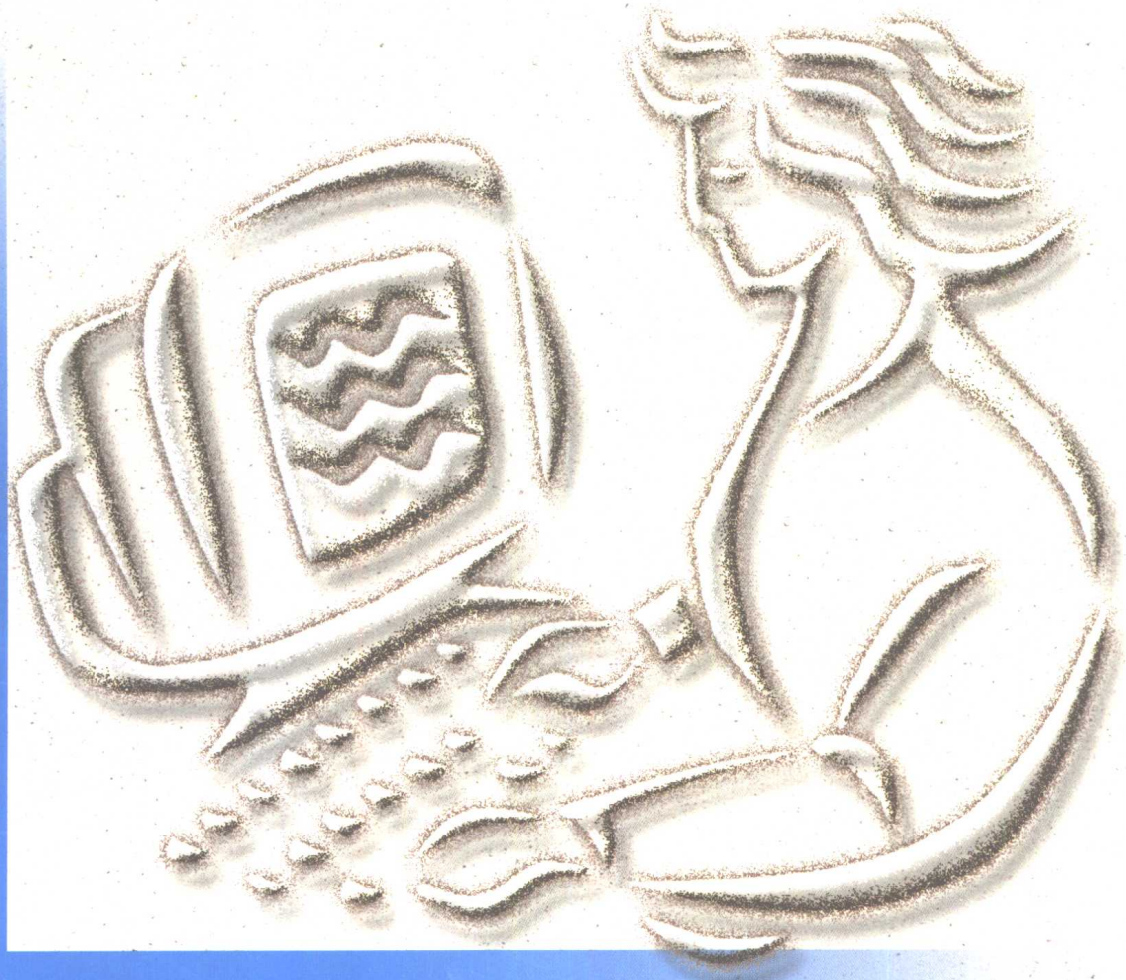
中国计算机软件专业
技术资格和水平考试辅导

程序员级
考试辅导书

陈明 编



科学出版社
http://www.sciencep.com

中国计算机软件专业技术资格和水平考试辅导

程序员级考试辅导书

陈 明 编

科学出版社
2002

内 容 简 介

本书是根据计算机软件专业技术资格和水平考试大纲（程序员级）编写的考试辅导书。本书共9章，主要内容包括：计算机硬件基础知识、程序语言知识、操作系统基础知识、软件工程基础知识、数据库基础知识、多媒体基础知识、网络基础知识、数据结构基础、常用算法设计方法，每章包括概述、知识点与难点、例题详解、练习题、小结和练习题答案。

本书将考试复习内容浓缩于内，知识精练，重点突出，例题丰富，解答详细，既可作为计算机程序设计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

