

计算机体系结构知识

适用班级：软件设计师、网络工程师

主讲：刘琳芳

网址：[www.bitpx.com](http://www.bitpx.com)

QQ:957052659

分值说明：

软件设计师考5-8分(早上试题)

网络工程师考3-6分(早上试题)

比特培训中心

贵州·贵阳

2018年09月

目录

[第1章 计算机科学基础 1](#_Toc493850948)

[1.1 数制及其转换 1](#_Toc493850949)

[1.1.1 计算机中常用的数制 1](#_Toc493850950)

[1.1.2 不同数制间的转换 1](#_Toc493850951)

[1.2 数据的表示 2](#_Toc493850952)

[1.2.1 数的机器码表示 2](#_Toc493850953)

[1.2.2 定点数和浮点数 4](#_Toc493850954)

[1.3 逻辑运算和算术运算 7](#_Toc493850955)

[1.3.1 逻辑运算 7](#_Toc493850956)

[1.3.2 算术运算 8](#_Toc493850957)

[1.3.3 溢出判别 8](#_Toc493850958)

[第2章 计算机体系结构 9](#_Toc493850959)

[2.1 计算机系统结构概述 9](#_Toc493850960)

[2.2 存储系统 9](#_Toc493850961)

[2.2.1 主存(MainMemory) 10](#_Toc493850962)

[2.2.2 高速缓存Cache 10](#_Toc493850963)

[2.2.3 主存与Cache的地址映射 11](#_Toc493850964)

[2.2.4 虚拟存储器 13](#_Toc493850965)

[2.2.5 硬磁盘存储设备 14](#_Toc493850966)

[2.2.6 相联存储器 16](#_Toc493850967)

[2.3 指令系统 16](#_Toc493850968)

[2.3.1 指令格式 17](#_Toc493850969)

[2.3.2 操作码格式 17](#_Toc493850970)

[2.3.3 地址码格式 17](#_Toc493850971)

[2.3.4 扩展操作码 17](#_Toc493850972)

[2.3.5 指令长度 18](#_Toc493850973)

[2.3.6 指令和数据的寻址方式 18](#_Toc493850974)

[2.3.7 CISC和RISC的区别 19](#_Toc493850975)

[2.3.8 指令类型 20](#_Toc493850976)

[2.4 中央处理器 21](#_Toc493850977)

[2.4.1 指令执行的过程 21](#_Toc493850978)

[2.4.2 CPU指令执行流程图 22](#_Toc493850979)

[2.4.3 CPU的组成 22](#_Toc493850980)

[2.5 中断系统 24](#_Toc493850981)

[2.6 流水线操作 25](#_Toc493850982)

[2.6.1 指令流水线 25](#_Toc493850983)

[2.6.2 流水线处理机的主要性能 25](#_Toc493850984)

[2.7 总线结构 27](#_Toc493850985)

[2.7.1 总线的定义与分类 27](#_Toc493850986)

[2.7.2 系统总线标准 29](#_Toc493850987)

[2.7.3 外总线标准 29](#_Toc493850988)

[2.8 输入输出系统 30](#_Toc493850989)

[2.9 多处理机与并行处理 32](#_Toc493850990)

[第3章 计算机可靠性与系统性能基础知识 33](#_Toc493850991)

[3.1 计算机可靠性与系统性能 33](#_Toc493850992)

[第4章 差错控制技术 34](#_Toc493850993)

[4.1 奇偶校验码 34](#_Toc493850994)

[4.2 海明码 35](#_Toc493850995)

[4.3 CRC码(循环冗余校验码) 36](#_Toc493850996)

# 计算机科学基础

## 数制及其转换

### 计算机中常用的数制

在计算机中，常涉及到的数制有二进制(基数为2)、八进制(基数为8)、十进制(基数为10)和十六进制(基数为16)。

#### 十进制数

基数为10，逢十进一，数字位的表示从0到9，进制表示符号为D，如“12.3”按照位权展开：(12.3)10＝1×101＋2×100+3×10-1＝12.3D。

采用**位置计数法**，十进制按位权展开规律如下：

* 整数部分

某数位对应的位权值是当前位值乘上以基数为底的幂，幂中指数为当前数位到小数点之间的数值个数(正数且不包含当前位)，即位权值=位值×基数离小数点的数位个数，如(634.57)10中整数部分位权值依次是：6×102、3×101、4×100。

* 小数部分

某数位对应的位权值是当前位值乘上以基数为底的幂，幂中指数为当前数位到小数点之间的数值个数(负数且包含当前位)，即位权值=位值×基数-离小数点的数位个数，如(634.57)10中小数部分位权值依次是：5×10-1、7×10-2。

* 合并

将整数部分和小数部分的位权值相加即为原数的位权展开式，如(634.57)10的位权展开式为6×102+3×101+4×100+5×10-1+7×10-2，即(634.57)10=6×102+3×101+4×100+5×10-1+7×10-2=634.57D。

位置计数法不仅仅适用于十进制数，同样适用于二进制、八进制和十六进制，只需要将幂中的底数改成相应的数制基数即可。

#### 二进制数

基数为2，数字位符号只有0和1，逢二进一，进制表示符号为B，如“11.1”按照位权展开：(11.1)2＝1×21＋1×20+1×2-1＝11.1B。

#### 八进制数

基数为8，逢八进一，数字位的表示从0到7，进制表示符号为O，如“11.1”按照位权展开：(11.1)8＝1×81＋1×80+1×8-1＝11.1O

#### 十六进制数

基数为16，逢十六进一，数字位的表示从0到9，10用A表示，以此类推11为B，12为C，13为D，14为E，15为F，进制表示符号为H，如“4FD”按照位权展开：(4FD)16＝4×162+15×161+13×160＝4FDH

### 不同数制间的转换

各种数制之间的相互转化知识在软考中频率很高，请一定学会这部分知识。

#### 二、八、十六进制转换为十进制

其他进制转化为十进制数使用的是**位置计数法**。

* 二进制转10进制：(11.1)2=11.1B=1×21＋1×20+1×2-1=2+1+0.5=(3.5)10
* 八进制转10进制：(11.1)8=11.1O=1×81＋1×80+1×8-1=8+1+0.125=(9.125)10
* 十六进制转10进制：(4FD)16=4FDH=4×162+15×161+13×160=1024+240+13=(1277)10

#### 十进制转其他进制

十进制转换为其他进制，将十进制数及其商连续除目标进制的基数(除数为0时停止)，最后将余数从下往上连续写出来便构成十进制数对应的其他进制数，过程如图1.1所示。在图中，127D=1111111B，127D=177O，127D=7FH。

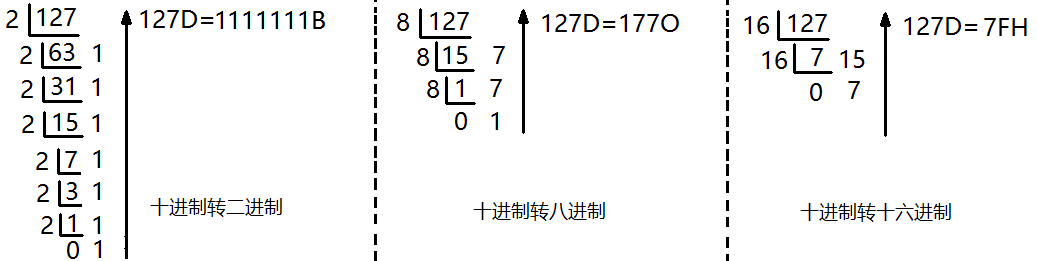


图1.1十进制转二进制、八进制和十六进制

#### 八进制转换为二进制

* 用“1位拆分为3位”的方法，如(7.2)8＝111.010B=(111.010)2

#### 十六进制转换为二进制

* 用“1位拆分为4位”的方法，A2B.CH=1010 0010 1011.1100 B=(1010 0010 1011.1100)2

#### 二进制转换为八进制和十六进制

二进制转换为八进制是八进制转换为二进制的逆过程，以二进制中的小数点为分界点分别向左和向右分组，每三位一组，整数部分分组位数不够时在分组前补0，小数部分分组位数不够，在分组后补零。如10111.01B转换为八进制，先分组并补零后得(带下划线的为补零位)：**0**10111.01**0**B，再转换为八进制为：27.2O。

二进制转换为十六进制的过程与二进制转换为八进制的过程雷同，不同的是4位为一组。如10111.01B转换为十六进制，先分组并补零后得：**000**10111.01**00**B，再转换为十六进制为：17.4H。

#### 各种进制小范围数据对照表

如表1.1所示为十进制、二进制、八进制、十六进制基数对照表，这些数据只是小范围数据，学员必须牢牢记住。

表1.1十进制、二进制、八进制、十六进制基数对照表

|  |  |  |  |
| --- | --- | --- | --- |
| 十进制 | 二进制 | 八进制 | 十六进制 |
| 0D | 0000B | 0O | 0H |
| 1D | 0001B | 1O | 1H |
| 2D | 0010B | 2O | 2H |
| 3D | 0011B | 3O | 3H |
| 4D | 0100B | 4O | 4H |
| 5D | 0101B | 5O | 5H |
| 6D | 0110B | 6O | 6H |
| 7D | 0111B | 7O | 7H |
| 8D | 1000B | 10O | 8H |
| 9D | 1001B | 11O | 9H |
| 10D | 1010B | 12O | AH |
| 11D | 1011B | 13O | BH |
| 12D | 1100B | 14O | CH |
| 13D | 1101B | 15O | DH |
| 14D | 1110B | 16O | EH |
| 15D | 1111B | 17O | FH |
| 16D | 10000B | 20O | 10H |

## 数据的表示

各种数值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，小数点则隐含表示而不占位置。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有符号数之分。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。对于有符号数，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，有符号的机器数可采用原码、反码和补码等不同的码制方法，机器数的这些编码方法称为码制。

### 数的机器码表示

这里假设机器字长为n位，即有符号数用n个二进制位表示，最高位为符号位(0表示正数，1表示负数)，其余n-1位为数值的绝对值。**以下的码制讲解都假设机器字长为8位，其中左侧1位为符号位。**

* 机器字长：机器字长也就是运算器进行定点数运算的字长，通常也是CPU内部数据通路的宽度。即字长越长，数的表示范围也越大，精度也越高。机器的字长也会影响机器的运算速度。倘若CPU字长较短，又要运算位数较多的数据，那么需要经过两次或多次的运算才能完成，这样势必影响整机的运行速度。

#### 原码：符号位用0表示正数，1表示负数，其余位不变。

[+1]原=0 0000001 [-1]原＝1 0000001 [+127]原＝0 1111111 [-127]原＝1 1111111

[+45]原=0 0101101 [-45]原=1 0101101 [+0.5]原=0 1000000 [-0.5]原=1 1000000

[+0]原＝00000000 [-0]原＝10000000

* X为纯整数：[X]原=X真 (0≤X＜2n-1)

[X]原=2n-1- X真 =2n-1 +| X真| (-2n-1＜X≤0)

* X为纯小数：[X]原=X 真 (0≤ X＜1)

[X]原=1-X真 =1+| X真| (-1＜ X ≤0)

**注意：**纯整数约定小数点的位置在机器数的最低位之后，纯小数约定小数点在机器数的最高数值位之前(符号位之后)，且小数点隐含表示而不占位置，但在理解时要加上小数的位置理解，如(+127)原＝0 1111111理解为“0 1111111**.**”，(+0.5)原=0 1000000理解为“0**.**1000000”，但其中的“0**.**”中的“0”(表示正数)为符号位，(-0.5)原=1 1000000理解为“1**.**1000000”，其中的“1**.**”中的“1”(表示负数)为符号位。故计算机在实际存储时将纯小数中数值意义上的“0**.**”隐含而不存储的方式进行处理，但我们自己要注意理解。

#### 反码：正数的反码与原码一样，负数的反码是对它的原码(除符号位外)各位取反。

[+1]反=0 0000001 [-1]反＝1 1111110 [+127]反＝0 1111111 [-127]反＝1 0000000

[+45]反=0 0101101 [-45]反=1 1010010 [+0.5]反=0 1000000 [-0.5]反=1 0111111

[+0]反＝00000000 [-0]反＝11111111

* X为纯整数：[X]反=X真 (0≤X＜2n-1)

[X]反=(2n-1)+ X真 =(2n -1)-| X真 | (-2n-1＜X≤0)

* X为纯小数：[X]反=X 真 (0≤ X＜1)

[X]反=(2-2n-1)+ X真 =(2-2n-1)-| X真 | (-1＜ X ≤0)

#### 补码：正数的补码与原码一样，负数的补码是其反码尾部加1。

[+1]补=0 0000001[-1]补＝1 1111111 [+127]补＝0 1111111 [-127]补＝1 0000001  **[-128]补＝1 0000000**

[+45]补=0 0101101 [-45]补=1 1010011[+0.5]补=0 1000000 [-0.5]补=1 1000000

[+0]补＝00000000[-0]补＝00000000

* X为纯整数：[X]补=X真 (0≤X＜2n-1)

[X]补=2n+ X真 =2n -| X真 | **(-2n-1≤X≤0)**

* X为纯小数：[X]补=X 真 (0≤ X＜1)

[X]补=2+ X真 =2-| X真 | **(-1≤ X ≤0)**

**注意：在(-0)反＝1 1111111的基础上加1后结果为10 0000000(9位)，而此部分讲解是假设计算机字长为8位，所以10 0000000(9位)中的1就被扔掉而变成8位的0 0000000，故(-0)补＝0 0000000。记住：在补码表示中，0有唯一的编码：(+0)补与(-0)补都是0 0000000。**

#### 移码：移码表示法是在数X上增加一个偏移量来定义的，通常用于表示浮点数的阶码。如果机器字长为n，其中1位为符号位，规定偏移量为2n-1，则移码定义如下：

* X为纯整数：[X]移=2n-1+X(-2n-1≤X＜2n-1)
* X为纯小数：[X]移=1+X(-1≤X＜1)

这里还是假设某机器字长为8，其中1位为符号位，数据位为7位，即n=7，则+1、-1、+127、-127、+45、-45、+0.5、-0.5、+0、-0对应的移码表示如下：

[+1]移=28-1+1=129=1 0000001 b [-1]移=28-1-1=127=0 1111111 b [+127]移=28-1+127=255=1 1111111 b

[-127]移=28-1-127=1=0 0000001 b [+45]移=28-1+45=173=1 0101101 b [-45]移=28-1-45=83=0 1010011 b

[+0.5]移=1+0.5=1.5=1 1000000 b [-0.5]移=1-0.5=0.5=0 1000000 b [+0]移=[-0]移=1 0000000 b

