处理器内容的比重。

本教材共分九章，其中第一章的第三、四节和第五章由陈景琦执笔，第三章、第四章及第九章的第二节和第三节由孙漪执笔，第一章的第一、二、五节和第二、六、七、八章、第九章的第一节由舒贞权执笔，附录由孙漪选编。

本书经仪表与测量编审组评审、推荐由西安交通大学出版社出版。天津大学王慧云副教授进行了全面而详细的审核，提出了许多宝贵意见，在此表示衷心感谢。

本教材除适用于无线电技术专业外，也适宜于其它电子工程类专业，也可作为科技人员的自学教材和参考书。

由于作者的水平有限，书中难免存在不少缺点和错误，恳请批评指正。

编　者

1987年9月

目　　录

第一章　微型计算机的基础知识

§1-1　微型计算机发展过程简介

微型计算机是电子计算机技术和大规模集成(LSI)电路技术的结晶，它的出现和发展是和LSI电路技术的迅速发展分不开的。

第一台数字电子计算机"ENIAC"是1946年问世的，它采用18000只电子管，机房占地面积为150平方米，机器重约30t，耗电近100kw，运算速度为每秒5000次。经过30多年的努力，计算机至今已经历了三代更新，目前发展到以全面采用大规模集成电路为主要特征的第四代计算机，运算速度为每秒数亿次的巨型机已投入运行，并已开始研制第五代超大规模集成电路计算机，运算速度将达到每秒百亿次。

从1946年开始的第一代电子计算机，使用的逻辑元件是电子管，主存贮器使用磁鼓，软件主要使用机器语言和汇编语言。由于设备庞大，资金昂贵，使得第一代电子计算机主要应用于少数军事部门。以现在的眼光来看，显然那时的计算机是很原始的，然而，它却奠定了计算机发展的技术基础，如数字编码，自动运算方式和程序存贮等。

从1957年到1964年为第二代。这一代的主要特点是：逻辑元件采用晶体管，主存贮器采用磁芯存贮器，辅助存贮器使用磁盘，软件已开始使用高级程序设计语言和操作系统，同第一代计算机相比，体积小，耗电少，运算速度与可靠性提高，价格大幅度下降。电子计算机开始用于民用事业（主要是数据处理）和工业生产部门（过程控制）。

60年代初，半导体制造厂创造了一种全新的工艺——集成电路工艺。集成电路技术和计算机技术相结合，产生了第三代电子计算机。从1964年美国IBM公司的IBM 360系列计算机问世到1971年为第三代，其特点是：逻辑元件采用集成电路；机种多样化，系列化；外部设备不断增加，品种繁多；高级程序设计语言发展很快，操作系统进一步发展，应用领域扩大。在此期间，在发展大型机的同时，小型机也飞速发展起来。小型机性能好，造价低，体积小，方便灵活，这些特点为计算机进入实验室和工厂的工业生产线创造了有利条件。小型机的发展，为以后微型计算机的发展奠定了基础。

到了70年代，半导体工艺技术的发展突飞猛进。70年代初开始能够大量地生产在一个硅片上集成数千个晶体管的大规模集成电路。

LSI电路技术的迅速发展，使电子计算机进入第四代。这时的计算机主存贮器采用半导体存贮器，使得容量和速度提高了几个数量级。这一代计算机的主要特点是向巨型机、微型机、多处理机系统和计算机网络等方向发展。巨型机是高速度、大容量的计算机，如HITAC-810系统，速度达每秒6.3亿次，存贮容量为256兆字节。

第四代计算机发展的另一个方面是微处理器与微型计算机的诞生和迅速发展。

1971年，从事生产半导体存贮器的英特尔(Intel)公司研制成功世界上第一个4位微处

理器Intel 4004，它在一块 $0.297\times0.404cm^2$ 的硅片上集成了 2250 个晶体管，一次能处理四位二进制数字组成的字组作为计算器芯片。1972年英特尔公司生产出功能比 4004 更强的位 8 位微处理器 Intel 8008，这是第一个通用的 8 位微处理器。此后，生产微处理器和微型计算机的厂家剧增，微处理器的集成度每两年增加一倍，性能增长一个数量级。从 1971 年以来，微处理器已经经历了四代的变化。

