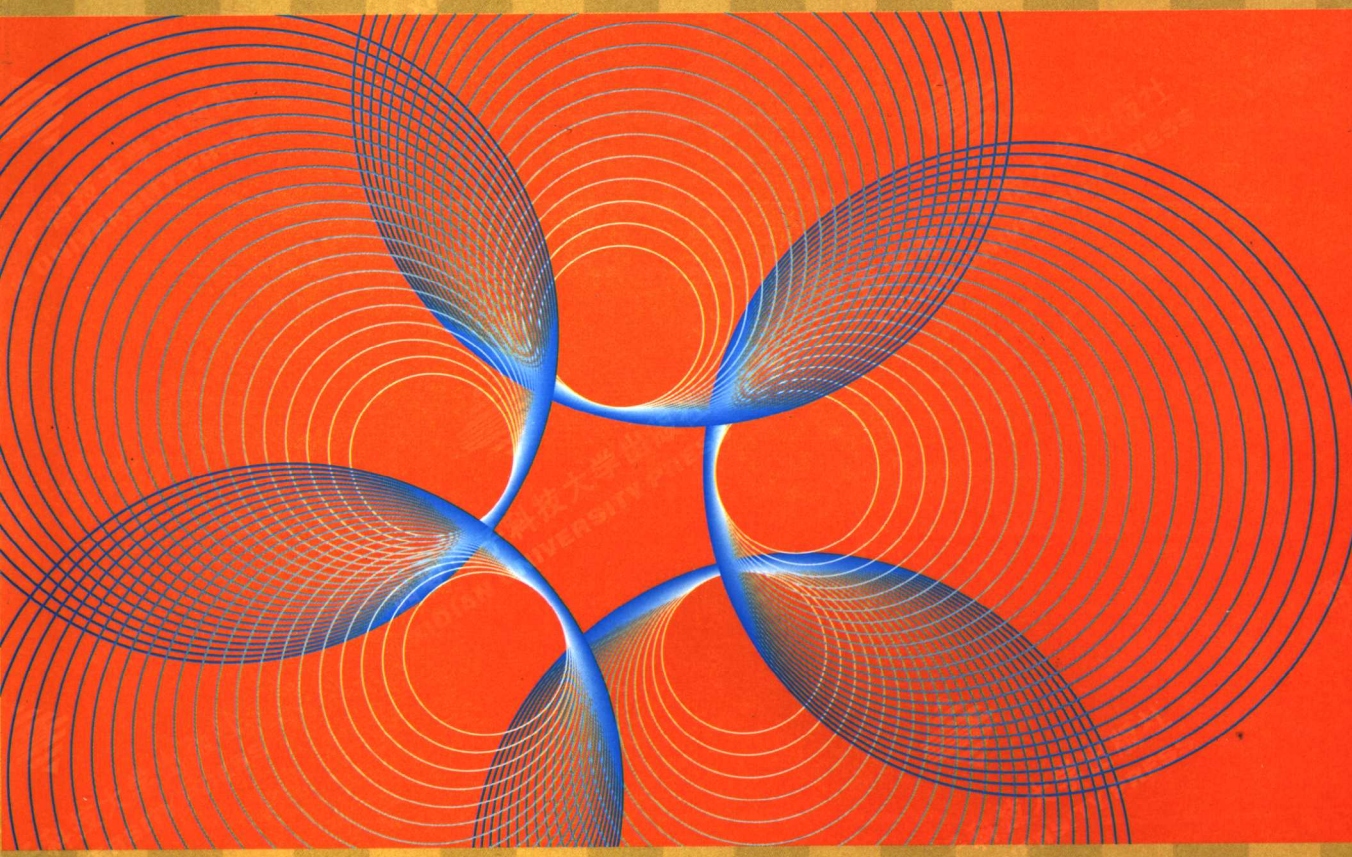
全国计算机技术与软件专业技术资格（水平）考试辅导用书



程序员考试辅导

全国计算机技术与软件专业技术资格（水平）考试办公室组编
张淑平　沈林兴　主编

西安电子科技大学出版社
http:// www.xduph. com

全国计算机技术与软件专业技术资格(水平)考试辅导用书

程序员考试辅导

全国计算机技术与软件专业技术资格(水平)考试办公室组编

张淑平　沈林兴　主编

西安电子科技大学出版社

2004

内 容 简 介

本书是根据《计算机技术与软件专业技术资格(水平)考试大纲(程序员级)》编写的考试辅导书。全书共11章，主要内容包括：计算机系统基础知识，操作系统基础知识，数据库基础知识，多媒体基础知识，网络基础知识，程序语言基础知识，软件工程基础知识，数据结构与算法，标准化基础知识，Visual Basic程序设计基础知识以及算法与C语言程序设计。每章均包括学习目标与要求、知识点概述、典型例题与分析、强化训练习题和参考答案。