程序员级考试辅导书/陈明编. —北京：科学出版社，2002

ISBN 7-03-010151-0

I.程… II.陈… 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（2002）第007295号

科学出版社 出版

北京东黄城根北街16号

邮政编码:100717

http://www.sciencep.com

新蕾印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2002年5月第 一 版 开本:787×1092 1/16

2002年5月第一次印刷 印张:16 3/4

印数:1—5 000 字数:385 000

定价:22.00元

（如有印装质量问题，我社负责调换〈环伟〉）

目　　录

第 1 章　计算机硬件基础知识

1.1　概　述

计算机系统由硬件和软件两部分构成，硬件是计算机系统中的实际装置，是系统的基础和核心，一般由中央处理器、存储器、输入/输出设备等组成。软件指的是操作系统，文本编辑程序、调试程序、汇编程序、编译程序、数据库管理系统、文字处理系统、视窗软件、网络软件以及其他各种应用程序等。较低层的软件（如操作系统、汇编程序等）与硬件密切相关，而用户在使用高级语言或第四代语言编写程序时基本上已与硬件的实现无关，但是硬件的结构和性能对程序处理的速度影响极大。

计算机软件和硬件在逻辑功能上是等效的，即某些操作可以用软件实现也可以用硬件实现，因此软硬件之间没有固定不变的分界线，而是受实际应用的需要以及系统性能价格比所支配。从用户来看，机器的速度、可靠性、可维护性是主要的硬件技术指标。具有相同功能的计算机系统，其软/硬件之间的功能分配可以有很大差异。回顾计算机的发展历史，在其早期，由于硬件昂贵，所以计算机的硬件比较简单，尽量让软件完成更多的工作，但是随着组成计算机的基本元器件的发展，价格不断下降，性能不断提高，因而硬件成本不断下降。与此同时，随着应用的不断发展，软件成本在计算机系统中所占的比例不断上升，这就造成了软、硬件之间的分界面的推移，即某些由软件完成的工作交给硬件去完成，同时还提高了计算机实际运行速度。

本章主要阐述计算机硬件的基本知识，主要内容包括：计算机系统概述、数据的计算机表示、计算机算术运算和逻辑运算、计算机基本结构、指令系统等。程序员级的考试大纲要求的范围是：数制及其转换、机内代码、算术运算和逻辑运算、计算机的体系结构和主要部件、指令系统等。

1.2　知识点与难点

1.2.1　数制及其转换

1. 数制介绍

数制是由基数 r 和 r 个不同的数组成。任意数用 r 进制表示为：

$$N=(R_nR_{n-1}\cdots R_1R_0R_{-1}\cdots R_{-m})r$$
$$=R_n*r^n+R_{n-1}*r^{n-1}+\cdots+R_1*r^1+r_0+R_{-1}*r^{-1}+\cdots+R_{-m}*r^{-m}$$

上式称为数 N 在 r 进制中的按权展开式。

2. 数制间的转换

（1）从十进制向 r 进制转换

1 个十进制数转换为 r 进制数分两步进行：把该数的整数部分和小数部分分别转换成对应的 r 进制数，然后把这两部分加起来。

算法 1：除 r 取余法（把十进制整数 I 转换成对应的 r 进制整数）

① k:=1；

② 把 I 除以 r 得到商 Q 和余数 R_k；

③ 若 Q=0 则转到④，否则 I:=Q，k:=k＋1，并转到②；

④ 把 R_k 按 k 从大到小排列起来即为所求结果。

算法 2：乘 r 取整法（把十进制小数 F 转换成 r 进制小数）

① k:=1；

② F 乘以 r 得到整数部分 R_k 和小数部分 F′；

③ 若 F′ =0 或 k 达到了规定的精度位数，则转到④，否则 F:=F′，k:=k＋1，并转到②；

④ 把 R_k 按 k 从小到大排列，R_1 是最靠近小数点的位。

（2）二进制、八进制和十六进制之间的相互转换

因为 2^3=8，2^4=16，所以每位八进制数对应着 3 位二进制数，每位十六进制数对应着 4 位二进制数，二进制数向八进制数转换的方法：从小数点开始，分别向左、向右每 3 位二进制数为一组，若不够 3 位，则补 0；每一组用对应的八进制数表示。