第一代（1971—1973年）是 4—8 位低档微处理器，以 4004 和 8008 为代表，其基本特征是采用 PMOS 工艺，有 4—8 位并行处理能力，引脚数为 16—24 条，系统结构还没有超出计算器的范围。

第二代（1973—1976年）是 8 位高档微处理器。典型的产品是英特尔公司的 Intel 8080，莫托洛拉(Motorola)公司的 MC 6800 和泽洛格(Zilog)公司的 Z 80。这一代微处理器的基本特征是采用 NMOS 工艺或其它工艺，具有 8 位并行处理能力，有 40—42 条引脚，并具有典型的计算机结构形式（多种寻址、多级中断）。这些微处理器至今仍是应用最广泛的机型。

第三代（1977—1980年）是 16 位微处理器，以 Intel 8086，Z 8000 和 MC68000 为代表。与第二代相比，它们的字长更长，速度更快，功能更强，由其组装的微型计算机已经可以与高档的小型计算机相媲美。第三代微处理器的基本特征是采用速度与集成度更高的 HMOS 或其它工艺，有16 位并行处理能力，引脚数为 40—64 条，具有小型机的体系结构。

第四代（1980年以后）是 32 位微处理器，如英特尔公司的 iAPX432，这是一种功能更强的微处理器，可直接寻址的地址空间达 2^{40}，采用微程序技术，应用高速缓存以提高系统的效率。

如今微型机的功能已相当于传统的中型机，小型机与微型机之间在性能方面不再有明显的差别。运用微型机构成的大型系统已在性能价格比和应用方面远远地超过了以往几代大型机。微型计算机的蓬勃发展改变了计算机市场的产品结构。长期以来，大型通用机垄断市场的局面发生了变化，计算机开始进入个人化时代，大量的个人计算机涌入市场。有人说，70 年代是微处理器的时代，80年代则是微型机系统的时代。

为什么微处理器会得到如此迅速的发展呢？

首先，微处理器同已往各代计算机相比有突出的优点：价格便宜、体积小、重量轻、耗电量小、可靠性高，对使用环境要求低，通用性和灵活性好。价格便宜，是微型计算机最突出的优点，一台微型计算机同一台性能相当的小型机相比价格要低一个数量级。

其次，LSI 电路技术的不断进步，使微处理器生产有了坚实的基础。1977 年，世界上生产了 830 万个微处理器，1978 年增至 1800 万个以上。近年来，微处理器的生产一直保持增长的趋势，现在 64K 位的 MOS 存贮器已普遍使用，16位微处理器也开始普及。

§1-2 微处理器与微型计算机的定义

在学习本课程时，时常要遇到以下一些术语：微处理器、微型计算机、微型计算机系统。目前的一些书刊对这些术语的定义往往不很一致。

所谓微处理器（Microprocessor，简称 MP 或 μP）。是一个大规模集成电路器件，在器件中集成了多个寄存器、算术逻辑运算部件、控制逻辑部件和数据通道等，因而这个LSI 或 VLSI（超大规模）电路器件具有传统的电子计算机中的中央处理单元(CPU)的功能。人们称

这种具有CPU功能的LSI或VLSI电路器件为微处理器。应该指出：首先，某些微处理器并非把完成CPU功能的各部件集成在一块硅片上，例如Intel 8080微处理器，必须配上“时钟电路”(8224)和“系统控制器”(8228)才具有CPU的功能；其次，微处理器并不是一个完整的计算机，它只完成电子计算机中控制器部分和运算部分的功能。

微型计算机(Microcomputer，简称MC或μC)是具有独立运行功能的计算机，除了微处理器之外，它还应包括存贮器(称为主存贮器或内存贮器)、输入/输出接口电路(I/O电路芯片)及系统总线等。如图1-1所示。因而，所谓微型计算机，就是由微处理器、存贮器、I/O接口电路、系统总线及其它支持逻辑(如译码器、驱动器等)所组成的具有完整运算功能的电子计算机。