本书浓缩了考试复习内容，知识精练，重点突出，例题丰富，解答详细，既可作为计算机技术与软件专业技术资格(水平)考试的应试辅导教材，也可作为大专院校师生的教学参考书。

图书在版编目(CIP)数据

程序员考试辅导/张淑平，沈林兴主编. —西安：西安电子科技大学出版社，2004.8

全国计算机技术与软件专业技术资格(水平)考试辅导用书

ISBN 7-5606-1435-3

Ⅰ.程... Ⅱ.①张...②沈... Ⅲ.程序设计—水平考试—自学参考资料

Ⅳ.TP311.1

中国版本图书馆CIP数据核字(2004)第068518号

策　　划　臧延新　陈宇光

责任编辑　张晓燕

出版发行　西安电子科技大学出版社(西安市太白南路2号)

电　　话　(029)88242885　88201467　　邮　编　710071

http://www.xduph.com　　E-mail：xdupfxb@pub.xaonline.com

经　　销　新华书店

印　　刷　陕西华沐印刷科技有限责任公司

版　　次　2004年8月第1版　2004年10月第4次印刷

开　　本　787毫米×1092毫米　1/16　印张27.75

字　　数　663千字

印　　数　14 001～16 000册

定　　价　40.00元

ISBN 7-5606-1435-3/TP·0765

XDUP　1706001—4

前　言

全国计算机软件考试实施至今已经历了十多年，在社会上产生了很大的影响，对我国软件产业的形成和发展做出了重要的贡献。为适应我国信息化发展的需要，国家人事部和信息产业部决定将考试的级别拓展到计算机技术与软件的各个方面，以满足社会对各种信息技术人才的需求。

为了适应新的考试大纲要求，编者受全国计算机技术与软件专业技术资格(水平)考试办公室委托，在《程序员教程》一书的基础上编写了这本关于程序员考试的辅导用书。由于在考试大纲中要求考生掌握的知识面很广，学习有一定的难度，因此，作者在编写本书时，注意与教材结合，与教材内容同步，对教材中的难点和重点知识进行了补充。本书的每一章均由学习目标与要求、知识点概述、重点与难点分析和强化训练习题四部分组成。

全书共分11章，由张淑平和沈林兴担任主编。第1章计算机系统基础知识由张淑平、李伯成编写，第2章操作系统基础知识、第3章数据库基础知识由王亚平编写，第4章多媒体基础知识由刘强编写，第5章网络基础知识由张凤琴编写，第6章程序语言基础知识由张淑平编写，第7章软件工程基础知识由褚华编写，第8章数据结构与算法由张淑平、王卫东编写，第9章标准化基础知识由刘强编写，第10章Visual Basic程序设计基础知识由沈林兴编写，第11章算法与C语言程序设计由张淑平编写，最后由张淑平统稿。

在本书的编写过程中，参考了许多相关的书籍和资料，编者在此对这些参考文献的作者表示感谢。同时感谢西安电子科技大学出版社在本书出版过程中所给予的支持和帮助。

因作者水平有限，书中难免存在错漏和不妥之处，望读者指正，以利改进和提高。

编　者

2004年4月于西安电子科技大学

目　录

第 1 章　计算机系统基础知识

1.1　学习目标与要求

本章的主要内容包括：计算机系统的基本组成和结构，计算机中数据的表示和基本运算，指令系统的基本概念和系统安全的基本知识。

★ 通过本章的学习，要求掌握如下内容：

(1) 理解计算机系统的基本组成及工作原理，熟悉中央处理器(CPU)、存储系统及输入输出系统的基本组成和结构。

(2) 掌握计算机中的常用数制，掌握十进制、二进制、八进制和十六进制之间相互转换的方法。

(3) 理解数据的机内表示方法，掌握原码、补码、反码、移码等码制及其特点。

(4) 掌握基本的算术和逻辑运算方法。

(5) 理解常用校验码的原理和特点，了解海明码、循环冗余码的编码方法和校验方法，掌握奇偶校验的原理和方法。

(6) 了解计算机的指令系统，包括指令格式、基本寻址方式、指令类型等基本概念。

(7) 了解有关计算机安全的基本知识，了解病毒的概念及基本原理。

1.2　知 识 点 概 述

1.2.1　计算机系统的组成

一、要求掌握的知识要点

(1) 了解计算机的类型和发展概况。

(2) 了解计算机系统的层次结构。

二、知识点概述

(一) 计算机系统概述

1. 计算机的类型

按照计算机中使用的电路器件划分，计算机经历了电子管器件时代、晶体管器件时代、中小规模集成电路时代、大规模及超大规模集成电路时代。按照计算机的类型划分，有超级计算机(Supercomputer)、大型机(Mainframe Computer)、小型机(Minicomputer)、微型机(Microcomputer)、专用计算机(Special-purpose Computer)等，其中微型机包括台