注意：实际上在偏移2n-1情况下**，只要将补码的符号位取反便是相应的移码表示**；在移码中，0有唯一的编码：1 0000000。

**记住：在原码、反码、补码和移码表示中，补码和移码中的0都有唯一值。**

#### 原码、反码、补码和移码的作用

* 原码：对于计算机，加减乘除是最基础的运算，要设计的尽量简单。计算机辨别“符号位”显然会让计算机的基础电路设计变得十分复杂!于是科学家想出了**将符号位也参与运算的方法**。我们知道，根据运算法则减去一个正数等于加上一个负数，即:1-1=1+(-1)=0，所以机器可以只有加法而没有减法，这样计算机运算的设计就更简单了。

在运算中，原码进行加法运算是没有问题的，但是在减法运算中会出问题，如：

1D-1D=1D+(-1)D=[0 0000001]原+[1 0000001]原=[1 0000010]原=-2D，这个结果明显是错误的！

如果用原码表示，让符号位也参与计算，显然对于减法来说，结果是不正确的。这也就是为何计算机内部不使用原码表示一个数的原因。为了解决原码做减法的问题，出现了反码。

* 反码：使用反码计算1-1过程如下：

1D-1D=1D+(-1)D=[00000001]原+[10000001]原=[00000001]反+[11111110]反=[11111111]反=[10000000]原=-0D，发现用反码计算减法，结果的真值部分是正确的。而唯一的问题其实就出现在“0”这个特殊的数值上。虽然人们理解上+0和-0是一样的，但在这里0带符号是没有任何意义的，而且会有[00000000]原和[10000000]原两个编码表示0。

于是补码的出现，解决了0的符号和0的两个编码问题，以及原码和补码进行减法运算时出现的问题。

* 补码：使用补码计算1-1过程如下：

1D-1D=1D+(-1)D=[00000001]原+[10000001]原=[00000001]补+[11111111]补=[0 0000000]反=[0 0000000]原=0D，这样0用00000000表示，而以前出现问题的-0则不存在了，且在补码中0也只有一种唯一的表示形式。

假设机器字长为8位，使用补码可以用[10000000]补表示-128D:

(-1)D+(-127)D=[10000001]原+[11111111]原=[11111111]补+[10000001]补=[10000000]补，最终运算结果应为“[11 0000000]补”，但由于机器字长为8位，最左边的“1”被丢掉了，故只剩下[10000000]补。

-1-127的结果应该是-128，在用补码运算的结果中，[10000000]补就是-128。但是注意因为实际上是使用以前的-0的补码来表示-128，所以-128并没有原码和反码表示(对-128的补码表示[10000000]补算出来的原码是[00000000]原,这是不正确的)。

使用补码，不仅仅修复了0的符号以及0的两个编码的问题，而且还能够多表示一个最低数。这就是为什么8位二进制使用原码或反码表示的范围为[-127，+127]，而使用补码表示的范围为[-128，127]的原因。

因为机器使用补码，所以对于编程中常用到的32位int类型，可以表示范围是(最高位是符号位)：[-231，231-1]，使用补码表示时可以多保存一个最小值-231。假设机器字长为n位，则使用补码表示的最小的一个数是-2n-1。

* 移码：常用来比较大小，一般会把浮点数的阶码用移码表示。当把数值用移码表示出来可一眼看出它们的大小，这样很容易判断阶码的大小，移码可用于简化浮点数的乘除法运算。

总之，反码用来解决负数加法运算问题，将减法运算转换为加法运算，从而简化运算规则；补码解决负数加法运算正负零问题，弥补了反码的不足。反码与补码都是为了解决负数运算问题，跟正数没关系，因此，不管是正整数还是正小数，原码，反码，补码都全部相同。将补码符号位取反即得到相应的移码。

各种编码的表示范围：假设机器字长为3位，为A0A1A2，其中A0为符号位，则表1.2所示为A0A1A2能够使用原码、反码及补码表示的数值范围：

表.2A0A1A2能够表示的数值范围

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 真实值 | 二进制 原码表示 | 二进制 反码表示 | 二进制 补码表示 | 二进制 移码表示 |
| +0 | 0 00 | 0 00 | **0 00** | **100** |
| +1 | 0 01 | 0 01 | 0 01 | 101 |
| +2 | 0 10 | 0 10 | 0 10 | 110 |
| +3 | 0 11 | 0 11 | 0 11 | 111 |
| -0 | 1 00 | 1 11 | **0 00** | **100** |
| -1 | 1 01 | 1 10 | 111 | 011 |
| -2 | 1 10 | 1 01 | 110 | 010 |
| -3 | 1 11 | 1 00 | 101 | 001 |
| -4 | 已超出原码表达范围，无对应形式 | 已超出反码表达范围，无对应形式 | 100 | 000 |

上表中-4的编码表达形式可使用公式：[X]补=2n+ X真 =2n -| X真 |计算，

[-4]补=23+(-4)=23 -| 4|=4D=100B,

从表1.2可看出，A0A1A2能够表示的数值范围(补码表示)：100(-4)～011(+3).

仿照原码转换成补码和反码的过程再重复执行一次，即可还原成原码形式。

###### 历年试题

●(2016年上)如果“2x”的补码是“90H”，那么X的真值是(3)。

(3)A.72 B.-56 C.56 D.111

●(2011年上)原码表示法和补码表示法是计算机中用于表示数据的两种编码方法，在计算机系统中常采用补码来表示和运算数据，原因是采用补码可以 (4) 。

(4)A.保证运算过程与手工运算方法保持一致 B.简化计算机运算部件的设计

C.提高数据的运算速度 D.提高数据的运算精度

●(2010年上)若某整数的16位补码为FFFFH(H表示十六进制)，则该数的十进制值为 (20) 。

(20)A.0 B.-1 C.216-1 D.-216+1

●(2009年上)计算机中常采用原码、反码、补码和移码表示数据，其中，±0编码相同的是(5)。

(5)A.原码和补码 B.反码和补码 C.补码和移码 D.原码和移码

### 定点数和浮点数

#### 定点数的表示方法

其实这个概念之前也有提到一点，所谓定点定点数，即约定机器中所有数据的小数点位置是固定不变的数。例如真值X存储如图1.2所示：



图1.2定点格式

* 定点整数：纯整数，小数点位于最低位Xn的右边，但在计算机中被隐含而不占用实际的存储位置。
* 定点小数：纯小数，小数点约定位于X0和X1之间，但在计算机中“0**.**”被隐含而不占用实际的存储位置。

为了方便，一般只采取三种简单的约定，相应有三种类型的约定：无符号整数、带符号定点整数、带符号定点小数。

假设机器字长为4，各种码制下带符号数的表示范围如表1.所示。

表1.3 字长为4bit，各种码制表示定点小数和定点整数时数据表示范围

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 原码表示  (二进制) | 对应真实值(十进制) | 反码表示(二进制) | 对应真实值(十进制) | 补码表示(二进制) | 十进制真实值 | 移码表示(二进制) | 移码表示(十进制真实值) |
| 定点小数 | +0 | 0.000 | +0 | 0.000 | +0 | 0.000 | +0 |  |  |
| 最小正数 | 0 .001 | 2-3 | 0 .001 | 2-3 | 0 .001 | 2-3 |  |  |
| 最大正数 | 0.111 | 1-2-3 | 0.111 | 1-2-3 | 0.111 | 1-2-3 |  |  |
| -0 | 1.000 | -0 | 1.111 | -0 | 1.111 | -0 |  |  |
| 负数 绝对值最小小数 | 1.001 | -2-3 | 1.110 | -2-3 | 1.111 | -2-3 |  |  |
| 负数 绝对值最大小数 | 1.111 | -(1-2-3) | 1.000 | -(1-2-3) | 1.000 | -(1) |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 定点整数 | +0 | 0000. | +0 | 0000. | +0 | 0000. | +0 | 1000 | 0 |
| 最小正数 | 0001. | 1 | 0001. | 1 | 0001. | 1 | 1001 | 1 |
| 最大正数 | 0111. | 23-1=7 | 0111. | 23-1=7 | 0111. | 23-1=7 | 1111 | 23-1=7 |
| -0 | 1000. | -0 | 1111. | -0 | 0000. | -0 | 1000 | 0 |
| 负数 绝对值最小整数 | 1001. | -1 | 1110. | -1 | 1111. | -1 | 0111 | -1 |
| 负数 绝对值最大整数 | 1111. | -(23-1)=-7 | 1000. | -(23-1)=-7 | 1000. | -23=-8 | 0000 | -23=-8 |

假设机器字长为n，各种码制下带符号数的表示范围如表1.所示。

表1.4机器字长为n时各种码制表示的有符号数的范围

|  |  |  |  |
| --- | --- | --- | --- |
| 编码方式 | 表示定点整数时真实  数值的范围 | 表示定点小数时真实  数值的范围 | 不同数值数据的总个数 |
| 原码 | -(2n-1-1)～+(2n-1-1) | -(1-2-(n-1))～+(1-2-(n-1)) | 2n-1(正负视为同一个数) |
| 反码 | -(2n-1-1)～+(2n-1-1) | -(1-2-(n-1))～+(1-2-(n-1)) | 2n-1(正负视为同一个数) |
| 补码 | -(2n-1) ～+(2n-1-1) | -(1) ～+(1-2-(n-1)) | 2n |
| 移码 | -(2n-1) ～+(2n-1-1) | -(1) ～+(1-2-(n-1)) | 2n |

●(2018年下)采用n位补码(包含一个符号位)表示数据，可以直接表示数值 (3) 。

(3) A.2n B.-2n C.2n-1  D.-2n-1

●(2015年上)机器字长为n位的二进制数可以用补码来表示 (1)个不同的有符号定点小数。

(1)A.2n B.2n-1 C.2n-1 D.2n-1+1

●(2014年上)某机器字长为n，最高位是符号位，其定点整数的最大值为(2) 。

(2)A.2n-1 B.2n-1-1 C.2n  D.2n-1

#### 浮点数表示方法

由于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，因此引入了浮点数，浮点数是小数点位置不固定的数，它能表示更大范围的数。

在十进制数中，83.125可写成0.083125×103或0.0083125×104等。同理，一个二进制也可以写成多种表示形式，如1011.10101可以写成24×0.101110101、2５×0.0101110101等。由此可知，一个二进制数N可表示成:N=RE\*M，其中M称为尾数，通常为小数，E为指数，是一个整数，即阶码，R为基数，可采用2，8或16进制等，这种表示法称为浮点数表示法。

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



下图为阶码部分共占7位，其中阶符部分占1位，阶码数值部分占6位，尾数部分共占9位，其中数符部分占1位，尾数数值部分占8位，该浮点数尾数部分用原码和补码表示时的表示范围。

图1.3非规格化时浮点数尾数部分用原码和补码表示时的表示范围

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 尾数用**原码**，阶码用移码表示时范围(非规格化表示形式) | | |  | 尾数用**补码**，阶码用移码表示时范围(非规格化表示形式) | | |
|  | 阶码二进制及真实值 | 尾数二进制及真实值 | 该浮点数真实值 | 备注 | 阶码二进制及真实值 | 尾数二进制及真实值 | 该浮点数真实值 |
|  | 0 000000(0) |  | 0.0 | 阶码为00..0时，不管尾数为何值，表示机器0 |  |  |  |
| 正数 最小值 | 0 000001(1-26=-63) | 0.00000001(2-8) | 2-8\*2-63 |  | 0 000001(1-26=-63) | 0.00000001(2-8) | 2-8\*2-63 |
| 正数 最大值 | 1 111111(27-1-26=63) | 0.11111111 | (1-2-8)\*263 |  | 1 111111(27-1-26=63) | 0.11111111 | (1-2-8)\*263 |
| 负数 绝对值最小数 | 0 000001(1-26=-63) | 1.00000001(-2-8) | -2-8\*2-63 |  | 0 000001(1-26=-63) | 1.11111111(-2-8) | -2-8\*2-63 |
| 负数 绝对值最大数 | 1 111111(27-1-26=63) | 1.11111111  -(1-2-8) | -(1-2-8)\*263 |  | 1 111111(27-1-26=63) | 1.00000000  -(1) | -(1)\*263 |

●(2016年下)设16位浮点数，其中阶符1位、阶码值6位、数符1位、尾数8位。若阶码用移码表示，尾数用补码表示，则该浮点数所能表示的数值范围是\_\_(3)\_\_。

(3)A.(1- B.(1- C. -(1-~(1- D.-(1-~(1-

一个数的浮点表示不是唯一的，当小数点的位置改变时，阶码也随着改变，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，而表示的精度由尾数决定。**为了充分利用尾数来表示更多的有效数字，通常采用规格化浮点数。规格化就是将尾数的绝对值限定在[0.5，1]区间。当尾数用补码表示时，需要注意如下的问题：

* 如尾数M≥0，则规格化的尾数形式为M=0.1×××•••××，其中×可为0，也可以为1，即将尾数M的范围限定在区间[0.5，1)。
* 如尾数M＜0，则规格化的尾数形式为M=1.0×××•••××，其中×可为0，也可以为1，即将尾数M的范围限定在区间[-1，-0.5)。

**注意：**若不对浮点数的表示作出明确规定，同一个浮点数的表示就不是唯一的，例：十进制数11，如果尾数只用2位存储，则可以表示为0.11\*102，也可以表示为0.01\*103(由于只有2位存储，0.001被丢失了)，可见0.11\*102精度高，**浮点数的精度是由尾数的长度来确定的**。为了提高数据的表示精度，应该对不满足规格化的尾数进行左归或右归变成规格化的表示形式：

* 左规：如当尾数出现0.0×××•••××时，即数字左移一位,阶码减1，直到符合规格化表示式为止。
* 右规：当尾数出现1.×××•••××时，数字右移一位,阶码加1，直到符合规格化表示式为止。

图1.4浮点数尾数部分用原码和补码表示时的表示范围

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 尾数用**原码**，阶码用移码表示时范围(规格化表示形式) | | |  | 尾数用**补码**，阶码用移码表示时范围(规格化表示形式) | | |
|  | 阶码二进制及真实值 | 尾数二进制及真实值 | 该浮点数真实值 | 备注 | 阶码二进制及真实值 | 尾数二进制及真实值 | 该浮点数真实值 |
|  | 0 000000(0) |  | 0.0 | 阶码为00..0时，不管尾数为何值，表示机器0 |  |  |  |
| 正数 最小值 | 0 000001  (1-26=-63) | 0.10000000(2-1) | 2-1\*2-63 |  | 0 000001  (1-26=-63) | 0.10000000(2-8) | 2-8\*2-63 |
| 正数 最大值 | 1 111111  (27-1-26=63) | 0.11111111 | (1-2-8)\*263 |  | 1 111111  (27-1-26=63) | 0.11111111 | (1-2-8)\*263 |
| 负数 绝对值最小数 | 0 000001  (1-26=-63) | 1.10000000(-2-1) | -2-1\*2-63 |  | 0 000001  (1-26=-63) | 1.01111111  (-2-1-2-8) | (-2-1-2-8)\*2-63 |
| 负数 绝对值最大数 | 1 111111  (27-1-26=63) | 1.11111111  -(1-2-8) | -(1-2-8)\*263 |  | 1 111111  (27-1-26=63) | 1.00000000  -(1) | -(1)\*263 |