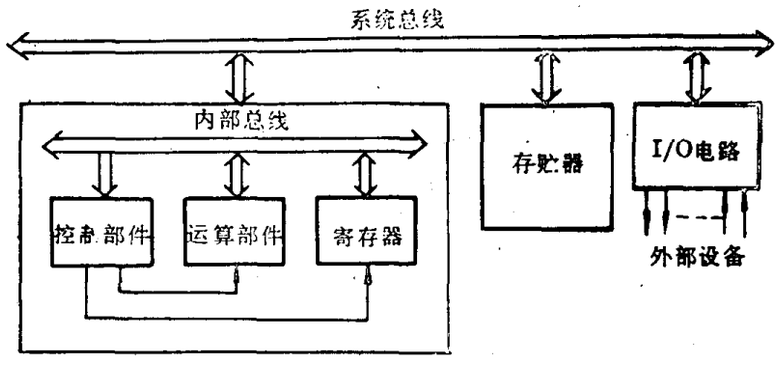

图 1-1 微型计算机结构框图

有些书刊把MP称为微处理机，或用微处理机泛指微处理器或微型计算机，这都容易造成概念上的混淆。此外，本教材将不区分Microprocessor(MP)与Microprocessor Unit(MPU)，它们都是指具有CPU功能的集成电路器件。

上面是从“硬件”角度来定义微处理器与微型计算机的。但只有硬件部分计算机是不能工作的，任何计算机的使用都离不开“软件”。微型计算机配上软件，外部设备和电源系统就构成微型计算机系统(Microcomputer System，简称MCS)。MP、MC和MCS之间的关系如图1-2所示。

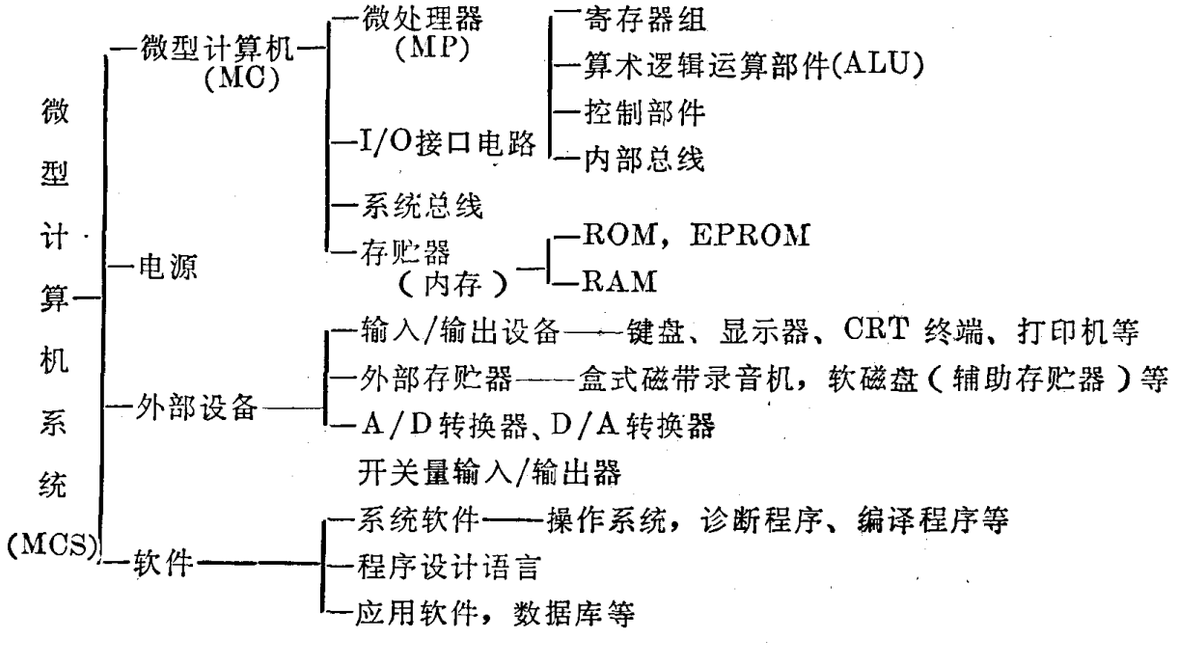

图 1-2 微处理器、微型计算机和微型计算机系统

§1-3 进位计数制及不同计数制之间的转换

一、进位计数制

进位计数制是将数划分为不同的数位，按位进行累计，累计到一定数量之后，又从零开始，同时向高位进一，每位数使用同样的一些数码符号。但是由于划分了数位的等级，同一数码在不同的数位上所表征的数值是不同的，低位数码数值小，高位数码数值大。进位计数法的优点是使用较少的数码就能表示较大的数。