八进制数向二进制数转换的方法：从小数点开始，把每一位八进制数转换为对应的 3 位二进制数。

二进制数和十六进制数之间的转换与此类似，每组的二进制位数是 4 位。

（3）特殊方法

因为八进制数与二进制数之间的转换简单，十进制数与二进制数之间的转换常用八进制数为中介。

如果一个分数的分母为 2 的整数次幂，则可以通过移动相应小数点位数进行转换。

从十进制数向八进制、十六进制数转换可以先转换为二进制数，然后再转换为八进制数、十六进制数。

（1）十进制整数转换成二进制数。

（2）十进制小数转换成二进制数。

1.2.2 机内代码

掌握机内代码、算术运算和逻辑运算等计算机基础知识。

1. 原码、补码、反码、移码（见表 1.1）

表 1.1

定点		运算		真值范围	正负符号表示	正负零之分
		$0\leqslant X\leqslant 2^{n-1}$（整） $0\leqslant X<1$(小)	$0\leqslant X\leqslant 2^{n-1}$(整) $0\leqslant X<1$(小)			
原码	整数	X	$2n-1-X$	$-(2^{n-1}-1)\leqslant X\leqslant 2^{n-1}-1$	0/1	[+0]=00…0 [−0]=10…0
	小数	X	$1-X$	$-(1-2^{-(n-1)})\leqslant X\leqslant(1-2^{-(n-1)})$		
补码	整数	X	2^n+X	$-(2^{n-1}-1)\leqslant X\leqslant 2^{n-1}-1$	0/1	[0]=00…0
	小数	X	$2+X$	$-1\leqslant X\leqslant 1-2^{-(n-1)}$		
反码	整数	X	$(2^n-1)+X$	$-(2^{n-1}-1)\leqslant X\leqslant 2^{n-1}-1$	0/1	[+0]=00…0 [−0]=11…1
	小数	X	$(2-2^{-n-1})+X$	$-(1-2^{-(n-1)})\leqslant X\leqslant(1-2^{-(n-1)})$		
移码	整数	$2^{n-1}+X$		$-(2^{n-1}-1)\leqslant X\leqslant 2^{n-1}-1$	1/0	[0]=00…0
	小数	$1+X$		$-1\leqslant X\leqslant 1-2^{-(n-1)}$		

2. 定点数与浮点数的机内表示

计算机内所有的数据类型分为两类：数值数据和符号数据，它们都是以二进制的形式存储和处理的。

（1）定点数据表示方法

小数点的位置固定不变，根据其所在位置的不同，分别表示整数和纯小数。

定点整数的格式：$N=b_sb_mb_{m-1}\cdots b_1b_0$，小数点在 b_0 右边，b_s 为符号位，b_m 为最高数据位，其取值范围为$|N|\leqslant 2^m-1$。

定点小数的格式：$N=b_sb_{-1}b_{-2}\cdots b_{-n}$，小数点在符号位 b_s 与最高位 b_{-1} 之间，其取值范围为$|N|\leqslant 1-2^n$。

（2）浮点数据表示方法

浮点数据表示成如下形式：

Ms	E	M

其中，Ms 是尾数的符号位，0 表示正号，而 1 表示负号；E 表示阶码，通常用移码表示；M 是定点小数形式的尾数，一般用补码或原码表示。

同一个数可以有不同的浮点表示形式，阶码的大小可以调节数值中小数点的位置。为提高数据的表示精度，可对浮点数进行规格化。规定浮点数的尾数部分用纯小数形式表示，当尾数值不为 0 时，其绝对位大于或等于 0.5，对于不符合这一规定的浮点数，可采用改变阶码的大小并同时左右移尾数的办法来满足这一规定。

3. ASCII 码及汉字编码

（1）ASCII 码

ASCII 码是美国标准信息交换码，它采用 7 位二进制数进行编码，一共有 128 个不

同的字符。这128个字符又可分为两类：可显示/打印字符95个和控制字符33个。7位的ASCII码也用1个字节来表示。最高1位没有使用，通常填0，也可以把它用作校验位或者用来扩展字符集。