#### 工业化标准IEEE754

IEEE754是由IEEE(Institute of Electrical and Electronics Engineers)提出有关浮点数的工业标准。该标准表示形式如下：(-1)S2E(b0b1b2b3•••bp-1)，其中，(-1)S为该浮点数的数符，当S为0时表示正数，S为1时表示负数；E为指数(阶码)，用移码表示；(b0b1b2b3•••bp-1)为尾数，其长度为P位，**用原码表示**。图1.5为32位单精度浮点数的IEEE754标准格式(以32位二进制形式在计算机中存储)。其中S为1位，即0或1，E为8位(最高位为符号位)，尾数长度P为23位：

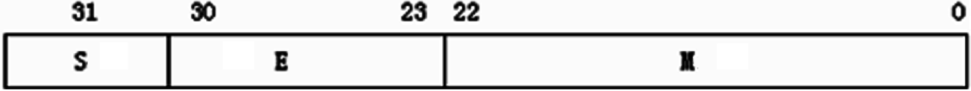


图1.532位浮点数的标准格式

在总长为32位的单精度浮点数存储结构中，最大指数为+127，最小指数为-126，指数偏移量为+127。在IEEE754标准中，约定小数点左边隐含一位，通常这个位就是1，因此单精度浮点数尾数的有效位数为24位(但计算机实际只存储23位，隐含的“1.”不存储)，即尾数为1.×××•••××。

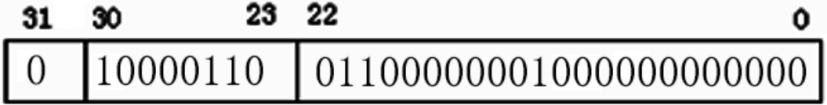
例：利用IEEE754标准将数176.0625表示为单精度浮点数。

解：首先将该十进制转换成二进制为：(176.0625)10=(10110000.0001)2，其次对二进制数进行规格化处理，10110000.0001=1.01100000001×27，这样保证了b0为1，且尾数为1.01100000001，在小数点右边部分最低位后补零满24位得1.01100000001000000000000，将“1.0”隐含掉得计算机实际存储的23位尾数：M=01100000001000000000000，这样便得到了满足图1.5所示的单精度尾数部分。

然后求阶码，上述表示中的指数为7，而单精度浮点数规定指数的偏移量为127(注意，不是前面移码描述中提到得的128)，即在指数7上加127。那么E=7+127=134，则指数的移码表示为10000110。

最后，由于本例中数是一个正数，故S为0。最终得到(176.0625)10的单精度浮点数结果为：

01000011001100000001000000000000，结合图1.5理解为：



###### 历年真题

●(2018年上)浮点数的表示分为阶和尾数两部分。两个浮点数相加时，需要先对阶，即(1)(n为阶差的绝对值)。

(1)A.将大阶向小阶对齐，同时将尾数左移n位

B.将大阶向小阶对齐，同时将尾数右移n位

C.将小阶向大阶对齐，同时将尾数左移n位

D.将小阶向大阶对齐，同时将尾数右移n位

●(2015年下)浮点数能够表示的数的范围是由(3)的位数决定的。

(3)A.尾数 B.阶码 C.数符 D.阶符

●(2012年下)以下关于数的定点表示和浮点表示的叙述中，不正确的是 (3) 。

(3)A.定点表示法表示的数(称为定点数)常分为定点整数和定点小数两种

B.定点表示法中，小数点需要占用一个存储位

C.浮点表示法用阶码和尾数来表示数，称为浮点数

D.在总位数相同的情况下，浮点表示法可以表示更大的数

●(2011年上)计算机中的浮点数由三部分组成：符号位S，指数部分E(称为阶码)和尾数部分M。在总长度固定的情况下，增加E的位数、减少M的位数可以(5)。

(5)A.扩大可表示的数的范围同时降低精度 B.扩大可表示的数的范围同时提高精度

C.减小可表示的数的范围同时降低精度 D.减小可表示的数的范围同时提高精度

●(2009年下)浮点数的一般表示形式为N=2E\*F，其中 E 为阶码，F为尾数。以下关于浮点表示的叙述中，错误的是(3)。两个浮点数进行相加运算，应首先(4)。

(3)A.阶码的长度决定浮点表示的范围，尾数的长度决定浮点表示的精度

B.工业标准 IEEE754 浮点数格式中阶码采用移码、尾数采用原码表示

C.规格化指的是阶码采用移码、尾数采用补码

D.规格化表示要求将尾数的绝对值限定在区间[0.5, 1]

(4)A.将较大的数进行规格化处理B.将较小的数进行规格化处理C.将这两个数的尾数相加D.统一这两个数的阶码

## 逻辑运算和算术运算

### 逻辑运算

逻辑运算有两种结果：真值和假值。在本课程的逻辑运算中，真值用1表示，而假值用0表示。

#### 逻辑与(逻辑乘法)

当两个条件同为真，判断结果才为真。通常使用符合∧、x、∩、AND来表示“与”，运算规则如下：0∧0=0、0∧1=0、1∧0=0、1∧1=1。若设X=001，Y=100，则求X∧Y的运算过程如下：

解：001

∧100

———

000

#### 逻辑或(逻辑加法)

当两个条件中任意一个为真时，判断结果为真。通常使用符合∨、+、∪、OR来表示“或”，运算规则如下：0∨0=0、0∨1=1、1∨0=1、1∨1=1。若设X=001，Y=100，则求X∨Y的运算过程如下：

解：001

∨100

———

101

#### 逻辑非(逻辑否定)

即求反,在变量上加一条横线表示求反，也用NOT表是非运算，运算规则如下：**说明: 绘图3**=1；**说明: 1**=0。

#### 逻辑异或

当两个条件一真一假时，判断结果才为真，其余情况结果为假。通常使用符号⊕表示，运算规则如下：0⊕0=0、0⊕1=1、1⊕0=1、1⊕1=0。若设X=001，Y=100，则求X⊕Y的运算过程如下：

解：001

⊕100

———

101

●(2017年上半年)要判断宇长为 16 位的整数 a 的低四位是否全为 0，则( 2 )。

(2)A.将 a 与0x000F进行"逻辑与"运算，然后判断运算结果是否等于0

B.将 a 与0x000F进行"逻辑或"运算，然后判断运算结果是否等于F

C.将 a 与0x000F进行"逻辑异或"运算，然后判断运算结果是否等于0

D.将 a 与 0x000F 进行"逻辑与"运算，然后判断运算结果是否等于F

### 算术运算

#### 定点数的算术运算：计算机中基本采用补码加减法，操作数用补码表示，连同符号位一起运算，结果也用补码表示。补码运算的公式如下：

* X补+Y补＝(X+Y)补
* X补+(-Y)补＝(X-Y)补

#### 浮点数的算术运算：设两个浮点数X=2Ex.Mx，Y=2Ey.My，则运算规则如下：

* 操作数检查
* 比较阶码大小完成对阶，**原则小阶向大阶看齐**
* 尾数进行算术运算
* 结果规格化并舍入处理

### 溢出判别

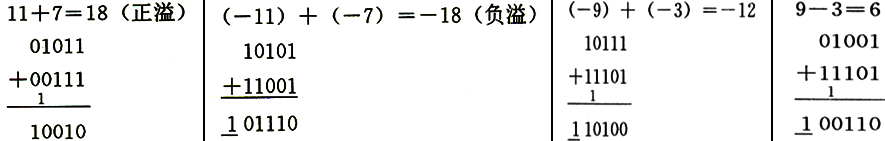
如果运算结果超出了机器数的表示范围，称为溢出。

假设机器字长为5位，为：A0A1A2A3A4，A0为符号位，求采用补码进行运算表示的数值范围？

A0A1A2A3A4表示补码范围是最大为01111(+15)，最小为10000(-16)。在补码讲解中，我们总结过了，在机器字长为n位时，补码比原码、反码都能多表示一个最小的数-2n-1，而这个数并没有对应的原码和反码表示，故A0A1A2A3A4表示补码的最小数为-16。

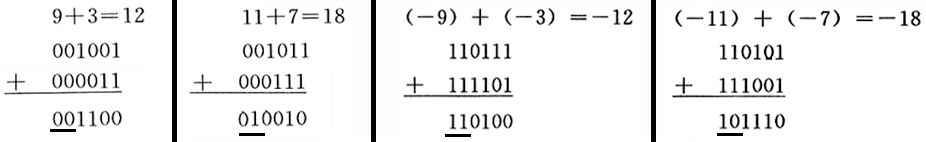
在知道了机器字长的情况下，我们可以求出机器数的表示范围，接下来可以使用如下的两种方式来判断机器数的运算结果是否溢出。

#### 单符号位法。假设机器数的符号位只有一位，假设符号位的进位用Cf表示，最高数值有效位进位用C表示，计算Cf⊕C结果，结果为1表示溢出，否则表示不溢出，即当Cf、C不同时，表示溢出。看看下面的运算结果判断举例，这里使用补码进行计算。



* 在11+7的运算中，Cf=0，C为1，则Cf⊕C=0⊕1=1，表示11+7溢出，由于字长为5位的机器数最大值为15，而11+7=18大于了最大值15，故为正溢出。
* 在(-11)+(-7)的运算中，Cf=1，C为0，则Cf⊕C=1⊕0=1，表示(-11)+(-7)溢出，由于字长为5位的机器数最小值为-16，而(-11)+(-7)=-18小于了最小值-16，故为负溢出。
* 在(-9)+(-3)的运算中，Cf=1，C为1，则Cf⊕C=1⊕1=0，表示(-9)+(-3)=-12未溢出。
* 在9-3的运算中，Cf=1，C为1，则Cf⊕C=1⊕1=0，表示9-3=6未溢出。

#### 双符号位法。假设机器数的符号位为2位，用Sf1表示第一个符号位，Sf2为第二个数符号位，Sf1⊕Sf2结果为1表示溢出，否则未溢出。



* 在9+3=12的运算中，Sf1=0，Sf2=0，则Sf1⊕Sf2=0⊕0=0，表示9+3=12未溢出。
* 在11+7=18的运算中，Sf1=0，Sf2=1，则Sf1⊕Sf2=0⊕1=1，表示11+7=18溢出，由于大于最大的数15，则为正溢出。
* 在(-9)+(-3)=-12的运算中，Sf1=1，Sf2=1，则Sf1⊕Sf2=1⊕1=0，表示(-9)+(-3)=-12未溢出。
* 在(-11)+(-7)=-18的运算中，Sf1=1，Sf2=0，则Sf1⊕Sf2=1⊕0=1，表示(-11)+(-7)=-18溢出，由于小于最小的数-18，则为负溢出。

###### 历年试题

●(2013年下)若计算机存储数据采用的是双符号位(00表示正号，11表示负号)，两个符号相同的数相加时，如果运算结果的两个符号位经 (3) 运算得1，才可断定这两个数相加的结果产生了溢出。

(3)A.逻辑与 B.逻辑或 C.逻辑同或 D.逻辑异或

●(2010年下)若计算机采用8位整数补码表示数据，则运算 (2) 将产生溢出。

(2)A.-127+1 B.127-1 C.127+1 D.127-1

●(2010年上)与等价的逻辑表达式是 (4) 。(表示逻辑异成，+表示逻辑加)

(4)A. B. C. D.

●(2008年下)计算机在进行浮点数的相加(减)运算之前先进行对阶操作，若堆区x的阶码大于y的阶码，则应将(2)。

(2)A.x的阶码缩小至与y的阶码相同，且使x的尾数部分进行算术左移

B.x的阶码缩小至与y的阶码相同，且使x的尾数部分进行算术右移

C.y的阶码扩大至与x的阶码相同，且使y的尾数部分进行算术左移

D.y的阶码扩大至与x的阶码相同，且使y的尾数部分进行算术右移

# 计算机体系结构

## 计算机系统结构概述

计算机系统结构指的是计算机的属性及功能特征，主要包括：

* 硬件所能识别和处理的数据类型
* 所能支持的寻址方式
* CPU内部寄存器数量及使用方式
* CPU指令系统
* 主存的组织与主存的管理
* 中断系统的功能
* 输入输出设备及连接接口

## 存储系统

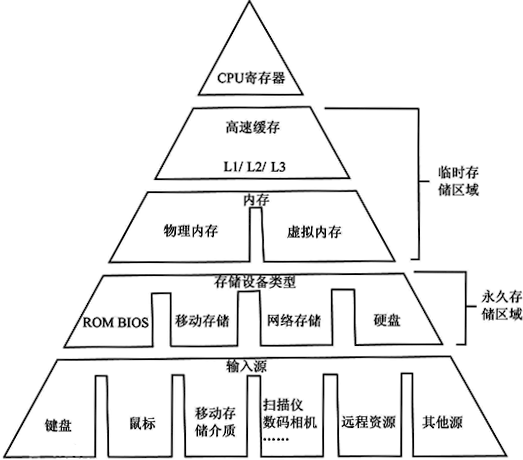


图2.1存储系统

存储器的类型和特点:

按存储介质分:

半导体存储器、磁表面存储器、光存储器

按读写性质分:

随机读写存储器(RAM)和只读存储器(ROM)

RAM又分为：

静态随机存储器(SRAM)；速度快，状态稳定，价格高

动态随机存储器(DRAM)，速度稍慢，状态稳定性稍差，价格稍低。

由于它们存储的内容断电则消失故称为易失性存储器

只读存储器(ROM)

包含掩膜型ROM，EPROM(紫外线擦除可编程序的只读存储器)，EEPROM(可电擦可编程序只读存储器)等类型，

由于其内容断电也不消失故称为非易失性存储器

### 主存(MainMemory)

主存储器简称主存，用来存放程序运行期间的大量程序和数据。存主存主要由半导体器件组成，一个双稳态半导体电路可以组成一个存储元，即表示可以存储1“位”(bit)二进制代码。若干个存储元构成一个存储单元，每个存储单元有唯一的编号，称为地址。许多存储单元构成一个存储器。存储内容断电后即消失，为非永久性存储器。在内存中的地址是以字节为基本单位的，是从小到大连续的，一个存储单元至少包含一个字节，也可包含多个字节。如图2.2所示的图中，一个存储单元恰好包含一个字节，即8个位。



图2.2内存示意图

存储器所有存储单元的总容量称为存储器的存储容量，几个基本的概念如下：

* 字节(B，Byte)：8位(bit)构成一个字节；
* 字节地址：存放一个字节的单元，称为字节存储单元，相应的地址称为字节地址，；
* 字长：CPU在同一时间能一次处理的二进制数的位数叫字长，可为一个字节长度或字节的倍数；
* 字地址：存放一个字的存储单元，称为字存储单元，相应的单元地址叫做字地址；
* 按字节寻址：如果计算机中可编址的最小单位为字节，则称为按字节寻址；
* 按字寻址：如果计算机中可编址的最小单位是字存储单元，则称为按字寻址；