1. 十进制数

十进制数有0、1、2、3、4、5、6、7、8、9十个数码，它的计数规则是“逢十进一”。数码处于不同位置，所代表的数值大小不同。例如66.6可表示为：

$$66.6=6\times10^{1}+6\times10^{0}+6\times10^{-1}$$

任何一个十进制数N都可以写成以“10”为底的幂的和式：

$$N=K_n(10)^n+K_{n-1}(10)^{n-1}+\cdots+K_1(10)^1+K_0(10)^0+K_{-1}(10)^{-1}$$
$$+\cdots+K_m(10)^{-m}=\sum_{i=n}^{-m}K_i(10)^i$$

式中m、n为正整数，n为小数点左面位数减1，m为小数点右面位数，K_i可取0、1、2、3、4、5、6、7、8、9十个数码中的任意一个，它由N决定。括号内的“10”称为计数制的基数。所谓基数，就是在计数制中所用到数码的个数。

2. 二进制数

二进制数只有0和1两个数码，它的计数规则是“逢二进一”。同样，数码的位置不同，所代表的数值大小也不同。例如111.1可表示为：

$$111.1=1\times2^{2}+1\times2^{1}+1\times2^{0}+1\times2^{-1}$$

任意二进制数B都可以写成以2为底的幂的和式：

$$B=B_n(2)^n+B_{n-1}(2)^{n-1}+\cdots+B_1(2)^1+B_0(2)^0+B_{-1}(2)^{-1}+\cdots+B_{-m}(2)^{-m}$$
$$=\sum_{i=n}^{-m}B_i(2)^i$$

式中m、n为正整数，n为小数点左面的位数减1，m为小数点右面的位数。B_i是0、1两个数码中的任意一个，它由B决定。括号内的2称为二进位计数制的基数。

3. 八进制和十六进制

比较十进制数和二进制数的数学表达式，我们可以得出任意Q进制数R的数学表达式：

$$R=R_n(Q)^n+R_{n-1}(Q)^{n-1}+\cdots+R_1(Q)^1+R_0(Q)^0+R_{-1}(Q)^{-1}+\cdots$$
$$+R_{-m}(Q)^{-m}=\sum_{i=n}^{-m}R_i(Q)^i$$

式中m、n为正整数，n为小数点左面的位数减1，m为小数点右面的位数，Q是进位制的基数，R_i可以是Q个数码中的任意一个，它由R决定。

显然当$Q=8$时，它就是八进制数，R_i可以是0、1、2、3、4、5、6、7，八个数码中的任意一个，它的基数是8，计数规则是“逢八进一”。对于任意一个八进制数R均可写成以8为底的幂的和式：

$$(R)_8=R_n(8)^n+R_{n-1}(8)^{n-1}+\cdots+R_1(8)^1+R_0(8)^0+R_{-1}(8)^{-1}+\cdots$$
$$+R_{-m}(8)^{-m}=\sum_{i=n}^{-m}R_i(8)^i$$

当 $Q=16$ 时它就是十六进制数。R_i可以是 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F、十六个数码中的任一个，其中A、B、C、D、E、F分别表示 10、11、12、13、14、15，它的基数是 16，计数规则是“逢十六进一”。对任意一个十六进制数R，均可写成以 16 为底的幂的和式：

$$(R)_{16}=R_n(16)^n+R_{n-1}(16)^{n-1}+\cdots+R_1(16)^1+R_0(16)^0+R_{-1}(16)^{-1}+\cdots$$
$$+R_{-m}(16)^{-m}=\sum_{i=n}^{-m}R_i(16)^i$$

二、不同计数制之间的转换

1. 二进制与十进制之间的相互转换

(1) 二进制转换成十进制

二进制转换为十进制用加权法。所谓加权法，就是将二进制数写成权展开式，然后将数码为 1 的那些位的权值相加，就得到相应的十进制数。

例 $(1101.11)_2=1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}=(13.75)_{10}$

(2) 十进制转换成二进制

十进制数转换成二进制数时，分整数和小数两种情况，两者的方法不同。

对于十进制整数，可用除 2 取余法。将十进制数除以“2”，得到商数和余数，再将商数除以“2”，又得到一个新的商数和余数，如此继续进行下去，直到商等于零为止。将所得各次余数，以最后余数为最高位数字，最先余数为最低位数字依次排列，就是所求二进制的各位数字。