（2）EBCDIC码

扩展的二/十进制交换码。采用8位编码来表述1个字符，共可以表示256个不同的字符，常用于IBM大型机中。

（3）汉字的编码与输入

GB2312共收集了常用汉字6763个：一级汉字3755个，二级汉字3008个。编码方法中主要可以分为3类：数字编码、拼音编码和字型码。

数字编码中，每一个汉字都分配给一个惟一的数字代码，用以代表该汉字。国际区位码、电报码都属于该类。常用的是国际区位码（又简称国际码或区位码）。拼音码中，用每个汉字的汉语拼音符号作为汉字的输入编码，这种编码很容易学会使用，成为最常用的输入方法。但是汉字同音字太多，重码率高，所以输入速度很难提高。字形码中，以汉字的形状特点为每个汉字进行编码，最受欢迎的一种字形编码方法是五笔字型编码，其思想是依据汉字的笔画特征，将基本笔画分为5类并分别赋以代号，根据汉字的结构特征把汉字分为4种字型分别赋以代号，并把使用频率较高的笔画结构和基本笔画一起作为汉字的构件。汉字的五笔字型编码就是依据其组成部件和结构特征进行编码的，能达到很高的输入速度。

（4）汉字的存储

汉字以内部编码（简称内码）的形式存放，以连续的两个字节来表示。为了和机内编码（ASCII码）相区别，这两个字节的最高位置1，两字节内码可表示128×128=16384个汉字。汉字内码是计算机在处理汉字信息时采用的机内代码。与此对应，通常又把汉字的输入编码称为外码。

（5）汉字的输出

汉字的输出是指输出汉字的点阵字型码：在显示器上显示汉字或打印机输出汉字。

汉字的字型点阵是以点阵形式描述的汉字字形代码，其点阵的密度决定了汉字的美观清晰度。普通的点阵为16×16，更高的有24×24，32×32，64×64等。汉字点阵码需要占用大量的存储空间，以普通16×16点阵为例，每个汉字占用16×16位=32 Byte，两级汉字共6763个，共占用6763×32=256 KB。

4. 奇偶校验码、海明校验码和循环冗余码（CRC）

计算机系统在运行时，各个部件之间经常需要进行数据交换，为了保证数据在传输过程中正确无误，不仅需要设计高可靠性的硬件电路，同时还必须引入某种差错检查机制，对数据信息进行校验以监测是否有数据传送错误。

常用的校验码有奇偶校验码、海明校验码和循环冗余校验码。

（1）奇偶校验码

奇偶校验码通过增加1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）来检测数据代码中奇数位出错的编码。因为其利用的是编码中1的个数的奇偶性，所以不能发现偶数位错误。校验位的添加方法可以有3种方

式：水平奇偶校验码、垂直奇偶校验码和水平垂直奇偶校验。

水平奇偶校验码对每一个数据的编码添加校验位。

垂直奇偶校验码中，为一组数据的相同位添加 1 个校验位，形成该组数据的 1 个奇（偶）校验码。

水平垂直奇偶校验中，先对一组数据进行垂直奇（偶）校验，所得结果（连同该垂直校验码）再添加 1 为水平校验位。

（2）海明校验码

海明校验码由贝尔实验室的 Richard Hamming 提出，也是利用奇偶校验来检错和纠错的。该校验码在数据位之间插入 k 个校验位，扩大数据编码的码距，从而检测出错误，并纠正它。

若能纠正 1 位错，k 个校验位可以有 2^k 个编码，其中 1 个用以表示数据无差错，而剩下 2^k-1 个编码则可以用于指示哪一位数据出错。由于 n 个数据位和 k 个校验位都可能出错，所以 k 必须满足：

$$2^k-1 \geqslant n+k$$

海明码的编码规则：

设 k 个校验位为 P_k，P_{k-1}，…，P_1，n 个数据位为 D_{n-1}，D_{n-2}，…，D_0，产生的海明码为 H_{k+n}，H_{k+n-1}，…，H_1，则有：

- P_i 在海明码的第 2^{i-1} 位置，数据位依次从低到高占据海明码中剩下的位置。
- 海明码的任一位都是由若干校验位来检验的。其对应关系为：被校验的海明位的下标等于所有参与校验该位的校验位的下标之和，校验位由其自身来校验。