当一个机器字包含多个字节时，即一个字存储单元包含多个字节单元，且每个字节单元能够独立编址的，这样的计算机可以按字节寻址，也可以按字寻址。例如一个字长为16位的计算机，可按字节寻址，也可以按字寻址。当存储器容量为4B时，按字节寻址时，可以寻访到4个字节单元，如果按字寻址时，可以寻访到2个字单元，寻访总容量仍为4B。

常用数值换算：1B=8bit、1KB=1024B=210B、1MB=1024KB=210KB、1GB=1024MB=210MB=220KB、1TB=1024GB=210GB=220MB=230KB。

###### 历年试题

●(2019年上)CPU访问存储器时，被访问数据一般聚集在一个较小的连续存储区域中。若一个存储单元已被访问，则其邻近的存储单元有可能还要被访问，该特性被称为 (3)

(3)A.数据局部性 B.指令局部性 C.空间局部性 D.时间局部性

●(2018年下)在微机系统中，BIOS(基本输入输出系统)保存在 (2) 中。

(2) A.主板上的ROM B.CPU的寄存器 C.主板上的RAM D.虚拟存储器

●(2017年下)内存按字节编址，若用存储容量为32K×8bit的存储器芯片构成地址从A0000H到DFFFFH的内存，则至少需要 (3) 片芯片。

(3)A.4 B.8 C.16 D.32

●(2017年下)计算机系统的主存主要是由 (4) 构成的。

(4)A.DRAM B.SRAM C.cache D.EEPROM

●(2013年下)若某计算机字长为32位，内存容量为2GB，按字编址，则可寻址范围为(6)。

(6)A.1024M B.1GB C.512M D.2GB

●(2012年上)内存单元按字节编址，地址0000A000H～0000BFFFH共有 (2) 个存储单元。

(2)A.8192K B.1024K C.13K D.8K

●(2010年下)设用2K×4位的存储器芯片组成16K×8位的存储器(地址单元为0000H～3FFFH，每个芯片的地址空间连续)，则地址单元0B1FH所在芯片的最小地址编号为 (4)。

(4)A0000H B.0800H C.2000H D.2800H

●(2007年下)若内存地址区间为4000H～43FFH，每个存贮单元可存储16位二进制数，该内存区域用4片存储器芯片构成，则构成该内存所用的存储器芯片的容量是(6)。

(6)A.512×16bit B.256×8bit C.256×16bit D.1024×8bit

### 高速缓存Cache

Cache是一种高速缓冲存储器，目的是**为了解决CPU与主存之间速度不匹配的问题**，为了追求快速，包括控制与管理等所有的功能全部**由硬件实现，对程序员是透明的**。Cache在存储器层次结构中的位置如图2.3所示。

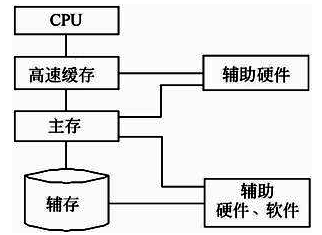


图2.3存储器层次结构示意图

#### Cache的基本原理

CPU与cache之间的数据交换是以字为单位的，而cache与主存之间的数据交换是以块为单位的。当CPU需要读取一个数据时，便用该数据的内存地址访问内存，并迅速判定该地址中的信息是否已进入Cache中，若进入，则将内存地址变换为Cache地址，立即访问Cache。若不在，则将该数据从内存中读出送往CPU，与此同时，把含有这个数据的整个数据块从主存中读取送到Cache中，任何时候Cache中的信息都是主存中某一部分信息的副本。

#### Cache的命中率

* 命中率：在一个程序执行期间，设Nc表示Cache完成存取的总次数，Nm表示主存完成的总次数，Cache的命中率h定义为，则有：h＝Nc/(Nc＋Nm)。
* 平均访问时间：若tc表示命中时Cache访问时间，tm表示未命中时的主存访问时间，1－h表示未命中率，则Cache存储系统的平均访问时间ta为：ta＝h tc＋(1－h)tm。
* Cache存储系统速度提高倍数r：使用Cache比不使用Cache的CPU访问存储器速度提高的倍数r=tm/ta。

### 主存与Cache的地址映射

在CPU需要读写指令和数据时，送出的是指令和数据的主存地址，为了从Cache存储器中读写信息，必须采用某种方法把主存地址转换为Cache存储器的地址，称为地址映象。

地址映射方式有全相联映象方式、直接映象方式和组相联映象方式三种。

#### 全相联映象方式

将主存中的一个块直接拷贝到Cache中任意一块上，Cache的数据块大小与主存的数据块存储的数据量大小相等(如图2.4所示)。在图中，主存共有8个块，要将这8个块的地址(块间地址)表示出来，需要用3个bit位：0号块地址000，1号块地址001，2号块地址010，……，7号块地址111。对于Cache，共有4个块，需用2个bit位表示块地址(块间地址)，分别是：0号块地址00、1号块地址01、2号块地址10、3号块地址11。图2.4中主存和Cache，它们的块容量都是一样的，每个块都包含4个格子，它们的块内地址依次是00、01、10、11(图2.4中放置于块内的“00”、“01”等不要理解为被存储的数据内容，它们实际上是块内地址)。

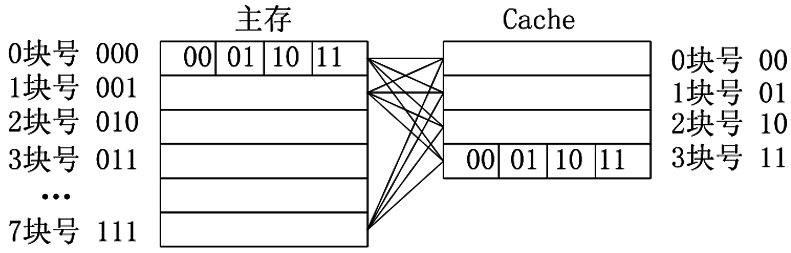


图2.4全相联映象方式示意图

系统采用全相联映像方式时，当CPU给出所需数据在主存中的地址时，同时需要将主存块号转换为Cache块号，为此需要设置一个块号对照表，如表2.1所示。

表2.1块号对照表

|  |  |  |
| --- | --- | --- |
| 主存块号 | Cache块号 | 装入位置 |
| 3 | 0 | 1 |
| 9 | 1 | 1 |
| 12 | 2 | 1 |

* 地址变换：如图2.5所示，CPU给出访存地址时，则根据主存块号立即查找块号对照表，从表中找出对应的Cache块号，将Cache块号与块内地址连接起来就是Cache的地址。

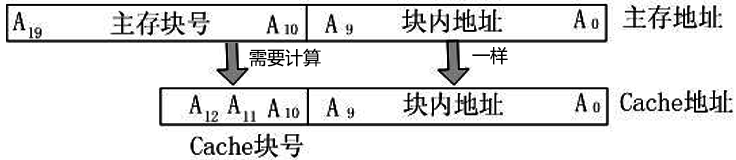


图2.5全相联映象方式地址结构

* 优点：主存的块调入Cache中的位置不受限制，所以冲突率最低。
* 缺点：无法从主存地址中直接获得Cache的块号，变换复杂，速度较慢。

#### 直接映象方式

直接映象方式是一个主存块只能直接拷贝到Cache的一个固定的位置上去，**做法是将主存分成与Cache一样大小的区**，每个区的第0块只能调入到Cache的第0块上，第1块只能放到第1块，其他依次类推(如图2.6所示)。直接映象方式是一种多对一的映射关系。

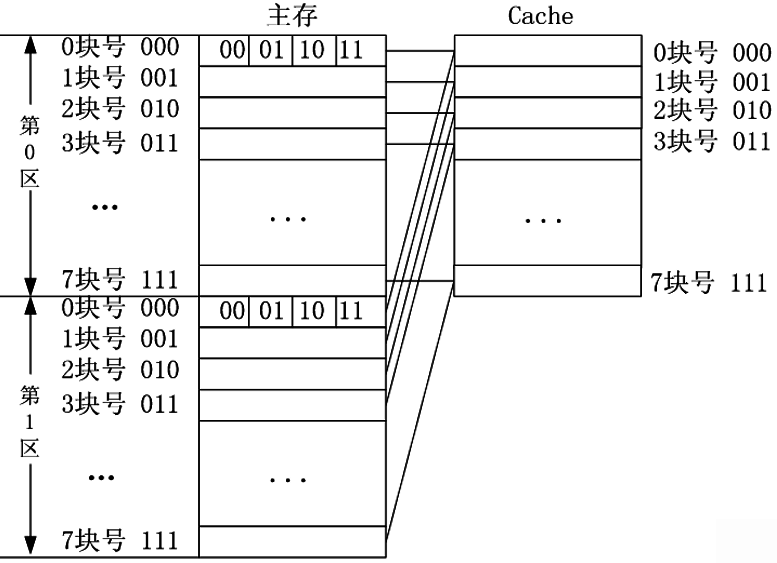


图2.6直接映象方式示意图

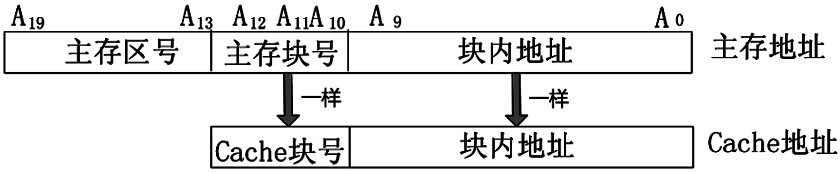


图2.7直接映像方式地址结构

* 地址变换：如图2.7所示，将主存地址的区内主存块号和块内地址连接在一起便构成Cache地址。
* 优点：地址变换很简单，访问速度快。
* 缺点：不同区的相同块号不能同时放进Cache，即便Cache内有空闲块，所以冲突率最大。

#### 组相联映象方式

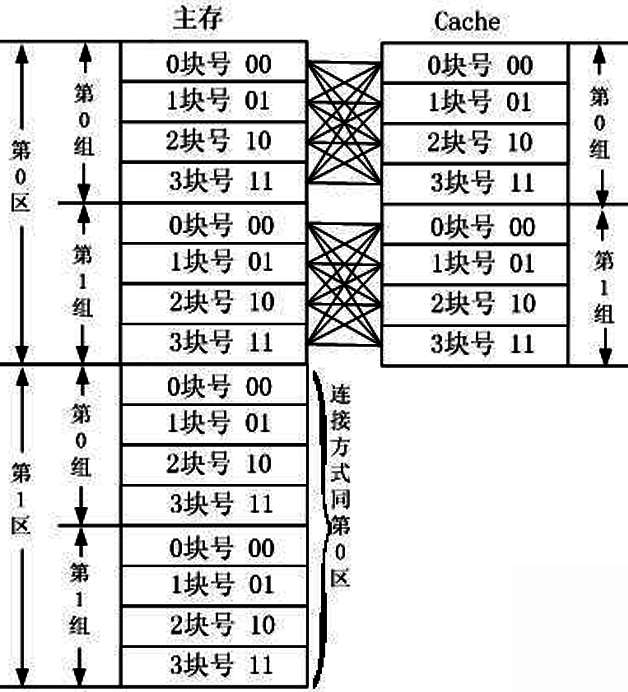


图2.8组相联映像方式示意图

组相联方式是前两种方式的折衷，是将主存中的区和Cache中的块再分成组。如图2.8所示，主存的每个区和Cache的大小一样，将主存中每个区和Cache中的每4块分为一组，主存中每个区就分为2组，分别是第0组和第1组，而Cache也一样，分为第0组第1组。这里规定主存中的组采取直接映像方式而块采取全相连映像方式。即主存中任何区的0组只能放存放到Cache中的0组，1组只能存放到Cache中1组，依此类推。而组内的块则采用全相连映像方式，即主存一组内的块可任意存到Cache相应组号内的任意块中。

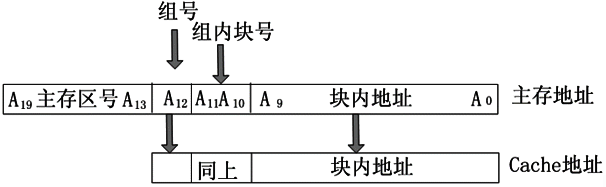


图2.9组相联映象方式地址结构图

如图2.9所示，通过直接映像方式来决定组号，在一组内再使用全相联映像方式来决定Cache中的块号。既具备了直接映像方式的简单和速度快的优点，同时也很好的降低了冲突率。

● 容量为64块的Cache采用组相联方式映像，字块大小为128个字节，每4块为一组。若主存容量为4096块，且以字节编址，那么主存地址应该为\_\_(7)\_\_位，主存区号为\_\_(8)\_\_位。  
(7)A.16 B.17 　 C.18　 D.19 (8)A.5 　 B.6 　 C.7　 　D.8

#### 替换策略

当Cache已满，而主存中又有块要调入Cache时，常采用替换算法。替换算法的目标是使Cache获得最高命中率。常采取的替换算法有：

* 随机替换算法(Rand)：用随机数发生器产生一个要替换的块号，将该块替换出去。此算法简单、易于实现，但未考虑Cache过去、现在及将来的使用情况，没有利用存储器上使用的“历史信息”，没有根据访存的局部性原理，故不能提高Cache的命中率，命中率极低。
* 先进先出算法(FIFO，First In First Out)：当有新内容要加入Cache中时，将最早进入Cache的数据块替换为新加入的Cache数据块。此法不需要记录各字块的使用情况，容易实现，系统开销小，但可能会把一些需要经常使用的程序块(如循环程序)也作为最早进入Cache的块替换掉，且没有根据访问的局部性原理，故不能提高Cache命中率。
* 近期最少使用算法(LRU，Least Recently Used)：即挑选近期最久没有被使用过的块作为被替换块，能较好的反映程序的局部性特点。此法很好地反映了程序局部性规律，但需要随时记录Cache中各块的使用情况，以确定哪个块是近期最少使用的块。此算法相对合理，但实现起来比较复杂，系统开销较大，通常需要对每一个块设置一个称为计数器的硬件或软件模块，用以记录其被使用的情况。
* 最优替换算法(OPTimal replacement)：必须先执行一次程序，统计Cache的替换情况。有了这样的先验信息，在第二次执行该程序时便可以用最有效的方式来替换。此算法从整体上把握，这种算法的Cache命中率是最高的，它是最优的替换算法，但此种算法实现起来难度大，故只是一种理想算法。
* 近期最不经常使用算法(LFU，Least Frequently Used)：选择近期被访问次数最少的页面作为被替换的页面。这是一种合理的算法，因为到目前为止最少使用的页面，很可能也是将来最少访问的页面。该算法充分利用了主存中页面调度情况的历史信息，又正确反映了程序的局部性。但此算法实现起来非常困难，要为每个页面设置一个很长的计数器，并且要选择一个固定的时钟为每个计数器定时计数。在选择被替换页面时，要从所有计数器中找出一个计数值最大的计数器。