例如，将 $(241)_{10}$ 转换为二进制数：

2 241		（低位）
2 120	得余数 1 ……b_0	↑
2 60	得余数 0 ……b_1	
2 30	得余数 0 ……b_2	
2 15	得余数 0 ……b_3	
2 7	得余数 1 ……b_4	
2 3	得余数 1 ……b_5	
2 1	得余数 1 ……b_6	
0	得余数 1 ……b_7	（高位）

于是得到$(241)_{10}=(11110001)_2$

对于十进制小数，可用乘 2 取整法：

先用“2”乘十进制纯小数，然后去掉乘积中的整数部分，再用 2 去乘剩下的纯小数部分，如此继续下去，直到满足所要求的精确度或直至纯小数部分等于零为止。把每次乘积的整数部分由上而下依次排列起来，即得所求二进制纯小数的小数点后各位数字。

例如，将 0.5625 转换为二进制小数，方法如下：

$$
\begin{array}{rll}
0.5625 & & \\
\times\quad 2 & & \\
\hline
1.1250 & \cdots\cdots\text{整数部分为 } 1 & b_{-1} \\
0.1250 & & \\
\times\quad 2 & & \\
\hline
0.2500 & \cdots\cdots\text{整数部分为 } 0 & b_{-2} \\
\times\quad 2 & & \\
\hline
0.5000 & \cdots\cdots\text{整数部分为 } 0 & b_{-3} \\
\times\quad 2 & & \\
\hline
1.0000 & \cdots\cdots\text{整数部分为 } 1 & b_{-4}
\end{array}
$$

所以$(0.5625)_{10}=(0.1001)_2$

当被转换的十进制数中既有整数又有小数时，应将其整数部分与小数部分分别进行转换，然后加以合并就可以得到转换结果。

2. 十进制与八进制、十六进制的相互转换

(1) 八进制、十六进制转换为十进制

要把八进制数转换为十进制数，只要将八进制数按“权”展开并相加，其和就是所要转换的十进制数。

例如，将八进制数$(153)_8$转换成十进制数：

$$(153)_8=1\times(8)^2+5\times(8)^1+3\times(8)^0=64+40+3=(107)_{10}$$

所以$(153)_8=(107)_{10}$

同样，把十六进制数转换为十进制数时，亦是将十六进制数按权展开并相加。

(2) 十进制转换成八进制、十六进制

上面讨论的十进制数转换成二进制数的原理和方法，同样适用于十进制数与八进制数，十六进制数的转换。

例如，把十进制数 676.49 转换成十六进制数：

先转换整数部分，用“除十六取余”的方法：

$$
\begin{array}{rlll}
16\overline{)676} & \cdots\cdots & \text{得余数}\quad 4 & a_0 \\
16\overline{)42} & \cdots\cdots & \text{得余数}\quad A & a_1 \\
16\overline{)2} & \cdots\cdots & \text{得余数}\quad 2 & a_2 \\
0 & & &
\end{array}
$$

即：$(676)_{10}=(2A4)_{16}$。

再转换小数部分，用“乘十六取整”的方法。

$$
\begin{array}{rll}
0.49 & & \\
\times\quad 16 & & \\
\hline
7.84 & \cdots\cdots\text{整数部分为 } 7 & a_{-1} \\
0.84 & & \\
\times\quad 16 & & \\
\hline
13.44 & \cdots\cdots\text{整数部分为 } D & a_{-2}
\end{array}
$$

$$
\begin{array}{r}
0.44 \\
\times \quad 16 \\
\hline
7.04 \qquad \cdots\cdots \text{整数部分为 } 7 \qquad a_{-3} \\
\vdots
\end{array}
$$

所以，$(0.49)_{10}=(0.7D7\cdots\cdots)_{16}$

最后答案为：$(676.49)_{10}=(2A4.7D7)_{16}$

3. 二进制与十六进制的相互转换

同二进制数与八进制数的转换一样，二进制与十六进制数之间存在着简单的转换关系。

由于二进制的权值 2^i 和十六进制的权值 $(16^i=2^{4i})$ 之间具有整指数倍数关系，即一位十六进制数相当于四位二进制数，故它们之间的转换十分简单。