（3）循环冗余校验码

循环冗余码利用生成多项式为 k 个数据位产生 r 个校验位进行编码，编码长度为 n+k=r，所以又称为（n，k）码。

① 编码规则

用 $C(x)=C_{k-1}C_{k-2}\cdots C_0$ 表示 k 个数据位，把 C(x)左移 r 位，给校验位空出 r 位来。

给定一个 r 阶生成多项式 g(x)，可求出校验位表达式 r(x)。

$$\frac{C(x)\times 2^r}{g(x)}=q(x)+\frac{r(x)}{g(x)}$$

在循环冗余编码过程中，四则运算采用模 2 运算，即不考虑借位和进位。

$C(x)\times 2^r+r(x)$为所求 n 位循环冗余码，它可以被 g(x)整除。

校验数据时用 n 位循环冗余码除以 g(x)，余数为 0 时数据正确，否则根据余数的值查出差错位。

② 运算规则

加减法规则：0±0=0，0±1=1，1±0=1，1±1=0。没有进位和借位。

乘法规则：利用模 2 加求部分积之和，没有进位。

除法规则：利用模 2 减求部分余数，没有借位，余数最高位为 1 则商 1，否则商 0；当部分余数的位数小于除数时，该余数为最后余数。

③ 常用的生成多项式

生成多项式应满足以下条件：

- 任何一位发生错误，$C(x)\times 2^r+r(x)$除以 g(x)的余数都不为 0；
- 余数继续作模 2 除，余数应循环。

若能纠正一位错，则不同的码位错不能有相同的余数。且给定生成多项式后，余数与出错位之间的对应关系应保持不变，与编码无关。

④ 通信与网络中常用的循环冗余

在数据通信与网络中，k 相当大，由很多数据位构成一帧，采用循环冗余码产生 r 位的校验位。它只能检测出错误，而不能纠正错误。

一般情况下，r 位生成多项式产生循环冗余码可检测出所有双错、奇数位错、突发长度小于等于 r 的突发错、$(1-2^{-(r-1)})$的突发长度为 r+1 的突发错和$(1-2^{-r})$的突发长度大于 r+1 的突发错。

（1）原码、反码、补码、移码。

（2）海明校验码。

（3）循环冗余校验码。

1.2.3 算术运算和逻辑运算

1. 计算机中的二进制数运算方法

（1）定点数运算

① 定点数加法、减法运算

定点数的加法和减法运算常使用补码。对于定点小数的运算规则如下：

$$\text{加法：}[X+Y]_{补}=\{[X]_{补}+[Y]_{补}\}\bmod 2$$

$$\text{减法：}[X-Y]_{补}=\{[X]_{补}+[-Y]_{补}\}\bmod 2$$

当运算结果超过定点数的表示范围时，则产生了溢出。常用的溢出检测机制主要有两种：

进位判决法：令 C_{n-1} 表示次高位向最高位（符号位）的进位，C_n 则表示符号位的进位，$C_n\oplus C_{n-1}=1$ 时溢出。

双符号位判决法：采用两位二进制位表示符号位：00 表示正号，11 表示负号，根据运算结果的符号位判定其是否溢出：00，11 表示无溢出，01，10 表示溢出。

② 定点数乘法运算

做定点数的乘法运算，常使用原码一位乘法来求两个定点数的乘积。

运算规则：

- 乘积的符号位等于乘数和被乘数的符号位进行异或的值；
- 乘积的值等于两数绝对值之积，即乘数和被乘数的绝对值进行移位相加。

X：被乘数，Y：乘数，P：部分积，flag：判别位，count：计数。

算法：

S1：P=0；count=0；flag=0；

S2：P 和 Y 一起右移一位（P 为高位部分，Y 为低位部分），P 的最高位补 0，P 的最低位移入 Y 的最高位，Y 的最低位移入到 Flag 中；

S3：若 flag=1 则 P=P+X，否则 P 不变；

S4：count=count+1，若 count 大于 Y 的位数则运算停止，否则转到第 S2 步。

P 和 Y 的内容是所求乘积的绝对值，其中 P 存放积的高位，Y 存放积的低位。

③ 定点数除法运算

定点数除法通常也用原码进行。常用加减交替法求两个数 X 和 Y 的商。

运算规则：

- 商的符号位为两数的符号位进行异或运算的结果；
- 商的数值为两数的绝对值部分进行相除的结果。

算法：

S1：R=X，counter=0；

S2：R=R－Y；若 R 为正则商 1，否则商 0；

S3：R 左移一位，counter=counter+1；

S4：若上一次商为 1，则 R=R－Y，否则 R=R+Y；

S5：R 为正则商 1，否则商 0；

S6：若 R=0 或者 counter≥N 则结束，否则转到第 S3 步。

（2）浮点数运算

在进行浮点运算之前，应首先对浮点数进行规格化。

① 浮点数的加减运算：设有浮点数 $X=M\cdot 2^i$，$Y=N\cdot 2^j$，求 X±Y 的运算过程如下：

S1：对阶：令 K=|i－j|，把阶码小的数的尾数右移 K 位，其阶码加上 K。当右移尾数时，若尾数用补码表示则符号位参加移位；若是原码，则符号位不参加移位，尾数最高位补 0。