式计算机(Desktop)、膝上型电脑(Laptop)或笔记本电脑(Notebook)、工作站(Workstation)、掌上型电脑、个人数字助理(Personal Digital Assistant，PDA)等类型的计算机。

2. 计算机系统的基本组成

计算机系统是由硬件系统和软件系统组成的。计算机硬件是计算机系统中看得见、摸得着的物理装置，计算机软件是程序、数据和相关文档的集合。计算机系统的基本组成如图 1 - 1 所示。

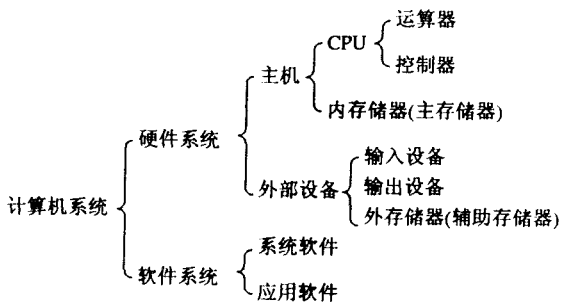


图 1 - 1 计算机系统的基本组成

3. 计算机系统的层次结构

人们用各种电路器件制造的计算机称为物理计算机或裸机。裸机使用的语言是二进制形式表示的机器语言。在机器语言的基础上，人们开发了抽象层次更高、更接近于人类自然语言的计算机语言，包括汇编语言和高级语言。汇编语言和高级语言需要进行翻译才能被计算机硬件识别。不同层次的用户使用不同层次的计算机语言与计算机进行交互，使计算机实现用户的要求，因此，可以把计算机看成是一个多层次的系统，如图 1 - 2 所示。裸机以上的用户所操作的计算机可以看成是一台相应层次的虚拟计算机。

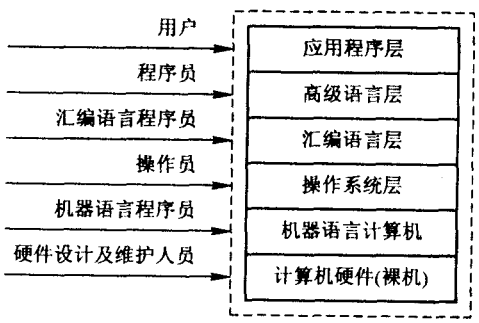


图 1 - 2 计算机系统的层次结构

(二) 计算机系统的硬件

基本的计算机硬件系统由运算器、控制器、存储器、输入设备和输出设备五大部分组成。随着微电子技术的发展，运算器、控制器等部件已被集成在一起，统称为中央处理单元(Central Processing Unit，CPU)。CPU 是硬件系统的核心，用于数据的加工处理，能完成各种算术、逻辑运算及控制功能。

运算器是对数据进行加工处理的部件，它主要完成算术和逻辑运算。控制器的主要功能则是从主存中取出指令并进行分析，控制计算机的各个部件有条不紊地完成指令的功能。

存储器是计算机系统中的记忆设备，分为内部存储器(Main Memory，MM，简称内存、主存)和外部存储器(简称外存)。内存的存取速度高，容量小，一般用来临时存放计算机运行时所需的程序、数据及中间结果。外存的容量大，存取速度慢，可用于长期保存信息。寄存器是 CPU 中的一种存储器件，用来临时存放指令、数据及运算结果。与内部存储器相比，寄存器的存取速度要快得多。

习惯上将 CPU 和主存的有机组合称为主机。输入/输出(或 I/O)设备位于主机之外，是计算机系统与外界交换信息的装置。

1.2.2　计算机中数据的表示及运算

一、要求掌握的知识要点

(1) 掌握不同数制之间的转换方法。

(2) 掌握数值数据的原码、反码、补码和移码表示。

(3) 掌握定点数和浮点数的表示及加减运算。

(4) 掌握奇偶校验编码方法，了解海明校验和循环冗余校验的原理。

(5) 掌握逻辑运算，了解逻辑表达式的化简方法。

(6) 了解非数值数据的编码方法。

二、知识点概述

(一) 计算机中数据的表示

计算机中采用二进制表示数据。计算机中处理的数据可分为两类：数值数据和非数值数据。数值数据指表示数量的数据，有正负和大小之分。非数值数据主要包括字符、声音、图像等，在计算机中存储和处理之前必须以某种编码形式转换为二进制表示形式。