(1) 二进制转换为十六进制

二进制数转换为十六进制数时，只要把二进制数以小数点为界分别向左、右每四位分为一组，最后一位不足四位者添零，写出与之对应的十六进制数即可。

例如，把二进制数 1110101.10100111 转换成十六进制数。

$$
\begin{array}{cccccc}
0111 & 0101 & . & 1010 & 0111 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
7 & 5 & . & A & 7
\end{array}
$$

即：$(1110101.10100111)_2=(75.A7)_{16}$

(2) 十六进制转换为二进制

每位十六进制数可以用四位二进制数表示，因此将十六进制数转换成二进制数时，只要写出每位十六进制数所对应的四位二进制数就行了。

例如，把十六进制数 59.A14.3C 转换成二进制数。

$$
\begin{array}{cccccccc}
5 & 9 & A & 1 & 4 & . & 3 & C \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
0101 & 1001 & 1010 & 0001 & 0100 & . & 0011 & 1100
\end{array}
$$

所以，$(59A14.3C)_{16}=(01011001101000010100.00111100)_2$

$=(1011001101000010100.001111)_2$

§1-4 计算机中的数和编码

一、编码

计算机中采用的是二进制数，因此，在计算机中表示的数、字母、符号等都以特定的二进制码来表示，这就是二进制编码。

1. 二—十进制 8-4-2-1 (BCD)码

8-4-2-1 码是最基本最简单的一种编码方案，应用十分广泛，这种码是把四位代码按自然二进制码的规律排列成 10 个码字，并指定前面 10 个码字依次表示 0 到 9 的 10 个数码，余下最后 6 个码字不用。如表 1-1 所示。

表 1-1　8-4-2-1二—十进制(BCD)码

十进制数	BCD 码
	8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

8-4-2-1码是一种有权码，各位的权值为8、4、2、1，故称为8-4-2-1 BCD代码。8-4-2-1码与自然二进制数有很好的对应关系，很容易实现彼此之间的转换。

8-4-2-1码的另一个优点是具有奇偶特征，凡是奇数码字的最低位皆为1，而偶数码字的最低位则为0。所以采用8-4-2-1码的十进制数容易判别奇偶性。

2. 字符编码：

在电子计算机中，字母和符号也是采用二进制代码表示的，最常用的字符编码是ASCII码。

表 1-2　ASCII 编码

字符 $b_7b_6b_5$ / $b_4b_3b_2b_1$	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0 0 0 0	NUL	DLE	SP	0	@	P	／	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	0/0	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	,	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	{
1 1 0 0	FF	FS	,	<	L	\	l	|
1 1 0 1	CR	GS	—	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	↑	n	~
1 1 1 1	SI	US	/	?	O	↓	o	DEL

表 1-3 ASCII 编码文字符的含意

字符	含　意	字符	含　意
NUL	空无效	DC1	设备控制 1
SOH	标题开始	DC2	设备控制 2
STX	正文开始	DC3	设备控制 3
ETX	本文结束	DC4	设备控制 4
EOT	传输结束	NAK	否定
ENQ	询问	SYN	空转同步
ACK	承认	ETB	信息组传输结束
BEL	报警符（可听信号）	CAN	作废
BS	退一格	EM	纸尽
HT	横何列表（穿孔卡片指令）	SUB	减
LF	换行	ESC	换码
VT	垂直制表	FS	文字分隔符
FF	走纸控制	GS	组分隔符
CR	回车	RS	记录分隔符
SO	移位输出	US	单元分隔符
SI	移位输入	SP	空间（空格）
DLE	数据键换码	DEL	作废

ASCII(American Standard Code for Information Interchange)是美国国家标准信息交换代码，它的编码表列于表 1-2 和表 1-3 中。这种编码也常见于通迅设备中，是一组八位二进制代码，用 b_1 到 b_7 七位二进制编码表示信息，用第八位作奇偶校验位，奇偶校验位的数值根据校验的类型决定是补奇还是补偶。

ASCII 码用七位（b_1 到 b_7）二进制代码进行编码时，可以得到 128 个码字，其中 26 个大写的英文字母，和 26 个小写的英文字母，共用去 52 个码字，数码用去 10 个码字，文字符用去 34 个码字，其余码字则分配给各种标点符号和运算符号等，没有剩余码字，其利用率是很高的，因为每个码字只表示一个字符，故不需要方式转换。ASCII 码所定义的各种符号，在各种计算装置中已足够使用。