**注意：**LRU是基于最久未被访问的时间来确定被替换的页面；而LFU是基于访问次数来确定，最近时间段内被访问次数最少的页面被替换掉。所以LRU是基于访问时间的，而LFU是基于访问次数的。

###### 历年试题

●(2017年下)在程序的执行过程中，C ache与主存的地址映射是由 (1) 完成的。

(1)A.操作系统完成 B.程序员调度 C.硬件自动 D.用户软件

●(2017年上)以下关于Cache (高速缓冲存储器)的叙述中，不正确的是( 6 )。

(6)A.Cache 的设置扩大了主存的容量 B.Cache 的内容是主存部分内容的拷贝

C.Cache 的命中率并不随其容量增大线性地提高 D.Cache 位于主存与 CPU 之间

●(2016年上)主存与Cache的地址映射方式中，(2)方式可以实现主存任意一块装入Cache任意位置，只有装满才需要替换。

(2)A.全相联 B.直接映射 C.组相联 D.串并联

### 虚拟存储器

程序均需经由内存执行，若执行的程序占用内存很大或很多，则会导致内存消耗殆尽。为解决该问题，Windows等操作系统中运用了虚拟存储(Virtual Memory)技术，即匀出一部分硬盘空间来充当内存使用。当内存耗尽时，电脑就会自动调用硬盘来充当内存，以缓解内存的紧张。若计算机运行程序或操作所需的随机存储器(RAM)即内存不足时，Windows会用虚拟存储器进行补偿。它将计算机的RAM和硬盘上的临时空间组合。当RAM运行速率缓慢时，它便将数据从RAM移动到称为“分页文件”的空间中。将数据移入分页文件可释放RAM，以便完成工作。一般而言，计算机的RAM容量越大，程序运行得越快。若计算机的速率由于RAM可用空间匮乏而减缓，则可尝试通过增加虚拟内存来进行补偿。但是，计算机从RAM读取数据的速率要比从硬盘读取数据的速率快，因而扩增RAM容量(可加内存条)是最佳选择。

###### 历年试题

●(2015年下)虚拟存储体系由(2)两级存储器构成。

(2)A.主存-辅存 B.寄存器-Cache C.寄存器-主存 D.Cache-主存

●(2015年上)Cache的地址映像方式中，发生块冲突次数最小的是 (3)。

(3)A.全相联映像 B.组相联映像 C.直接映像 D.无法确定的

●(2009年下)以下关于 Cache 的叙述中，正确的是(6)。

(6)A.在容量确定的情况下，替换算法的时间复杂度是影响 Cache 命中率的关键因素

B.Cache 的设计思想是在合理成本下提高命中率

C.Cache 的设计目标是容量尽可能与主存容量相等

D.CPU 中的Cache容量应大于CPU之外的Cache容量

●(2007年上)在CPU与主存之间设置高速缓冲存储器Cache,其目的是为了 (2)

(2)A.扩大主存的存储容量B.提高CPU对主存的访问效率 C.既扩大主存容量又提高存取速度D.提高外存储器的速度

### 硬磁盘存储设备

* 优点：存储容量大，价格低，信息可以长期保存不丢失等。
* 缺点：存取速度慢，机械结构复杂。

一个存储器的性能通常用速度、容量、价格三个主要指标来衡量。计算机对存储器的要求是容量大、速度快、成本低，需要尽可能地同时兼顾这三方面的要求。但是一般来讲，存储器速度越快，价格也越高，因而也越难满足大容量的要求。目前通常采用多级存储器体系结构，使用高速缓冲存储器、主存储器和外存储器，如图2.10所示：

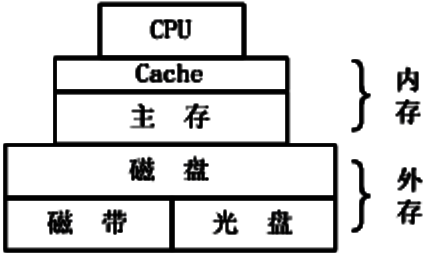


图2.10分层次存储系统

* CPU能直接访问的存储器称为内存储器(简称内存)，包括高速缓冲存储器和主存储器。CPU不能直接访问的存储器称为外存储器(简称外存，也叫辅助存储器)，外存的信息必须调入内存才能被CPU使用。
* 计算机存储系统的这种多层次结构，很好地解决了容量、速度、成本三者之间的矛盾。这些不同速度、不同容量、不同价格的存储器，用硬件、软件或软硬件结合的方式连接起来，形成一个系统。这个存储系统对应用程序员而言是透明的，在应用程序员看来它是一个存储器，其速度接近于最快的那个存储器，存储容量接近于容量最大的那个存储器，单位价格则接近最便宜的那个存储器。

#### 磁记录原理

磁性材料被不同的电流磁化后，会形成两个稳定的剩磁状态，利用这两个稳定的状态表示二进制代码1和0。磁性材料上呈现剩磁状态的地方形成了一个磁化元或存储元，是记录一个二进制信息位的最小单位。如图2.11所示，写线圈和读线圈会形成磁化电流而从磁性材料中写入和读出数据。

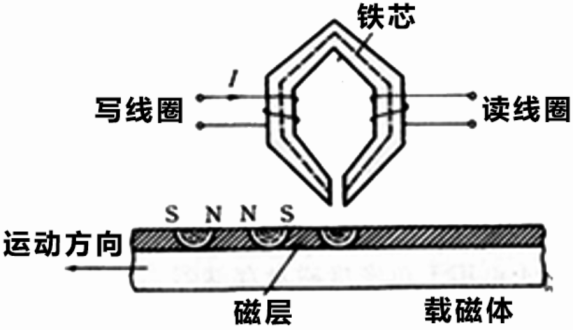


图2.11磁记录原理图

#### 记录方式

磁性材料被写入二进制代码0或1，是靠不同的写入电流波形来实现的。形成不同写入电流波形的方式，称为记录方式。常用的记录方式如图2.12所示。

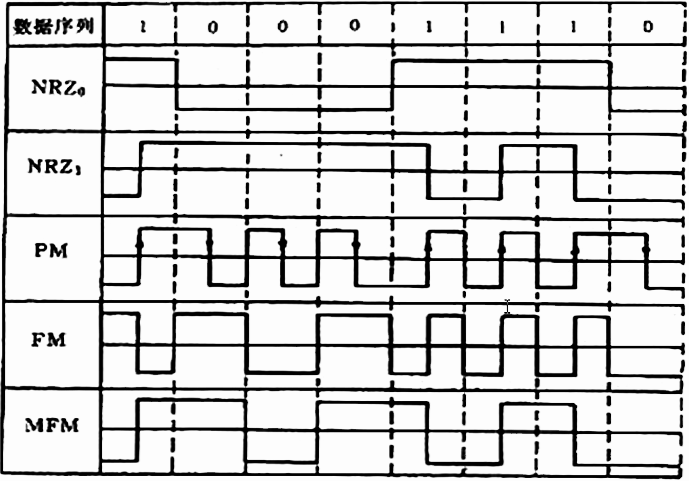


图2.12常见的磁记录方式

* 不归零制(NRZ0)：其特点是，磁头线圈中始终有电流，不是正向电流(代表1)就是反向电流(代表0)，因此不归零制记录方式的抗干扰性能较好。
* 见1就翻不归零制(NRZ1)：与NRZ0相同之处是磁头线圈中始终有电流。不同之处在于，记录“0”时电流方向不变，只有遇到“1”时才改变方向。
* 调相制(PM)：其特点是，在一个位周期中间位置，电流由负到正为1，由正到负为0，即利用电流相位的变化来进行写1和0，所以通过磁头中电流方向一定要改变一次。
* 调频制(FM)：其特点如下：①在相邻两个存储元交界处电流都要改变方向；②记录1时电流一定要在位周期中间改变方向，即写1的电流频率是写0电流频率的2倍，称为倍频法。
* 改进调频制(MFM)：与调频制的区别在于只有连续记录两个或两个以上0时，才在位周期的起始位置翻转一次，而不是在每个位周期的起始处都翻转。

#### 磁盘的性能

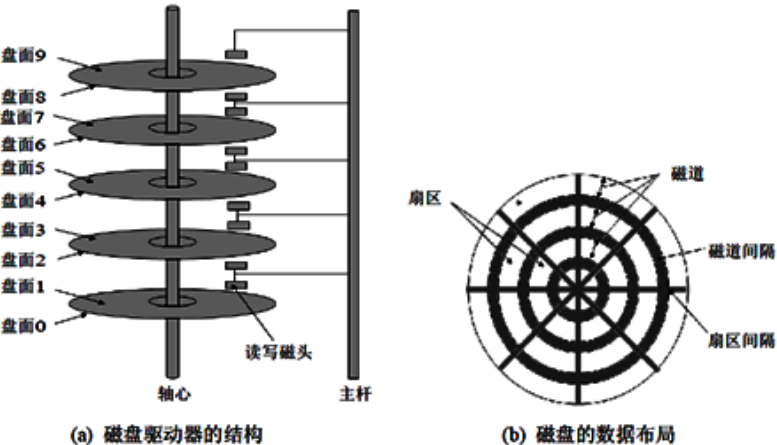


图2.13磁盘示意图

如图2.13所示，硬磁盘由一组绕轴旋转的盘片组成。每个盘片的上下两面都能记录信息，通常把磁盘表面称为记录面。记录面上一系列同心圆称为磁道。每个盘片表面通常有几十到几百个磁道，磁道的编址是从外向内依次编号的，最外一个同心圆叫0磁道，往内依次增加，最里面的一个同心圆叫n磁道，n磁道里面的圆面积不用来记录信息。具有相同的直径，同时位于一组磁头下方的所有磁道被称为柱面。文件存储在硬盘上时尽可能放在同一柱面上，或者放在相邻的柱面上。

每个磁道又分为若干个扇区，扇区是读写的最小单元，磁盘中所有磁道具有相同数目的扇区,由于外磁道长，扇区也比较长，所以记录密度比内磁道低，所以最里圈的位密度称为最大位密度。磁盘示意图如图2.13所示。

* 存储密度：道密度是沿磁盘半径方向单位长度上的磁道数，单位是道/英寸；位密度是磁道单位长度上能记录的二进制代码位数，单位为位/英寸；面密度是位密度和道密度的乘积，单位是位/平方英寸。
* 存储容量：指一个磁盘存储器所能存储的字节总数。格式化容量是指按照某种特定的记录格式所能存储信息总量，也是用户可以真正使用的容量。

①格式化容量=面数×(磁道数/每面)×(扇区数/道)×(字节数/扇区)

②非格式化容量是磁记录表面可以利用的磁化单元总数，非格式化容量=面数×(磁道数/每面)×内圆周长×最大位密度

* 数据传输率：磁盘存储器在单位时间内向主机传送数据的字节数叫做数据传输率。

平均数据传输率=(扇区数/道)×(字节数/扇区)×盘片转速

#### 磁盘调度

##### 固定头磁盘

这种磁盘在每条磁道上都有一读/写磁头，所有的磁头都被装在一刚性磁臂中。通过这些磁头可访问所有各磁道，并进行并行读/写，有效地提高了磁盘的I/O 速度。这种结构的磁盘主要用于大容量磁盘上。

##### 移动头磁盘

每一个盘面仅配有一个磁头，也被装入磁臂中。为能访问该盘面上的所有磁道，该磁头必须能移动以进行寻道。可见，移动磁头仅能以串行方式读/写，致使其I/O 速度较慢；但由于其结构简单，故仍广泛应用于中小型磁盘设备中。在微型机上配置的温盘和软盘都采用移动磁头结构，故本节主要针对这类磁盘的I/O进行讨论。

磁盘的一般工作过程是：磁头先移动到目标磁道上，然后期望的扇区旋转到磁头下方，接着读取扇区中的数据。

磁盘调度分为移臂调度和旋转调度，并且是先进行移臂调度，在移动臂定位后，然后再进行旋转调度。

* 移臂调度

**先来先服务FCFS：**根据进程请求访问磁盘的先后次序进行调度。

**最短寻道时间 SSTF：**该算法选择这样的进程，其要求访问的磁道与当前磁头所在的磁道距离最近，使得每次的寻道时间最短，容易发生饥饿现象(离磁头所在的磁道距离远的请求总是得不到服务)。

**扫描算法 SCAN：**该算法不仅考虑到准备访问的磁道与当前磁道的距离，更优先考虑的是磁头的当前移动方向。例如当前磁头由内向外移动，下面将要访问的磁道应该是距离已访问磁道最近，且在该磁道外侧的磁道，直到再无更外层的磁道需要访问时才将磁臂换方向。也称为电梯调度算法

**循环扫描算法CSCAN：**规定磁头作单向扫描，例如自里向外时，当磁头访问外最外层的磁道后，立即返回最内的欲访问磁道，即最小的磁道号紧接着最大的磁道号访问。

* 旋转调度

①请求访问的是同一磁道的不同编号的扇区；

②请求访问的是不同磁道上的不同编号的扇区；

③请求访问的是不同磁道上具有相同编号的扇区。

对①与②旋转调度总是让首先到达读写磁头位置下的扇区先进行传送操作；对③旋转调度可以任选一个读写磁头位置下的扇区进行传送操作。

#### 磁盘访问时间

* 寻道时间：指把磁臂(磁头)从当前位置移动到指定磁道上所经历的时间。
* 旋转延迟时间：指定扇区移动到磁头下面所经历的时间。
* 传输时间：指把数据从磁盘读出，或向磁盘写入数据所经历的时间。

磁盘的访问时间=寻道时间+旋转时间+传输时间+控制器开销时间。

### 相联存储器

按内容访问的存储器。把任一项存储内容作为地址来存取。

###### 历年试题

●(2014年下)计算机采用分级存储体系的主要目的是为了解决 (2) 的问题。

(2)A.主存容量不足 B.存储器读写可靠性 C.外设访问效率 D.存储容量、成本和速度之间的矛盾

●(2012年下) (4) 不属于按寻址方式划分的一类存储器。

(4) A.随机存储器 B.顺序存储器 C.相联存储器 D.直接存储器

●(2012年上)相联存储器按 (3) 访问。

(3)A.地址 B.先入后出的方式 C.内容 D.先入先出的方式

●(2011年下) 假设磁盘每磁道有18个扇区，系统刚完成了10号柱面的操作，当前移动臂在13号柱面上，进程的请求序列如下表所示。若系统采用SCAN(扫描)调度算法，则系统响应序列为(26)；若系统采用CSCAN(单向扫描)调度算法，则系统响应序列为 (27)。

|  |  |  |  |
| --- | --- | --- | --- |
| 请求序列 | 柱面号 | 磁头号 | 扇区号 |
| ①  ②  ③  ④  ⑤  ⑥  ⑦  ⑧  ⑨  ⑩ | 15  20  30  20  5  2  15  6  8  15 | 8  6  9  10  4  7  8  3  7  10 | 9  5  6  5  5  4  1  10  9  4 |

(26)A.⑦⑩①②④③⑨⑧⑤⑥ B.①⑦⑩②③④⑥⑤⑧⑨

C.⑦⑩①②④③⑥⑤⑧⑨ D.①⑦⑩②③④⑧⑨⑥⑤

(27)A.⑦⑩①②④③⑨⑧⑤⑥ B.①⑦⑩②③④⑥⑤⑧⑨

C.⑦⑩①②④③⑥⑤⑧⑨ D.①⑦⑩②③④⑧⑨⑥⑤

●(2010年下)正常情况下，操作系统对保存有大量用数据的硬盘进行 (6)操作时，不会清除有用数据。

(6)A.磁盘分区和格式化 B.磁盘格式化和碎片整理 C.磁盘清理和碎片整理 D.磁盘分区和磁盘清理

●(2010年下) 某磁盘磁头从一个磁道移至另一个磁道需要10ms。文件在磁盘上非连续存放，

逻辑上相邻数据的平均移动距离为10个磁道，每块的旋转延尺时间及传输时间分别为100ms和2ms，则读取一个100块的文件需要 (26)ms时间。

(26)A.10200 B.11000 C.11200 D.20200

●(2009年上)假设某硬盘由5个盘片构成(共有8个记录面)，盘面有效记录区域的外直径为30cm，内直径为10cm，记录位密度为250位/mm，磁道密度为16道/mm，每磁道分16个扇区，每扇区512字节，则该硬盘的格式化容量约为(2)MB。

(2)A. B.