S2：尾数进行加、减运算。

S3：规格化处理：若结果不是规格化的数，则需要对其进行规格化。当尾数溢出时，向右规格化，阶码加 1；当尾数用补码表示，尾数的高位与符号位相同时，向左规格化。

S4：舍入操作：常用的舍入方法有两种，第 1 种为 0 舍 1 入法，即当向右移规格化时，若移掉的最高位为 1，则在尾数末位加 1，否则舍去；第 2 种为“恒 1”法，即不管移走的数为何值，尾数最末位恒置 1。

S5：溢出判断：以阶码为准。若阶码上溢，结果溢出；若阶码下溢，结果为 0；否则结果无溢出。

② 浮点数的乘除运算：浮点数相乘，其乘积的阶码为两数阶码相加，积的尾数为两尾数相乘；浮点数相除，其商的阶码为两阶码之差，商的尾数为两尾数相除。其结果都需要进行规格化处理，同时还需要判断其阶码是否溢出。

2. 逻辑代数的基本运算和逻辑表达式的化简

（1）基本的逻辑运算

① 与（AND）运算：又称为逻辑乘运算，其运算符号用 AND、∩、∧、• 表示。两个变量的与运算的运算规则：当 A，B 中任一变量为 0 时，其运算结果为 0。

② 或（OR）运算：又称为逻辑加运算，其运算符号为 OR、∪、∨、＋。两个变量的或运算规则：当 A，B 中任何一个为 1 时，其运算结果为 1。

③ 非（NOT）运算：又称为逻辑求反运算。通常用 $\overline{A}$ 表示对变量 A 进行求反。其运算规则：$\overline{1}=0$，$\overline{0}=1$。

④ “异或”运算：又称为半加运算，其运算符号为 XOR 或⊕。异或运算是一种复合逻辑运算，可以利用基本逻辑运算表示如下：

$$A \oplus B=\overline{A} \cdot B+A \cdot \overline{B}$$

（2）常用的逻辑运算公式

① 交换律

$$A+B=B+A \qquad A \cdot B=B \cdot A$$

② 结合律

$$A+(B+C)=(A+B)+C \qquad A \cdot B \cdot C=(A \cdot B) \cdot C$$

③ 分配律

$$A \cdot (B+C)=A \cdot B+A \cdot C \qquad A+(B \cdot C)=(A+B) \cdot (B+C)$$

④ 反演律

$$\overline{A+B}=\overline{A} \cdot \overline{B} \qquad \overline{A \cdot B}=\overline{A}+\overline{B}$$

⑤ 重叠律

$$A+A=A \qquad A \cdot A=A$$

⑥ 互补律

$$A+\overline{A}=1 \qquad A \cdot \overline{A}=0$$

⑦ 0，1 律

$$0+A=A \qquad 0 \cdot A=0$$

$$1+A=1 \qquad 1 \cdot A=A$$

⑧ 对合律

$$\overline{\overline{A}}=A$$

⑨ 常用公式

$$AB+A\overline{B}=A \qquad A+AB=A$$

$$A+\overline{A}B=A+B \qquad AB+\overline{A}C+BC=AB+\overline{A}C$$

$$\overline{A \oplus B}=\overline{A} \oplus B=A \oplus \overline{B}$$

（3）逻辑表达式及其化简

① 逻辑表达式与真值表

逻辑表达式是用逻辑运算符把若干逻辑变量连接在一起表示某种逻辑关系的表达式，一个逻辑函数通常有多种不同的表达式。常用一张表来描述一个逻辑函数与其变量之间的关系，即把变量的各种取值情况都列出来，这张表称为真值表。当变量不太多时，可利用真值表证明某些逻辑恒等式。