ASCII 码目前已成为许多国家通用的一种国际标准码。

二、原码、补码、反码

1. 机器数

不带符号的数是数的绝对值，在绝对值前加上表示正负的符号就成了符号数。直接用正号“+”和负号“−”来表示其正负的二进制数叫做符号数的真值。

通常在计算机中，一个数的最高位为符号位，若字长为 8 位，即 D_7 为符号位，D_6-D_0 为数字位，符号位用“0”表示正，用“1”表示负，如：

$$X=(01010110)_2=+86$$

$$X=(11010110)_2=-86$$

把符号数值化以后，就能将它用于机器中。机器中使用的符号数称做机器数，由符号位和数值两部分组成。除了符号数值化以外，机器数的特点还有字长一定，运算精度有限，以及必须规定小数点的位置等。常用的机器数有原码、补码和反码三种形式。

2. 原码

将数的真值形式中的正负号用代码 0 或 1 来表示时叫做数的原码形式，简称原码。

例如，

$$X=(+85)_{10}=+1010101$$

$$[X]_{原}=01010101$$

$$X=(-85)_{10}=-1010101$$

$$[X]_{原}=11010101$$

若字长为 n 位，原码一般可表示为：

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

当X为正数时$[X]_{原}$和X一样，即$[X]_{原}=X$。

当X为负数时$[X]_{原}=2^{n-1}-X$。由于X本身为负数，所以，实际上是将$|X|$数值部分绝对值前面的符号位上写成“1”即可。

原码表示法比较直观，但它的加减法运算较复杂。当两数相加时，机器首先要判断两数的符号是否相同，如果相同则两数相加，若符号不同，则两数相减。在做减法之前，还要判断两数绝对值的大小，然后用大数减去小数，最后再确定差的符号。要实现这些操作，电路就很复杂。为了把减法变成加法进行运算，就要借助于数的反码和补码表示法。

3. 补码

为了克服原码运算的缺点，采用另外两种符号数的表示法，即补码和反码。使用补码和反码，可以用加法来代替减法，完全消除了加法和减法的界限，这就使设备大大简化，另一方面符号位也和数值部分一起参加运算，不再需要专门处理符号的附加设备了。

在介绍补码之前，先看一个减法通过加法来实现的例子。假定现在是北京时间 6 点整，有一只手表却是 8 点整，比北京时间快了 2 小时，校准的方法有两种，一种是倒拨 2 小时，一种是正拨 10 小时。若规定倒拨是做减法，正拨是做加法，那么对手表来讲，减 2 与加 10 是等价的，也就是说减 2 可以用加 10 来实现。这是因为 8 加 10，等于 18，然而手表最大只能指示 12，当大于 12 时，12 自然丢失，18 减去 12 就只剩 6 了。这说明减法在一定条件下，是可以用加法代替的。这里“12”称为“模”，10 称为“−2”对模(12)的补数，将它们用数学式表达则为：

$$
\begin{array}{r}
8-2=(8+10)-12=6 \\
10 \\
+\quad 8 \\
\hline
\boxed{1}\,6 \\
\nearrow \quad \text{(以12为模)} \\
\text{自动丢掉}
\end{array}
$$

式中“模”12可以看作是向高位的进位。

从上例中可以得出这样的结论：求两个正整数8-2之差，可以用加上减数（2）的补数(10)，然后舍去进位（模）来实现。补数(10)就是模(12)与减数(2)的负数之和。推广到一般则有：

$$
\begin{aligned}
A-B&=A+(-B+M) \qquad \text{(以}M\text{为模)} \\
&=A+(-B)_{补}
\end{aligned}
$$

可见，在模为M的条件下，A减去B，可以用A加上$-B$的补数来实现。这里模(module)可视为计数器容量，对上述手表的例子，模为12。

现在我们再回过来考虑一下计算机运算的特点。计算机中的部件都有固定的位数，假定位数为n，则计数值为2^n，也即计数器容量为2^n，因此计算机中的补码是以“2^n”为模。

(1) 补码定义

$$
[X]_{补}=\begin{cases} X & 0\leqslant X<2^{n-1} \\ 2^n+X & -2^{n-1}\leqslant X<0 \pmod{2^n} \end{cases}
$$