C. D.

●(2009年上)(3)是指按内容访问的存储器。

(3)A.虚拟存储器 B.相联存储器 C.高速缓存Cache D.随机访问存储器

## 指令系统

指令就是计算机执行某种操作的命令。计算机的程序就是由一系列指令组成的，计算机的指令有微指令、机器指令和宏指令之分。微指令是微程序级的命令，它属于硬件；宏指令是由若干条机器指令组成的软件指令，它属于软件，而机器指令则介于微指令和宏指令之间，简称为指令，每一条指令可以完成一个独立的算术运算或逻辑运算操作。本节指令系统是指机器指令。一台计算机中所有机器指令的集合，称为这台机器的指令系统。

计算机发展初期，指令较少，完成功能简单。随着集成电路的发展，出现了系列计算机，即基本指令系统相同、基本体系结构相同的一系列计算机。如Pentium系列机。

### 指令格式

指令个事是指指令用二进制代码表示的结构形式，通常由操作码字段和地址码字段构成，如图2.14所示。操作码字段表征指令的操作特性和功能，让CPU完成什么性质的操作，地址码字段通常指定参与操作的操作数的地址或运算结果所在的地址。



图2.14指令格式

### 操作码格式

操作码的位数决定操作类型的多少，位数越多，所能表示的编码就越多，不同的编码代表不同的操作码，表示不同的指令，完成不同的动作。如系统中只有8种运算操作，则只需要3位操作码就够了，即23＝8。

组成操作码字段的位数一般取决于计算机指令系统的规模，较大的指令系统就需要更多的位数来表示每条特定的指令。

操作码在计算机中传输和存储都必须是二进制，但阅读书写不便，通常使用3个或4个英文缩写的字母来表示操作码，这种缩写称为指令助记符，如表2.2，假定该指令系统只有6条指令，所以只需要3位二进制。

表2.2典型指令助记符

|  |  |  |
| --- | --- | --- |
| 指令功能 | 二进制表示的操作码 | 指令助记符 |
| 加法 | 001 | ADD |
| 减法 | 010 | SUB |
| 传送 | 011 | MOV |
| 跳转 | 100 | JMP |
| 存储 | 101 | STR |
| 读数 | 110 | LDA |

### 地址码格式

根据一条指令中有几个操作数地址，又有三地址指令、二地址指令、单地址指令和零地址指令等种指令格式，如图2.15所示。

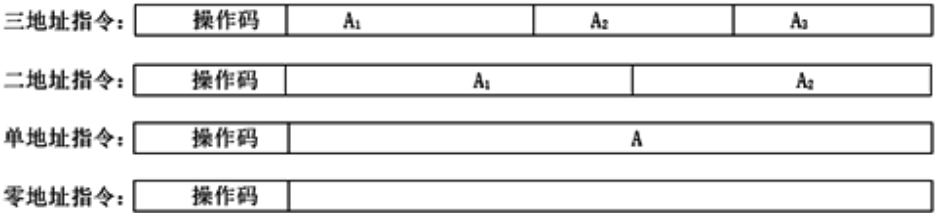


图2.15不同地址指令构成

这里操作数有被操作数、操作数和操作结果。

零地址指令：指令中只有操作码，没有地址码，如停机指令。

单地址指令：也称为单操作数指令，这种指令通常对地址码所对应的存储单元中的数据进行操作后结果仍放入该单元，或者将指令中地址码表示的操作数和CPU内部AC累加寄存器中的已有数据进行操作，结果默认放入累加寄存器AC中，且操作完成后结果仍存入累加寄存器AC中。如：汇编指令：inc ax，表示将CPU内寄存器ax中的数据自动加1，结果仍存入ax中；

二地址指令：也称为双操作数指令，它有两个地址码字段A1和A2，分别指明参与操作的两个数在内存中或CPU内寄存器的位置，其中A1也作为存放操作结果的地址。如：汇编指令add bx,ax，表示将寄存器ax和bx内的数据相加，结果仍存入bx中。

三地址指令：A1为源操作数地址，A2为终点操作数地址，A3为存放结果的地址。

### 扩展操作码

对于机器的指令系统，指令中操作码字段和地址码字段通常是固定，但有些单片机中，由于指令字长较短，为了充分利用指令，指令操作码和地址码长度不固定，以便尽可能用较短的指令字长来表示更多的操作种类，完成更多功能。

#### 定长操作码：操作码的位数与位置固定，便于译码。

#### 扩展操作码：当指令字长较短时，利用将某些指令的地址段位数的减少，扩展操作码的位数。

案例：设系统为定长操作码指令，全部为二地址指令，指令长度共12位，操作码占4位，每个地址码占4位，共有多少条二地址指令？

0 0 0 0 A1 A2

0 0 0 1 A1 A2

.

.

1 1 1 0 A1 A2

1 1 1 1 A1 A2

操作码的编码是从0000~1111，共计16条指令(相同操作码的指令，即使地址码A1、A2不同，也属于同一种功能的指令，只是操作数不一样而已)。

如果留下操作码的1111作为扩展操作码标志，将第一个地址码A1扩展为操作码，将指令扩展为单操作数的指令时，则只有15种组合表示二地址指令，即操作码范围为0000~1110的指令表示二地址指令，如果扩展操作码后，全部用来表示单地址指令，则最多可表示多少条单地址指令？

0 0 0 0 A1 A2

0 0 0 1 A1 A2

.

.

1 1 1 0 A1 A2以上为双地址指令

1 1 1 1 0 0 0 0 A2以下为单地址指令

1 1 1 1 0 0 0 1 A2

.

.

1 1 1 1 1 1 1 0 A2

1 1 1 1 1 1 1 1 A2

在以上的编码组合中，双指令地址的操作码从0000~1110，共计15条，而单指令的操作码从1 1 1 10 0 0 0~1 1 1 1 1 1 1 1，共计16条，故留下操作码的1111作为扩展操作码标志，将第一个地址码A1扩展为操作码时，共有指令条数为15+16=31条。

可以继续扩展单操作数指令操作码，例如将11111111留下来作为扩展指令标志来表示零地址指令，则可表示多少条零地址指令？

0 0 0 0 A1 A2

0 0 0 1 A1 A2

.

.

1 1 1 0 A1 A2 以上为双地址指令

1 1 1 1 0 0 0 0 A2

1 1 1 1 0 0 0 1 A2

.

.

1 1 1 1 1 1 1 0 A2以上为单地址指令

1 1 1 1 1 1 1 1 0 0 0 0

1 1 1 1 1 1 1 1 0 0 0 1

.

.

1 1 1 1 1 1 1 1 1 1 1 0

1 1 1 1 1 1 1 1 1 1 1 1以上为零地址指令

在以上的编码中，双地址指令的操作码编码为0 0 0 0~1 1 1 0，共计15条；单地址指令操作码为1 1 1 10 0 0 0~1 1 1 1 1 1 1 0，共计15条；零地址指令操作码为1 1 1 11 1 1 10 0 0 0~1 1 1 1 1 1 1 111 1 1 1，共计16条；在以上的编码中，指令条数共计46条。

### 指令长度

一个指令中包含二进制代码的位数称为指令字长度，机器字长是指机器中直接处理的二进制数据的位数，它决定了计算机的运算精度。机器字长通常与主存单元的位数一致。指令字长度通常和机器字长的长度密切关联。

如果指令长度等于机器字长度的指令，称为单字长指令，指令字长度等于双字长的指令，称为双字长指令，如果长度为一个机器字长的一半，即为半字长指令。如IBM系列机中，指令有半字长的，有单字长的，还有一个半字长的。

使用多字长的指令，便于在指令中提供足够的地址位来要寻访的内存中的任何单元，扩大了寻址范围，但需要多次访问存储指令的那些单元才能取出一条完整的指令，降低了CPU的运行速度，也占用了更大的存储空间。

如果一个系统中所有指令字长度相等，称为等长指令字结构。

如果一个系统中既有单字长指令，又有双字长指令，称为变长指令字结构，这种指令字结构灵活，但控制和编译较麻烦。

### 指令和数据的寻址方式

一般来说，指令中所给出的地址码，并不是操作数的真正有效地址，根据指令中给出的地址码寻找操作数的有效地址的方式，称为寻址方式。指令系统中采用不同寻址方式的目的是扩大寻址空间并提高编程灵活性。

#### 立即寻址

指令的地址字段指出的不是操作数的地址，而是操作数本身。如指令MOV R1，#45，即将数值45传送到寄存器R1中。

#### 直接寻址

指令的地址字段直接指出操作数在存储器中的地址，不需要经过任何变换，如图2.16所示。

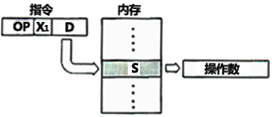


图2.16直接寻址

#### 间接寻址

相对于直接寻址而言的。指令地址字段中的地址所指向的单元不是操作数，而是存放操作数的真正地址，如图2.17所示。

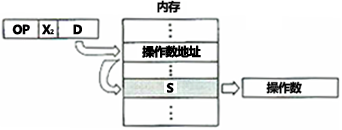


图2.17间接寻址

#### 寄存器寻址方式和寄存器间接寻址方式

寄存器寻址方式的操作数不是放在内存中，而是放在CPU的通用寄存器中，即地址码表示的是该寄存器的地址。寄存器间接寻址方式与寄存器寻址方式的区别在于，指令格式中寄存器中内容不是操作数，而是操作数的地址。

#### 相对寻址方式

相对寻址是把程序计数器PC的内容加上指令中地址码形成操作数的有效地址，如所示。

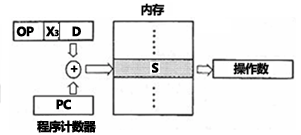


图2.18相对寻址方式

### CISC和RISC的区别

#### CISC(Complex Instruction Set Computer)：复杂指令集计算机。

如Pentium机有191条，９种寻址方式。经实际分析发现：

##### 各种指令使用频率相差悬殊。80%指令使用很少。

##### 指令系统的复杂性带来系统结构的复杂性，增加了设计时间和售价，也增加了VLSI([超大规模集成电路](http://baike.baidu.com/subview/1081357/7552995.htm)Very Large Scale Integration的简称)设计负担，不利于微机向高档机器发展。

##### 复杂指令操作复杂、运行速度慢。

所以复杂指令系统增加硬件复杂性，降低机器运行速度。

#### RISC(Reduced Instruction Set Computer)：精简指令集计算机。

RISC不是简单地简化指令系统，而是通过简化指令使计算机的结构更加简单合理，从而提高运算速度。

计算机执行程序所需用的时间*P*可用下式表示：

*P=I\*CPI\*T*

其中：I是程序在机器上运行的指令数，CPI为执行每条指令所需的平均周期数，T是每个机器周期的时间。

由于RISC指令比较简单，原CISC机中比较复杂的指令在这里都用多个子程序代替，因此RISC中运行的指令数I要比CISC中多20%--40%，但是大多数指令只用一个机器周期实现，所以执行每条指令所需的平均周期数CPI要小，且RISC结构简单，所以完成一个操作经过的数据通路较短，每个机器周期的时间T也减少，故计算机执行程序所需用的时间*P大大降低。*

RISC在CISC基础上，克服了CISC的缺点基础上产生并发展起来的大部分RISC具有如下特点：

##### 优先选取使用频率高的一些简单指令和很有用但不复杂指令，指令条数少。

##### 指令长度固定，指令格式少，寻址方式少。

##### 只有取数/存数指令访问存储器，数据在寄存器和存储器之间传送，其余指令都在寄存器中进行，即限制内存访问。

##### CPU中通用寄存器数量相当多；算数逻辑运算指令的操作数都在通用寄存器中存取。

##### 大部分指令都在一个机器周期内完成。

##### 以硬布线逻辑为主，不用或少用微程序控制，[超大规模集成电路](http://baike.baidu.com/subview/1081357/7552995.htm)VLSI(Very Large Scale Integration, 简称VLSI)工艺的规整性要求高。

##### 原CISC机中比较复杂的指令都用多个子程序代替，RISC编译器的子程序库通常要比CISC编译器的子程序库大。

##### 特别重视编译工作，以简单有效的方式支持高级语言，减少程序执行时间。

### 指令类型

一台计算机的指令系统可以有上百条指令，这些指令按其功能可以分成几种类型。

1.数据传送类指令

(1)一般传送指令

一般传送指令具有数据复制的性质，即数据从源地址传送到目的地址，而源地址中的内容保持不变。一般传送类指令常用助记符MOV表示，根据数据传送的源和目的的不同，运算类指令又可分为：

⑴主存单元之间的传送。

⑵从主存单元传送到寄存器。在有些计算机中，该指令用助记符LOAD表示。

⑶从寄存器传送到主存单元。在有些计算机里，该指令用助记符STORE表示。

⑷寄存器之间的传送。

(2)堆栈操作指令

堆栈指令实际上是一种特殊的数据传送指令，分为进栈(PUSH)和出栈(POP)两种，在程序中它们往往是成对出现的。如果堆栈是主存的一个特定区域，那么对堆栈的操作也就是对存储器的操作。

2.运算类指令

(1.)算术运算类指令

算术运算指令主要用于定点和浮点运算。这类运算包括定点加、减、乘、除指令，浮点加、减、乘、除指令以及加1、减1、比较等，有些机器还有十进制算术运算指令。例如加法常用助记符add表示，减法常用助记符sub表示。