1. 进位计数制及其转换

任何一种进位计数制表示的数都可以写成按权展开的多项式之和，即任意一个 r 进制数 N 可表示为

$$N_r = \sum_{i=m-1}^{-k} D_i \times r^i$$

其中的 D_i 为该数制采用的基本数符，r^i 是权，r 是基数。

例如，十进制数 1234.55 可表示为

$$1234.55 = 1\times10^3 + 2\times10^2 + 3\times10^1 + 4\times10^0 + 5\times10^{-1} + 5\times10^{-2}$$

计算机中常用的计数制有二进制、八进制、十六进制等。

2. 进位计数制之间的相互转换

(1) 将二进制数转换为十进制数时，把二进制数各位按权展开求和。

(2) 将十进制数转换成二进制数时，整数部分和小数部分分别转换，然后再合并。十进制整数转换为二进制整数的方法是“除 2 取余”；十进制小数转换为二进制小数的方法是“乘 2 取整”。十进制数转换成二进制数还有一种简便的方法：把一个十进制数写成按二进制数权的大小展开的多项式，按权值从高到低依次取各项的系数就可得到相应的二进制数。

(3) 二进制数与八进制数、十六进制数之间的对应关系如表 1 - 1 所示。

表 1-1 二进制数与八进制数、十六进制数之间的对应关系

二进制	八进制	二进制	十六进制	二进制	十六进制
000	0	0000	0	1000	8
001	1	0001	1	1001	9
010	2	0010	2	1010	A
011	3	0011	3	1011	B
100	4	0100	4	1100	C
101	5	0101	5	1101	D
110	6	0110	6	1110	E
111	7	0111	7	1111	F

可类比推出十进制数转换为 r 进制数的方法：将十进制数的整数部分“除 r 取余”得到对应 r 进制数的整数部分；将十进制数的小数部分“乘 r 取整”得到 r 进制数的小数部分。

3. 二进制数的运算规则

(1) 二进制加法的进位规则是“逢二进一”：

$0+0=0$　　$1+0=1$　　$0+1=1$　　$1+1=0$(有进位)

(2) 二进制减法的借位规则是“借一当二”：

$0-0=0$　　$1-0=1$　　$1-1=0$　　$0-1=1$(有借位)

(3) 二进制乘法规则是：

$0\times0=0$　　$1\times0=0$　　$0\times1=0$　　$1\times1=1$

(4) 二进制除法是乘法的逆运算，其运算方法与十进制除法是一样的。

4. 机器数和码制

各种数据在计算机中表示的形式称为机器数，其特点是采用二进制计数制，数的符号用 0、1 表示，小数点隐含表示，不占位置。机器数对应的实际数值称为数的真值。

机器数有无符号数和带符号数之分。无符号数表示正数，在机器数中没有符号位。对于无符号数，若约定小数点的位置在机器数的最低位之后，则是纯整数；若约定小数点的位置在机器数的最高位之前，则是纯小数。对于带符号数，机器数的最高位是表示正、负的符号位，其余位则表示数值。若约定小数点的位置在机器数的最低数值位之后，则是纯整数；若约定小数点的位置在机器数的最高数值位之前(符号位之后)，则是纯小数。

为了便于运算，带符号的机器数可采用原码、反码和补码等不同的编码方法，机器数的这些编码方法称为码制。

1) *原码表示法*

数值 X 的原码记为$[X]_{原}$。如果机器字长为 n(即采用 n 个二进制位表示数据)，则原码定义如下：

若 X 是纯整数，则

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

若 X 是纯小数，则

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

在原码表示法中，最高位是符号位，0 表示正号，1 表示负号；其余的 n－1 位表示数值的绝对值。

2）反码表示法

数值 X 的反码记作$[X]_{反}$。如果机器字长为 n，则反码定义如下：

若 X 是纯整数，则

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

若 X 是纯小数，则

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

在反码表示中，最高位是符号位，0 表示正号，1 表示负号，正数的反码与原码相同，负数的反码则是其绝对值按位求反。

3）补码表示法

数值 X 的补码记作$[X]_{补}$。如果机器字长为 n，则补码定义如下：

若 X 是纯整数，则

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

若 X 是纯小数，则

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

在补码表示中，最高位为符号位，0 表示正号，1 表示负号；正数的补码与其原码和反码相同，负数的补码则等于其反码加 1。

4）移码表示法

移码表示法是在数 X 上增加一个偏移量来定义的，常用于表示浮点数中的阶码。如果机器字长为 n，规定偏移量为 2^{n-1}，则移码定义如下：

若 X 是纯整数，则

$$[X]_{移}=2^{n-1}+X \qquad -2^{n-1}\leqslant X<2^{n-1}$$

实际上，在偏移量为 2^{n-1} 的情况下，只要将补码的符号位取反即可获得相应的移码表示。

5. 定点数和浮点数

1）定点数

所谓定点数，就是小数点的位置固定不变的数。小数点的位置通常有两种约定方式：定点整数(纯整数，小数点在最低有效数值位之后)和定点小数(纯小数，小数点在最高有效数值位之前)。