② 逻辑表达式的化简

利用基本逻辑运算规律和一些常用的逻辑恒等式对逻辑表达式进行合并项、吸收项、配项、消去项等操作来化简。

（1）定点数乘法的运算。

（2）浮点数的加减运算。

（3）运用逻辑运算公式化简复杂的逻辑表达式。

1.2.4　计算机的体系结构和主要部件

1. 中央处理器 CPU

CPU 负责大部分的信息处理操作，它主要有 3 大部分：寄存器组（register block）、算术逻辑单元（Arithmetic Logic Unit，ALU）和控制单元（Control Unit，CU）。寄存器组在指令执行过程中存放操作数和中间数据，运算器完成指令中的运算，CU 控制运算器和寄存器组正确地完成操作。还有内部总线和外部总线，前者为寄存器之间以及寄存器和运算器之间提供高速的数据通路；后者则把 CPU、存储器和 I/O 设备连接起来，完成数据交换。

（1）寄存器组

CPU 中寄存器可分为两类：用户可见的寄存器和状态控制寄存器。

① 用户可见的寄存器

用户能够通过机器语言来访问这些寄存器。合理地使用这些寄存器可以减少对主存的访问次数，从而提高程序的执行速度。

按其功能可把寄存器分为 4 类：通用寄存器，它的用途由程序员编程决定，例如可以用来存放操作数；数据寄存器，用来存放数据，不能对操作数的地址进行计算；地址寄存器，用来存放操作数的地址，它还可以分为段地址寄存器、间址寄存器和堆栈寄存器；标志寄存器，它是由 CPU 根据运算的结果而设立的一些指示位，以反映该次运算的某些特征，比如结果为正、负、零、溢出等，并可以用作分支跳转的依据，一般而言，标志寄存器允许程序员进行读操作，但不允许对其直接进行写操作。

② 状态控制寄存器（CSR）

状态控制寄存器用来控制 CPU 的操作。通常程序员看不见，但也有某些寄存器在某种特定的状态下可以由某些特权指令进行访问。常见的状态控制寄存器有：程序计数器、指令寄存器、存储器地址寄存器、存储器缓冲寄存器以及程序状态字。

在某些 CPU 中还可能有一些其他类型的状态控制寄存器，例如中断向量寄存器，指向其他状态信息、内存区域的指针寄存器，用以支持虚拟存储器的系统中的页面寄存器以及进行控制 I/O 操作的寄存器等。

（2）运算器

运算器的功能是对数据进行算术以及逻辑运算。计算机系统的其他部分把待处理的数据输入到运算器，由运算器进行处理，然后把运算结果输出，同时设置标志位。

（3）控制器

控制器负责控制整个计算机系统的运行，读取指令寄存器、状态控制寄存器以及从外部来的控制信号，发布外控制信号控制 CPU 与存储器、I/O 设备进行数据交换，发布内控制信号控制寄存器间的数据交换，控制运算器完成指定的运算功能以及管理其他的 CPU 内部操作。

① 控制器的基本功能

控制器的基本功能有时序控制和执行控制。控制器使 CPU 按一定的时序关系执行一系列的微操作,从而完成程序规定的动作。控制器的输入信号有时钟信号、指令寄存器标志位和控制总线的控制信号。控制器的输出信号包括 CPU 内的控制信号、发往控制总线的控制信号。

- 时钟信号

控制器根据由时钟电路产生的时钟信号进行定时，控制各种操作按指定的时序进行。

- 指令寄存器

控制器需要完成取指令、分析指令和执行指令的操作。控制器根据程序计数器中的内容（指令地址）从存储器中取出该指令，然后对指令进行译码，确定所需执行的微操作，最后控制器根据分析的结果发布一系列的控制信号控制各部件完成规定的操作，同时产生下一条指令的地址。

- 中断控制逻辑

计算机系统通常都提供了中断机制，允许某一事件的发生（如 I/O 设备提供服务请求）可以中止 CPU 正在执行的程序，转去对该事件进行处理，处理之后再返回被中止处继续执行。中断机制的主要作用就是提高 CPU 的处理效率，同时使 CPU 能够及时地响应各种预先不知的异常事务并处理，还可以分时操作等。

如果系统允许响应中断请求，那么在每条指令执行完成后，控制器都要检查是否有中断发生：如果没有中断请求信号，控制器则转去取下一条指令；如果有中断事件发生，则暂停执行当前程序并保存现场状态信息（如下一条指令的地址、相关寄存器的内容等），然后转去执行中断服务程序。当完成中断服务程序后，再恢复以前保存的现场信息，从中止处继续执行。

通常把 CPU 中断处理过程分为两个阶段：中断响应过程和中断服务过程。中断响应过程关闭中断，禁止 CPU 响应新的中断请求，保存断点信息，把程序的断点即 PC 的