(2)逻辑运算类指令

计算机都具有与、或、非、异或等逻辑运算指令。这类指令在没有设置专门的位操作指令的计算机中常用于对数据字(字节)中某些位(一位或多位)进行操作，例如或运算的助记符为or。

3.移位类指令

移位指令分为算术移位、逻辑移位和循环移位三类，它们又可分为左移和右移两种。

⑴算术移位

算术移位的对象是带符号数，在移位过程中必须保持操作数的符号不变。当左移一位时，如不产生溢出，则数值×2；而右移一位时，如不考虑因移出舍去的末位尾数，则数值÷2。

⑵逻辑移位

逻辑移位的对象是无符号数，因此移位时不必考虑符号问题。

⑶循环移位

循环移位按是否与进位位一起循环又分为两种：

小循环(不带进位循环)

大循环(带进位循环)

4.程序控制类指令

(1)转移指令

在程序执行过程中，通常采用转移指令来改变程序的执行顺序。转移指令又分无条件转移和条件转移两种：

⑴无条件转移又称必转，它在执行时将改变程序的常规执行顺序，不受任何条件的约束，直接把程序转向该指令指出的新的位置执行，其助记符一般为JMP。

⑵条件转移必须受到条件的约束，若条件满足时才执行转移，否则程序仍顺序执行。条件转移指令主要用于程序的分支，当程序执行到某处时，要在两个分支中选择一支，这就需要根据某些测试条件作出判断。

5.输入输出类指令

(1)独立编址的I/O

独立编址方式使用专门的输入/输出指令(IN/OUT)。以主机为基准，信息由外设传送给主机称为输入，反之称为输出。指令中应给出外部设备编号(端口地址)。这些端口地址与主存地址无关，是另一个独立的地址空间。80x86采用的就是独立编址方式。

(2)统一编址的I/O

所谓统一编址就是把外设寄存器和主存单元统一编址。在这种方式下，不需要专门的I/O指令，就用一般的数据传送类指令来实现I/O操作。一个外部设备通常至少有两个寄存器：数据寄存器和命令与状态寄存器。每个外设寄存器都可以由分配给它们的唯一的主存地址来识别，主机可以像访问主存一样去访问外部设备的寄存器。

###### 历年试题

●(2019年上)以下关于RISC (精简指令系统计算机)技术的叙述中，错误的是 (6) 。

(6)A.指令长度固定、指令种类尽量少 B.指令功能强大、寻址方式复杂多样

C.增加寄存器数目以减少访存次数 D.用硬布线电路实现指令解码，快速完成指令译码

●(2016年上)VLIW 是 (1) 的简称 。

(1)A.复杂指令系统计算机 B.超大规模集成电路

C.单指令流多数据流 D.超长指令字(Very Long Instruction Word)

●(2016年上)移位指令中的(4)指令的操作结果相当于对操作数进行乘2操作。

(4)A.算术左移 B.逻辑右移 C.算术右移 D.带进位循环左移

●(2015年下)在机器指令的地址字段中，直接指出操作数本身的寻址方式称为(3)。

(3)A.隐含寻址 B.寄存器寻址 C.立即寻址 D.直接寻址

●(2015年下)CISC是(6)的简称。

(6)A.复杂指令系统计算机 B.超大规模集成电路 C.精简指令系统计算机 D.超长指令字

●(2014年下)以下关于RISC和CISC的叙述中，不正确的是 (5) 。

(5)A.RISC通常比CISC的指令系统更复杂 B.RISC通常会比CISC配置更多的寄存器

C.RISC编译器的子程序库通常要比CISC编译器的子程序库大得多

D.RISC比CISC更加适合VLSI工艺的规整性要求

●(2013年下) (5) 不是RISC的特点。

(5)A.指令种类丰富 B.高效的流水线操作 C.寻址方式较少 D.硬布线控制

●(2012年上)若CPU要执行的指令为：MOV R1，#45(即将数值45传送到寄存器R1中)， 则该指令中采用的寻址方式为 (4) 。

(4)A.直接寻址和立即寻址 B.寄存器寻址和立即寻址 C.相对寻址和直接寻址 D.寄存器间接寻址和直接寻址

●(2011年下)若某条无条件转移汇编指令采用直接寻址，则该指令的功能是将指令中的地址码送入 (1) 。

(1)A.PC(程序计数器) B.AR(地址寄存器) C.AC(累加器) D.ALU(算逻运算单元)

●(2011年上)指令系统中采用不同寻址方式的目的是 (2) 。

(2)A.提高从内存获取数据的速度 B.提高从外存获取数据的速度

C.降低操作码的译码难度 D.扩大寻址空间并提高编程灵活性

●(2010年上)关于64位和32位微处理器，不能以2倍关系描述的是 (6) 。

(6)A.通用寄存器的位数 B.数据总线的宽度 C.运算速度 D.能同时进行运算的位数

●(2009年下)以下关于 CISC(Complex Instruction Set Computer，复杂指令集计算机)和 RISC(Reduced Instruction Set Computer，精简指令集计算机)的叙述中，错误的是 (2) 。

(2)A.在 CISC 中，其复杂指令都采用硬布线逻辑来执行B.采用 CISC 技术的 CPU，其芯片设计复杂度更高

C.在 RISC 中，更适合采用硬布线逻辑执行指令D.采用 RISC 技术，指令系统中的指令种类和寻址方式更少

●(2008年下)计算机内存一般分为静态数据区、代码区、栈区和堆区，若某指令的操作数之一采用立即数寻址方式，则该操作数位于(1)。

(1)A.静态数据区 B.代码区 C.栈区 D.堆区

●(2007年下)在指令系统的各种寻址方式中，获取操作数最快的方式是(1)。若操作数的地址包含在指令中，则属于(2)方式。

(1)A.直接寻址 B.立即寻址 C.寄存器寻址 D.间接寻址

(2)A.直接寻址 B.立即寻址 C.寄存器寻址 D.间接寻址

●(2007年上)下面的描述中， (3) 不是RISC设计应遵循的设计原则。

(3)A.指令条数应少一些 B.寻址方式尽可能少

C.采用变长指令，功能复杂的指令长度长而简单指令长度短 D.设计尽可能多的通用寄存器

## 中央处理器

当用计算机解决某个问题时，首先必须为它编写程序。程序是一个指令序列，这个序列明确告诉计算机应该执行什么操作。在什么地方找到用来操作的数据。一旦把程序装入内存储器，就可以由计算机来自动完成取出指令和执行指令的任务。专门用来完成此项工作的计算机部件称为中央处理器，即CPU。

CPU对整个计算机系统的运行是极其重要的，它具有如下四方面的功能：

* 指令控制。程序的顺序控制，称为指令控制。由于程序是一个指令序列，这些指令的相互顺序不能任意颠倒，必须严格按程序规定的顺序进行，因此，保证机器按顺序执行程序是CPU的首要任务。
* 操作控制。一条指令的功能往往是由若干个操作信号的组成来实现的，因此，CPU管理并产生由内存取出的每条指令的操作信息，把各种操作信号送往相应的部件，从而控制这些部件按指令的要求进行动作。
* 时间控制。对各种操作实施时间上的控制，称为时间控制。因为在计算机中，各种指令的操作信号均受到时间的严格控制。另一方面，一条指令的整个执行过程也受到时间的严格控制，只有这样，计算机才能有条不紊地自动工作。
* 数据加工。对数据进行算数运算和逻辑运算处理。完成数据的加工处理，这是CPU的根本任务，因为，原始信息只有经过加工处理后才能对人们有用。

其他相应概念：

* 时钟周期：为了使计算机各部件同步工作，计算机中有一个脉冲源，通常是晶振，这个脉冲源产生主振脉冲，主振脉冲的时间间隔称为时钟周期。
* 时钟频率：指CPU晶振所产生脉冲的频率，一秒钟产生的主振次数。
* CPU周期：CPU从内存中读一个指令的最短时间称为CPU周期，一般由若干个时钟周期组成。
* 指令周期：指一条机器指令从主存中取出来到执行完毕所需要的时间，通常将分成两个阶段，**取指令、分析指令阶段和执行指令阶**，由于指令有不同的复杂程度，所以每种指令的指令周期可能不同，指令周期一般由若干个CPU周期组成。

### 指令执行的过程

在现代计算机中，一条指令周期中一般又定义几种基本的机器周期，例如取指机器周期，读存储器机器周期，写存储器机器周期和执行运算机器周期等。

任何一条指令的第一个机器周期必须是取指令机器周期，完成从主存器中取出指令，并分析指令的功能。

### CPU指令执行流程图

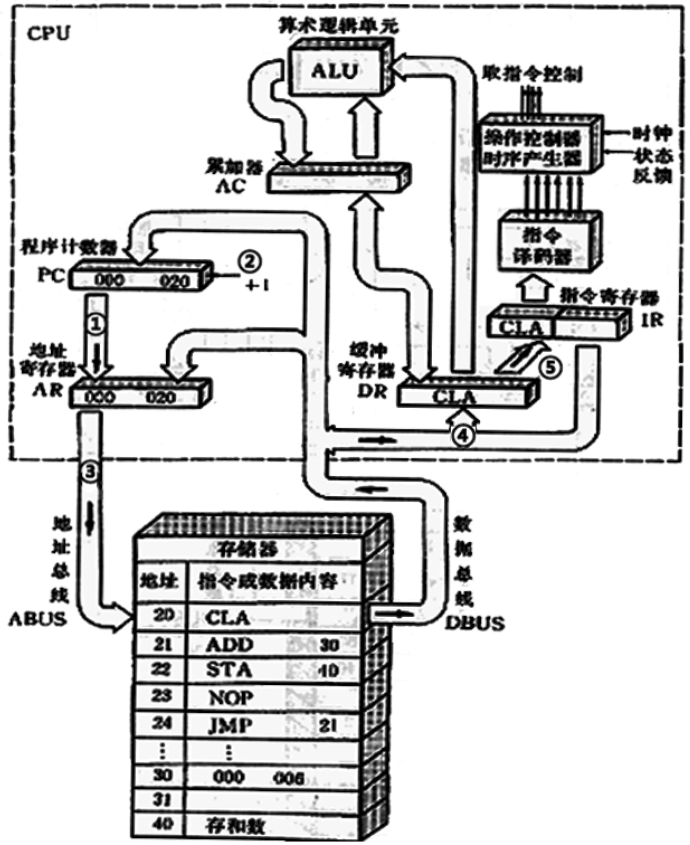


图2.19执行CLA指令过程

### CPU的组成

传统的CPU由运算器和控制器两大部分组成。

#### 控制器组成部分及主要功能

**控制器的组成部分为：**由程序计数器(PC)、指令寄存器(AR)、指令译码器(IR)、时序产生器、操作控制器、状态条件寄存器(PSW)以及中断系统组成，用来协调和指挥整个计算机系统工作。

**控制器的主要功能为：**

* 从内存中取出一条指令，并指出下一条指令在内存中的位置。
* 对指令进行译码或测试，并产生相应的操作控制信号，以便启动规定的动作，并处理异常事件。
* 指挥并控制CPU、内存和输入输出设备之间数据流动的方向。

**各部分具体功能如下：**

* 程序计数器(PC，Program Counter)：初始化存放从内存中提取程序的第一条指令地址的指令地址寄存器。由于多数情况程序是顺序执行的，所以程序计数器设计有自动加1的装置，用来存放下一条指令的地址。当出现转移指令时，需要重新程序计数器。
* 地址寄存器(AR，Address Register)：用来保存当前CPU所访问的内存单元的地址。
* 指令寄存器(IR，Instruction Register)：用来保存当前正在执行的一条指令，它的位数取决于指令字长的位数。此寄存器对于程序人员来说是“透明”的，不能直接控制它。这里的“透明”(Transparency)是计算机学科中常用的个专业术语，表示实际存在，但在某个角度看好像没有。
* 状态条件寄存器(PSW，Program Status Word)：存放各种条件内容，如运算结果进位标志，中断和系统工作状态信息等。
* 时序产生器：每条机器指令的操作过程是由指令操作流程图严格规定的，各条机器指令的指令周期中包含的机器周期数各不相同，每个机器周期包含的时钟周期也不一定相同，所以指令周期、机器周期和时钟周期等时序信号就由时序产生器根据操作流程规定产生，从而进行时序控制。
* 操作控制器：在时序信号的控制下，各条机器指令在各个机器周期的各个时钟周期中应产生哪些微操作控制信号，由指令操作流程图作出严格规定，操作控制器就根据指令流程图的安排，在各个时钟周期中中产生相应的微操作控制信号，有效完成指令的操作过程。

#### 运算器组成部分及主要功能

**运算器的组成部分：**由算术逻辑单元(ALU)、累加寄存器(AC)、数据缓冲寄存器组成，是数据加工处理部件。

**运算器的主要功能为：**

* 执行所有的算术运算
* 执行所有的逻辑运算和逻辑测试。

**各部分具体功能如下：**

* 算术逻辑单元(ALU，Arithmetic Logical Unit)：进行算术逻辑运算。
* 数据缓冲寄存器(DR，Data Register)：用来暂时存放由读出或写入的一条指令或数据字。
* 累加寄存器(AC)：一个通用寄存器，为逻辑和算术运算提供一个工作区，用来为ALU执行算术逻辑运算提供数据并暂存运算结果。

#### 地址线和数据线

* 地址线：用来传输地址信息，CPU在内存或硬盘里面寻找一个数据时，通过地址线找到该数据在内存中的存储单元，然后再通过数据线将数据取出来，地址线的根数决定了它的寻址能力，20根能寻址1M，这个空间是CPU可以访问的空间，
* 数据线：用来传输数据信息，通常与微处理的字长相一致。例如Intel 8086微处理器字长16位，其数据总线宽度也是16位。

#### CPU的主要寄存器

程序计数器(PC,Program Counter)、地址寄存器(AR,Address Register)、数据缓冲寄存器(DR,Data Register)、指令寄存器(IR,Instruction Register)、状态条件寄存器(psw)、累加寄存器(AC)。

###### 历年试题

●(2019年上)计算机执行指令的过程中，需要由 (1) 产生每条指令的操作信号并将信号送往相应的部件进行处理，以完成指定的操作。

(1)A.CPU的控制器 B.CPU的运算器 C.DMA控制器 D.Cache控制器

● (2018年下)CPU在执行指令的过程中，会自动修改 (1) 的内容，以使其保存的总是将要执行的下一条指令的地址。

(1) A.指令寄存器 B.程序计数器 C.地址寄存器 D指令译码器

●(2017年上)CPU:执行算术运算或者逻辑运算时，常将源操作数和结果暂存在(1)中。

(1)A.程序计数器 (PC) B.累加器 (AC) C.指令寄存器 (IR) D.地址寄存器 (AR)