设机器字长为 n，各种码制表示下的带符号数的范围如表 1－2 所示。

表 1－2 机器字长为 n 时表示的带符号数的范围

码制	定点整数	定点小数
原码	$-(2^{n-1}-1)\sim+(2^{n-1}-1)$	$-(1-2^{-(n-1)})\sim+(1-2^{-(n-1)})$
反码	$-(2^{n-1}-1)\sim+(2^{n-1}-1)$	$-(1-2^{-(n-1)})\sim+(1-2^{-(n-1)})$
补码	$-2^{n-1}\sim+(2^{n-1}-1)$	$-1\sim+(1-2^{-(n-1)})$
移码	$-2^{n-1}\sim+(2^{n-1}-1)$	$-1\sim+(1-2^{-(n-1)})$

2）浮点数

当机器字长为 n 时，定点数的补码和移码可表示 2^{n-1} 个数，而其原码和反码只能表示 $2^{n-1}-1$ 个数(0 占用了两个编码)。因为定点数所能表示的数值范围比较小，运算中很容易因结果超出范围而溢出，所以引入了浮点数。浮点数是小数点位置不固定的数，它能表示更大范围的数。

二进制数 N 的浮点数表示方法为

$$N=2^{E}\times F$$

其中，E 称为阶码，F 称为尾数。

在浮点数表示法中，阶码通常为带符号的纯整数，尾数为带符号的纯小数。浮点数的一般表示格式如下：

阶符	阶码	数符	尾数

很明显，一个数的浮点表示不是惟一的。当小数点的位置改变时，阶码也随之相应改变，因此可以用多种浮点形式表示同一个数。

浮点数所能表示的数值范围主要由阶码决定，所表示数值的精度则由尾数决定。为了充分利用尾数来表示更多的有效数字，通常对浮点数进行规格化。规格化就是将尾数的绝对值限定在区间[0.5，1)内。当尾数用补码表示时，需要注意：

(1) 若尾数 F≥0，则其规格化的尾数形式为 F=0.1×××…×。其中×可为 0，也可为 1，即将尾数限定在区间[0.5，1)内。

(2) 若尾数 F<0，则其规格化的尾数形式为 F=1.0×××…×。其中×可为 0，也可为 1，即将尾数 F 的范围限定在区间[－1，－0.5)内。

如果浮点数的阶码(包括 1 位阶符)用 R 位的移码表示，尾数(包括 1 位数符)用 M 位的补码表示，则这种浮点数所能表示的数值范围为

最大的正数 $+(1-2^{-M+1})\times 2^{2^{R-1}-1}$

最小的负数 $-1\times 2^{2^{R-1}-1}$

6. 十进制数与字符的编码表示

用四位二进制代码表示一位十进制数，称为二-十进制编码，简称 BCD 编码。因为 $2^4=16$，而十进制数只有 0～9 十个不同的数符，故有多种 BCD 编码。根据四位代码中每一位是否有确定的权来划分，可分为有权码和无权码两类。

应用最多的有权码是 8421 码，即四个二进制位的权从高到低分别为 8、4、2 和 1。无权码中使用较多的是余 3 码和格雷码。余 3 码是在 8421 码的基础上，把每个数的代码加上 0011 后构成的。格雷码的编码规则是相邻的两个代码之间只有一位不同。

常用的 8421BCD 码、余 3 码、格雷码与十进制数的对应关系如表 1－3 所示。

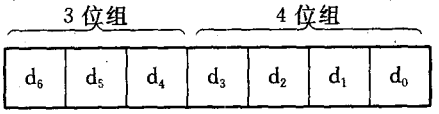
表 1－3　8421BCD 码、余 3 码、格雷码与十进制数的对应关系

十进制数	8421BCD 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	1110
6	0110	1001	1010
7	0111	1010	1000
8	1000	1011	1100
9	1001	1100	0100

7. ASCII 码

ASCII 码(American Standard Code for Information Interchange)是美国标准信息交换码的简称,该编码已被国际标准化组织 ISO 采纳,成为一种国际通用的信息交换用标准代码。