以2^n为模的补码，也称为对2的补码。

补码有如下几个简单性质：

(a) 当X为正数时，补码和原码相同，当X为负数时，负数的补码等于

$$
2^n+X=2^n-|X|
$$

(b) $[+0]_{补}$和$[-0]_{补}$是相同的，所以在补码表示中，“0”的表示是唯一的。

(c) 字长为n位的补码，可以表示的数X的范围为：

$$
-2^{n-1}\leqslant X<2^{n-1}
$$

(2) 求补码的方法

由上面已知，正数的补码就是它本身，等于原码，只有负数才有求补的问题。这里我们介绍两种求补码的方法。

(a) 根据定义求：

$$
X=2^n+X=2^n-|X| \qquad X<0
$$

即负数X的补码等于模2^n加上其真值。（或减去其真值的绝对值）。

如：$X=-1010111 \qquad (n=8)$

则
$$
\begin{aligned}
[X]_{补}&=2^8+[-1010111]=2^8-|-1010111| \\
&=100000000-1010111 \\
&=10101001 \pmod{2^8}
\end{aligned}
$$

这种方法因为要作一次减法，很不方便。

(b) 利用原码求：符号位除外，求反加1。

计算机通常是将原码数值部分按位求反，即“1”变“0”，“0”变“1”，再在最低位加1

来求补码，而符号位不变。

例如，$X=-1010101$　　$[X]_{原}=11010101$

则　$[X]_{补}=\text{囗}0101010+1$

$=\text{囗}0101011$

即　$[-1010101]_{原}=\text{囗}1010101$

↓求反

囗0101010

\+　　1

囗0101011$=[X]_{补}$

如果将$[X]_{补}$再求一次补，即将$[X]_{补}$除符号位以外变反加1就得到$[X]_{原}$。

如果已经知道$[X]_{补}=X_nX_{n-1}X_{n-2}\cdots X$，那么对$[X]_{补}$的每一位（包括符号位）都按位求反，然后再加1，结果即为$[-X]_{补}$。由$[X]_{补}$求$[-X]_{补}$，通常称为变补。

(3) 补码运算

带符号数的运算的基本规则如下：

(*a*) 数的最高位是符号位，0代表正数，1代表负数，把符号也看成数，一同参加运算。

(*b*) 参加运算的数都用补码方式表示。正数的补码就是它自己，负数的补码表示为：

$$2^n+X=2^{n-1}+(2^{n-1}-|X|)\qquad X<0$$

(*c*) 对于补码的加减法可用下面一般公式表示：

$$[X]_{补}+[\pm Y]_{补}=[X\pm Y]_{补}$$

即两数相加或相减，都通过两个补码的加法运算来实现，结果也以补码形式表示。若结果的符号为0表示正数，为1时表示负数。

(4) 补码溢出判别

上面已经指出，对于位数为n的补码，可表示的数X的范围为$-2^{n-1}\leqslant X<2^{n-1}$，因而若运算结果数超出上述范围，则运算结果出错，这种现象称为“溢出”。那么，如何判断“溢出”呢？这里主要介绍微型计算机中采用的“双高位”判别法。

所谓“双高位”判别法，就是用字长最高位及次高位的进位状态来判别溢出的方法。两数相加发生溢出，只可能有两种情况，即两数同时为正或同时为负，并且其和的绝对值又大于2^{n-1}时。两个正数相加，若数值部分之和大于2^{n-1}，则数值部分必有进位，即Cp=1，而符号位却无进位，即Cs=0，这种Cs Cp的状态为“01”时的溢出称为“正溢出”。同样，若两个负数相加，出现Cs Cp的状态为“10”，则为“负溢出”。

Cs表示字长最高位（符号位）的进位，若有进位，Cs=1，否则Cs=0。

Cp表示数值部分最高位（字长次最高位）的进位，若有则Cp=1，否则Cp=0。

在微型机中，常用异或线路来检查Cs Cp状态，即溢出$V=Cs\oplus Cp$。

当Cs Cp同号时(00,11)，不产生溢出，$V=0$。而当Cs Cp异号(10,01)时，便产生溢出，$V=1$。

4. 反码

对原码除符号位外，逐位求反，可得机器数的另一种表示法，即反码表示法。反码定义为：