ASCII 码采用 7 个二进制位对字符进行编码，可表示 128 个符号。ASCII 码的低 4 位组 $d_3d_2d_1d_0$ 用作行编码，高 3 位组 $d_6d_5d_4$ 用作列编码，其格式为



根据 ASCII 码的构成格式，可以很方便地从对应的代码表中查出每一个字符的编码。

EBCDIC 是扩展的二-十进制交换码，采用 8bit 编码表示一个字符，共可以表示 256 个不同的符号，常用于 IBM 大型机中。

8. 汉字编码

在一个汉字处理系统中，输入、内部处理、存储和输出对汉字代码的要求不尽相同，所采用的编码也不相同。在使用汉字信息处理系统处理汉字和词语时，关键的问题是要进行一系列的汉字代码转换。汉字的编码方法有多种。

1) 输入码

汉字输入码的编码方法主要有三类：数字编码、拼音码和字形编码。

(1) 数字编码。数字编码用数字串代表一个汉字的输入，常用的是国标区位码。国标区位码将国家标准局公布的 6763 个两级汉字分成 94 个区，每个区 94 位。使用区位码方法输入汉字时，必须先在表中查找汉字并找出对应的代码。数字编码输入的优点是无重码，而且输入码和内部编码的转换比较方便，但是每个编码都是等长的数字串，代码难以记忆。

(2) 拼音码。拼音码是以汉语读音为基础的输入方法。由于汉字同音字太多，输入重码率很高，因此，按拼音输入后还必须进行同音字选择，影响了输入速度。

(3) 字形编码。字形编码是以汉字的形状确定的编码。五笔字形编码是最有影响的字形编码方法。

2) 内部码

汉字内部码(简称汉字内码)是汉字在设备或信息处理系统内部最基本的表达形式，是

在设备和信息处理系统内部存储、处理、传输汉字用的代码。在西文的表示中没有交换码和内码之分。汉字数量多，用一个字节无法区分，可采用国家标准局 GB 2312－80 中规定的汉字国标码，用两个字节存放一个汉字的内码，每个字节的最高位置"1"，以此作为汉字机内码。由于两个字节各使用了 7 位，因此可表示 16 384 个可区别的机内码。

为了统一地表示世界各国的文字，1993 年国际标准化组织公布了"通用多八位编码字符集"的国际标准 ISO/IEC 10646，简称 UCS(Universal Code Set)。UCS 包含了中、日、韩等国的文字，这一标准为包括汉字在内的各种正在使用的文字规定了统一的编码方案。

我国相应的国家标准为 GB 13000，详细内容请查阅网址：http://www.unicode.org。

3) 字形码

汉字字形码是表示汉字字形的字模数据，通常有点阵、矢量函数等表示方式。用点阵表示字形时，汉字字形码指的就是这个汉字字形点阵的代码。字形码也称字模码。根据输出汉字的要求不同，点阵的多少也不同，简易型汉字为 16×16 点阵，高精度型汉字为 24×24 点阵、32×32 点阵、48×48 点阵等。

汉字的矢量表示法将汉字看成是由笔画组成的图形，提取每个笔画的坐标值，这些坐标值就可以决定每一笔画的位置，将每一个汉字的所有坐标值信息组合起来就是该汉字字形的矢量信息。显然，汉字的字形不同，其矢量信息也就不同，每个汉字都有自己的矢量信息。由于汉字的笔画不同，矢量信息不同，因而每个汉字矢量信息所占的存储空间大小也不一样。

(二) 机器数的运算

1. 定点数的加、减运算

计算机中通常只设置加法器，减法运算是转换为加法运算来实现的。对原码表示的机器数进行运算时，符号位和数值位需要分别处理，因此计算机的加、减运算中常采用机器数的补码表示形式。

1) 补码加、减法

(1) 补码加法的运算法则是：和的补码等于补码的和，即$[X+Y]_{补}=[X]_{补}+[Y]_{补}$。

(2) 补码减法的运算法则是：差的补码等于被减数的补码加上减数取负后的补码，即$[X-Y]_{补}=[X]补+[-Y]_{补}$。因此，在补码表示中，可将减法运算转换为加法运算。

(3) 由$[X]_{补}$求$[-X]_{补}$的方法是：$[X]_{补}$的各位取反(包括符号位)，末位加"1"。

【例 1.1】 设二进制整数 X＝＋1000100，Y＝＋1110，求 X＋Y、X－Y 的值。

解： 设用 8 位补码表示带符号数据，由于 X 和 Y 都是正数，因而$[X]_{补}=01000100$，$[Y]_{补}=00001110$，那么$[-Y]_{补}=11110010$，则

01000100	01000100	01000100
＋ 00001110	－ 00001110	＋ 11110010
01010010	00110110	00110110
$[X]_{补}+[Y]_{补}$	X－Y	$[X]_{补}+[-Y]_{补}$

由于 X 和 Y 均是正数，因此 X+Y 的值就等于$[X]_{补}+[Y]_{补}$，即 X+Y=+1010010；因为 X 的绝对值大于 Y 的绝对值，所以 X−Y 的值就等于$[X]_{补}+[-Y]_{补}$，即 X−Y = +110110。

2）溢出及其判定

在确定了运算的字长和数据的表示方法后，数据的范围也就确定了。一旦运算结果超出范围，就会发生溢出。

当两个同符号的数相加(或者是相异符号的数相减)时，运算结果有可能溢出。

【例 1.2】 设正整数 X = +1000001，Y = +1000011，若用 8 位补码表示，则$[X]_{补}$=01000001，$[Y]_{补}$=01000011，求$[X+Y]_{补}$。

解： 计算$[X]_{补}+[Y]_{补}$为

$$\begin{array}{rr} & 0\ 1000001 \\ + & 0\ 1000011 \\ \hline & 1\ 0000100 \end{array}$$

两个正数相加的结果为一个负数，结果显然是荒谬的，产生此错误的原因就是溢出。

常用的溢出检测机制主要有双符号位判决法和进位判决法。

(1) 双符号位判决法。若采用两位表示符号，即 00 表示正号、11 表示负号，则溢出时两个符号位就不一致了，从而可以判定发生了溢出。

【例 1.3】 设正整数 X = +1000001，Y = +1000011，若采用双符号位补码表示，则$[X]_{补}$=00 1000001，$[Y]_{补}$=00 1000011，求$[X+Y]_{补}$。

解： 计算$[X]_{补}+[Y]_{补}$为

$$\begin{array}{rr} & 00\ 1000001 \\ + & 00\ 1000011 \\ \hline & 01\ 0000100 \end{array}$$

由于两个符号位不一致，因此可断定发生溢出，运算结果不反映 X+Y 的真值。

(2) 进位判决法。令 C_{n-1}表示最高数值位向最高位的进位，C_n 表示符号位的进位，当 $C_{n-1}\oplus C_n=1$ 时表示发生溢出。

2. 定点数的乘、除运算

1）定点乘法

定点乘法有定点原码乘法和定点补码乘法之分。原码一位乘法的运算法则是：

(1) 乘积的符号由两个乘数的符号位经异或后产生。

(2) 将两个乘数的数值部分相乘就得到乘积的数值部分(原码表示)，运算时将部分积和乘数进行右移，再将部分积和被乘数的数值部分相加。

(3) 将乘积的符号与其数值部分拼接在一起就构成了积的原码。

原码乘法在实现过程中要将符号和数值分别处理，不仅麻烦，而且影响执行速度。由 Booth 夫妇首先提出的补码乘算法可将包括符号位在内的两个补码数直接相乘，一次运算就可以获得包括符号位在内的乘积。

2）定点除法

定点除法常采用原码除法，基本方法有恢复余数法和加减交替法。

3. 浮点运算

1）浮点加、减运算

设有浮点数 $X=M\times 2^i$，$Y=N\times 2^j$，求 X±Y 的运算过程如下：

（1）对阶：使两个数的阶码相同。令 $K=|i-j|$，把阶码小的数的尾数右移 K 位，使其阶码加上 K。若尾数用补码表示，则尾数右移时，符号位参与移位且保持不变；若尾数用原码表示，则符号位不参加移位，移位时尾数的高位补 0。

（2）求尾数和(差)。

（3）结果规格化：若运算所得的尾数不是规格化的数，则需要进行规格化处理。当尾数溢出时，尾数右移一位，阶码加 1。用补码表示的尾数最高位与符号位相同时，尾数应向左移。尾数每左移一位，阶码就减 1，直到尾数最高位与符号位相反为止。

（4）舍入：在对结果右规时，尾数的最低位将因移出而丢掉；另外，在对阶过程中也会将尾数右移使最低位丢掉，这就需要进行舍入处理，以求得最小的运算误差。常用的舍入处理方法有：

· 截断法。将要保留的数据末位右边的数据全都截去，不管数据是 0 还是 1。

· 末位恒 1 法。将要保留的末位数据恒置 1，不管右移丢掉的数据是 0 还是 1。

· 0 舍 1 入法。舍去的数据为 0 时，保持末位原始状态；若舍去的数据为 1，则将末位加 1。这类似于十进制中的四舍五入，但在数据为 0.1111…1，即在尾数全为 1 的特殊情况下，这种舍入会再次产生溢出，遇到这种情况可用硬件进行判断，并在舍去 1 时末位不再加 1。

（5）溢出判别：以阶码为准，若阶码溢出，则运算结果溢出；若阶码下溢，则结果为 0；否则结果正确，无溢出。

2）浮点乘、除法运算

浮点数相乘，其积的阶码等于两乘数的阶码相加，积的尾数等于两乘数的尾数相乘。浮点数相除，其商的阶码等于被除数的阶码减去除数的阶码，商的尾数等于被除数的尾数除以除数的尾数。乘、除运算的结果都需要进行规格化处理并判断阶码是否溢出。

（三）逻辑代数及逻辑运算

逻辑代数是 1849 年英国数学家乔治·布尔提出的，它是用代数的方式对逻辑变量进行描述和分析的数学工具，也称为布尔代数。逻辑变量的取值只有“真”和“假”，通常以 1 表示“真”，0 表示“假”。

1. 基本的逻辑运算

在逻辑代数中有三种最基本的运算：“与”运算、“或”运算、“非”运算，其他逻辑运算可用这三种基本运算的组合来表示。

1）“与”运算

“与”运算又称为逻辑乘，其运算符号常用 AND、∩、∧或“·”表示。设 A 和 B 为两个逻辑变量，当且仅当 A 和 B 的取值都为“真”时，A“与”B 的值为“真”，否则 A“与”B 的值为“假”，即 1·1=1，0·0=0·1=1·0=0。

2）“或”运算

“或”运算也称为逻辑加，其运算符号常用 OR、∪、∨或“+”表示。设 A 和 B 为两个逻辑变量，当且仅当 A 和 B 的取值都为“假”时，A“或”B 的值为“假”，否则 A“或”B 的值为“真”，即 1+1=0+1=1+0=1，0+0=0。

3）“非”运算

“非”运算也称为逻辑求反运算，常用 $\overline{A}$ 表示对变量 A 的值求反。其运算规则很简单：$\overline{1}=0$，$\overline{0}=1$。

4）“异或”运算

常用的逻辑运算还有“异或”运算，又称为半加运算，其运算符号常用 XOR 或⊕表示。设 A 和 B 为两个逻辑变量，当且仅当 A、B 的值不同时，A“异或”B 为真。A“异或”B 的运算可由前三种基本运算表示，即 $A\oplus B=\overline{A}\cdot B+A\cdot\overline{B}$。

2. 逻辑表达式及其化简

1）逻辑表达式与真值表

逻辑表达式就是用逻辑运算符把逻辑变量(或常量)连接在一起表示某种逻辑关系的表达式。我们常常用表格来描述一个逻辑表达式与其变量之间的关系，也就是把变量和表达式的各种取值都一一对应列举出来，称之为真值表。

2）逻辑表达式的化简

利用逻辑运算的规律和一些常用的逻辑恒等式可以对一个逻辑表达式进行化简。

(四) 校验码

计算机系统运行时，各个部件之间要进行数据交换，为了确保数据在传送过程中正确无误，常采用的措施一是提高硬件电路的可靠性；二是提高代码的校验能力，包括查错能力和纠错能力。通常使用校验码来检测所传送的数据是否出错，其基本思想是把数据可能出现的编码分为两类：合法编码和错误编码。合法编码用于传送数据，错误编码是不允许在数据中出现的编码。合理地设计校验码的编码规则，可使得数据在传送中出现某种错误时会变成错误编码，这样就可以检测出接收到的数据是否有错。

码距是校验码中的一个重要概念。所谓码距，是指一个编码系统中任意两个合法编码之间至少有多少个二进制位不同。例如，4 位 8421 码的码距为 1，在传输过程中，该代码的 1 位或多位发生错误，都将变成另外一个合法编码，因此这种代码无检错能力。常用的校验码有奇偶校验码、海明码和循环冗余校验码。

1. 奇偶校验码(Parity Code)

奇偶校验是一种简单有效的校验方法。这种方法通过在编码中增加一位校验位来使编码中 1 的个数为奇数(奇校验)或者为偶数(偶校验)，从而使码距变为 2。对于奇校验，它可以检测代码中奇数位出错的情况，但不能发现偶数位出错的情况，即当合法编码中有奇数位发生了错误(编码中的 1 变成 0 或 0 变成 1)时，该编码中 1 的个数的奇偶性就会发生变化，从而可以发现错误。

目前应用的奇偶校验码有三种：水平奇偶校验码、垂直奇偶校验码和水平垂直校验码。

(1) 水平奇偶校验码：对每一个数据的编码添加校验位，使信息位与校验位处于同一行。

(2) 垂直奇偶校验码：把数据分成若干组，每一个数据占一行，排列整齐，再加一行校验码，针对同一组中的每一列采用奇校验或偶校验。