●(2016年下)在程序运行过程中，CPU需要将指令从内存中取出并加以分析和执行。CPU依据\_(1)\_来区分在内存中以二进制编码形式存放的指令和数据。

(1)A.指令周期的不同阶段 B.指令和数据的寻址方式

C.指令操作码的译码结果 D.指令和数据所在的存储单元

●(2016年下)计算机在一个指令周期的过程中，为从内存读取指令操作码，首先要将(2)的内容送到地址总线上。

(2)A.指令寄存器(IR) B.通用寄存器(GR) C.程序计数器(PC) D.状态寄存器(PSW)

●(2015年上)计算机中CPU对其访问速度最快的是 (2)。

(2)A.内存 B.Cache C.通用寄存器 D.硬盘

●(2014年下)属于CPU中算术逻辑单元的部件是 (3) 。

(3)A.程序计数器 B.加法器 C.指令寄存器 D.指令译码器

●(2014年上)在CPU中，常用来为ALU执行算术逻辑运算提供数据并暂存运算结果的寄存器是(1)。

(1)A.程序计数器 B.状态寄存器 C.通用寄存器 D.累加寄存器

●(2013年下)指令寄存器的位数取决于 (2) 。

(2)A.存储器的容量 B.指令字长 C.数据总线的宽度 D.地址总线的宽度

●(2012年下)在CPU中， (1) 不仅要保证指令的正确执行，还要能够处理异常事件。

(1) A.运算器 B.控制器 C.寄存器组 D.内部总线

●(2011年下)在CPU的寄存器中， (5) 对用户是完全透明的。

(5)A.程序计数器 B.指令寄存器 C.状态寄存器 D.通用寄存器

●(2011年下)CPU中译码器的主要作用是进行 (6) 。

(6)A.地址译码 B.指令译码 C.数据译码 D.选择多路数据至ALU

●(2011年上)在CPU中用于跟踪指令地址的寄存器是 (1) 。

(1)A.地址寄存器(MAR) B.数据寄存器(MDR) C.程序计数器(PC) D.指令寄存器(IR)

●(2010年下)若内存容量为4GB，字长为32，则 (3) 。

(3)A.地址总线和数据总线的宽度都为32 B.地址总线的宽度为30，数据总线的宽度为32

C.地址总线的宽度为30，数据总线的宽度为8 D.地址总线的宽度为32，数据总线的宽度为8

●(2010年下)编写汇编语言程序时，下列寄存器中程序员可访问的是 (5) 。

(5)A.程序计数器(PC) B.指令寄存器(IR) C.存储器数据寄存器(MDR) D.存储器地址寄存器(MAR)

●(2010年上)为实现程序指令的顺序执行，CPU (1) 中的值将自动加1。

(1)A.指令寄存器(IR) B.程序计数器(PC) C.地址寄存器(AR) D.指令译码器(ID)

●(2010年上)计算机指令一般包括操作码和地址码两部分，为分析执行一条指令，其 (5) 。

(5)A.操作码应存入指令寄存器(IR)，地址码应存入程序计数器(PC)

B.操作码应存入程序计数器(PC)，地址码应存入指令寄存器(IR)

C.操作码和地址码都应存入指令寄存器(IR)

D.操作码和地址码都应存入程序计数器(PC)

●(2009年下)以下关于 CPU 的叙述中，错误的是(1) 。

(1)A.CPU 产生每条指令的操作信号并将操作信号送往相应的部件进行控制

B.程序计数器 PC 除了存放指令地址，也可以临时存储算术/逻辑运算结果

C.CPU 中的控制器决定计算机运行过程的自动化

D.指令译码器是 CPU 控制器中的部件

●(2008年下)在CPU中， (3) 可用于传送和暂存用户数据，为ALU执行算术逻辑运算提供工作区。

(3)A.程序计数器 B.累加寄存器 C.程序状态寄存器 D.地址寄存器

●(2008年上)在计算机体系结构中，CPU 内部包括程序计数器 PC、[存储](http://www.educity.cn/incsearch/search.asp?key=%B4%E6%B4%A2)器数据寄存器 MDR、指令寄存器IR 和存储器地址寄存器MAR 等。若CPU 要执行的指令为： MOV R0, #100(即将数值100传送到寄存器R0中)，则CPU 首先要完成的操作是 (1) 。

(1)A.100→R0 B.100→MDR C.PC→MAR D.PC→IR

●(2008年上)CPU 中的数据总线宽度会影响 (4) 。

(4)A.内存容量的大小 B.系统的运算速度 C.指令系统的指令数量 D.寄存器的宽度

●(2007年上)(1) 不属于计算机控制器中的部件。

(1)A.指令寄存器IR B.程序计数器PC C.算术逻辑单元ALU D.程序状态字寄存器PSW

## 中断系统

#### 中断系统的定义

计算机工作的过程是执行指令的过程，当出现某种不可预知的事件时，需要打断CPU正在执行的工作，让CPU去完成更加重要的任务，待事件处理完成后，又回到原来程序的断点继续执行，如图2.20所示。

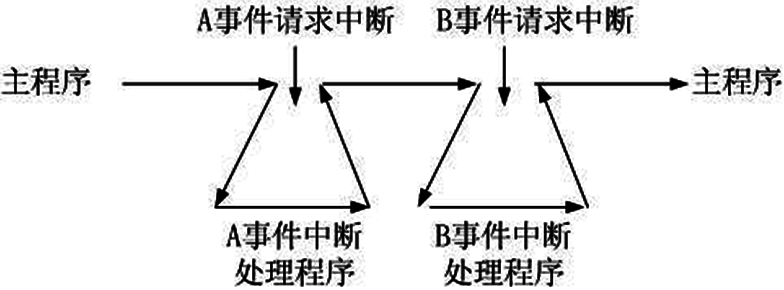


图2.20中断处理示意图

当事件发生时向CPU发出的请求，称为中断请求，对中断请求进行响应并处理的部件，称为中断系统。

#### 中断处理过程

CPU在执行完一条指令时，如果外设向CPU发出中断请求，那么CPU在满足响应条件的情况下，将发出中断响应信号，同时关闭中断，表示不再受理另外一个设备的中断，并保存被中断程序的断点(即程序计数器PC的内容，寄存器的内容以及状态等信息)到堆栈中。CPU将在**中断向量表中**寻找中断请求源，找出该中断服务程序的入口地址(中断向量值)，并装入PC中，准备执行相应的中断服务程序。**以上过程由硬件完成，这个过程消耗的时间为中断响应时间。**

待中断处理程序完成后，将刚才保存的现场恢复，继续执行被中断的程序，**整个过程消耗的时间为中断处理时间。**

有以下几个问题加以说明：

##### 外界中断请求是随机的，但CPU只有在当前一条指令执行完毕后，即转入公操作时才受理设备的中断请求。

##### 为了在中断服务程序执行完毕后，能够正确的返回到原来主程序被中断的断点(PC等内容)而继续执行主程序，必须把程序计数器PC的内容，以及当前指令执行结束后CPU的状态等都保存到**堆栈**中去，这些叫保护现场。

##### 当CPU响应中断后，为避免多个中断服务程序出现混乱，所以在CPU中断管理部件中必须有一个“中断屏蔽”触发器，置“1”为设置屏蔽，置“0”为取消屏蔽，只有为“0”时，CPU才能受理中断。当CPU接收中断服务请求并作出响应时，它会把“中断屏蔽”标志置为“1”，表示不再受理另外新的中断源发来的中断请求。在CPU把中断服务程序执行完毕以后，再设置“中断屏蔽”标志置为“0”，即开放中断。

##### 中断处理过程由硬件和软件结合完成，其中响应中断，找出中断源和保护PC由硬件实现，而中断服务程序由指令序列实现。

###### 历年试题

●(2018年上)计算机运行过程中，遇到突发事件，要求CPU暂时停止正在运行的程序，转去为突发事件服务，服务完毕，再自动返回原程序继续执行，这个过程称为(2)，其处理过程中保存现场的目的是(3)。

(2)A.阻塞 B.中断 C.动态绑定 D.静态绑定

(3)A.防止丢失数据. B.防止对其他部件造成影响

C.返回去继续执行原程序 D.为中断处理程序提供数据

●(2015年上)计算机中CPU的中断响应时间指的是 (4)的时间。

(4)A.从发出中断请求到中断处理结束 B.从中断处理开始到中断处理结束

C.CPU分析判断中断请求 D.从发出中断请求到开始进入中断处理程序

●(2013年上)为了便于实现多级中断嵌套，使用(3) 来保护断点和现场最有效

(3)A.ROM B.中断向量表 C.通用寄存器 D.堆栈

● 中断向量可提供(2) 。

(2)A.IO设备的端口地址 B.所传送数据的起始地址 C.中断服务程序的入口地址 D.主程序的断点地址

## 流水线操作

### 指令流水线

把指令的执行过程细分为**取指令、指令译码、执行、写回**4个子过程，并改进运算器的结构以加快其执行子过程。如图2.21非流水线时空图所示，CPU在将指令I1的四个子过程都执行完毕后才去执行指令I2，这种执行过程称为串行执行的非流水线执行方式，效率较低；如图2.22标量流水线时空图所示，在第1个单位时间内，CPU在完成I1的取指令(IF)子过程；在第2个单位时间内执行I1的指令译码(ID)子过程，同时执行I2指令的取指令(IF)子过程，以此类推，图1.24所示中的5条指令都是在流水线式并行执行各个指令的子过程。

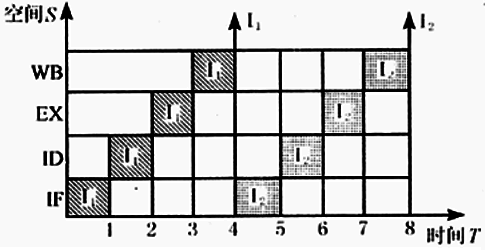


图2.21非流水线时空图

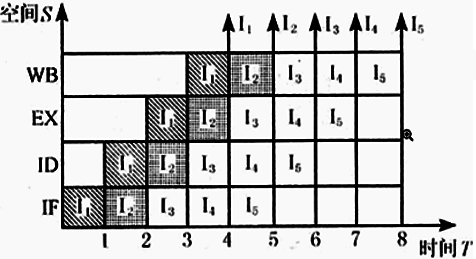


图2.22标量流水线时空图

指令流水线的概念就是将一条指令分解成一连串执行的子过程，在CPU中变一条指令的串行执行子过程为若干条指令的子过程在CPU中重叠(并行)执行。从图2.21、图2.22可知，在8个单位时间内，非流水线执行方式仅执行完2条指令，而流水线方式则执行完5条指令，流水线执行方式明显效率高得多。

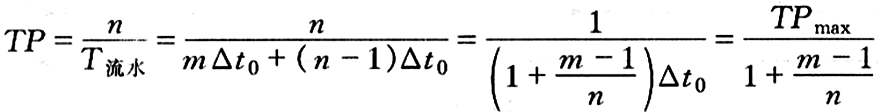
### 流水线处理机的主要性能

#### 吞吐率

* 吞吐率(Throughput Rate)：是指单位时间内流水线所完成的任务数或输出结果的数量。对指令而言就是单位时间内执行的指令数。
* 最大吞吐率：指流水线在连续流动达到稳定状态后所得到的吞吐率。
* 实际吞吐率：指实际工作中流水线的吞吐率，因流水线开始时总需要一段建立时间，另有一些原因导致流水线会断流，流一段时间，停一段时间，因此流水线的实际吞吐率TP总比TPmax小的多。
* 加速比(SP,Speedup Ratio)：表示流水线方式相对非流水线顺序串行方式速度提高的比值。

对于执行各子任务的时间相等且为Δt0的执行情况，如图2.23所示，设流水线由m段(这里m为4，即S1、S2、S3、S4四个执行指令子阶段)组成，完成n个任务所需时间为：*T*流水=mΔt0+(n-1)Δt0。当图中流水线到达稳定状态后，相当于在一个Δt0时间段内有四个阶段同时都在进行，即最大吞吐率为：

最大吞吐率：TPmax=1/Δt0

实际吞吐率为：

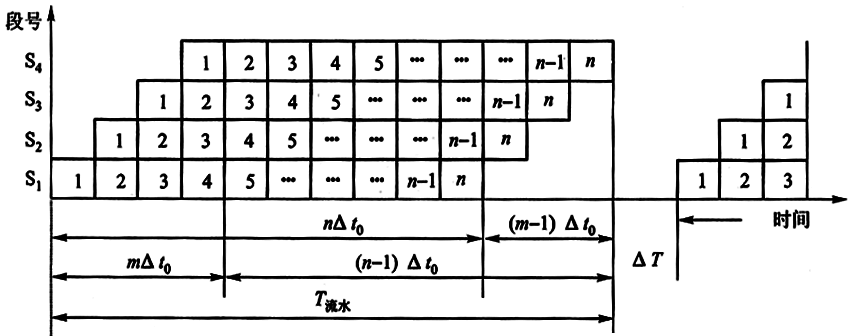


图2.23各子阶段耗时相等情况下n条指令执行时空图

因非流水线顺序串行方式连续完成n个任务需要时间n\*m\*Δt0，故加速比为：

Sp=n\*m\*Δt0/(m\*Δt0+(n-1)Δt0)=m/(1+(m-1)/n)

即流水线各子阶段相同时，仅当n>>m时，即流水线连续输入的任务数n远大于流水线的子过程个数m时，其加速比才能趋于最大值。

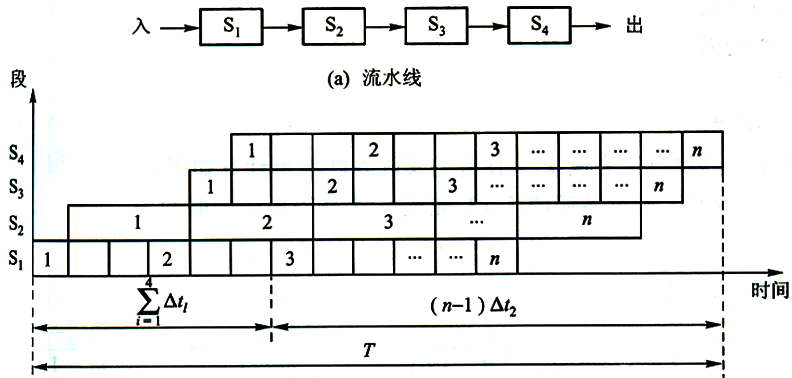
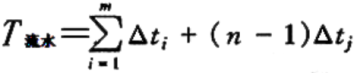
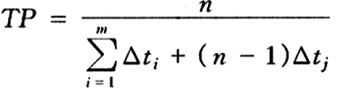


图2.24各子阶段耗时不相等情况下n条指令执行时空图

对于流水线的子过程所用时间不一样长，如图2.24所示。流水线中△t2＝3△t1＝3△t3＝3△t4＝3△t0，则完成n个任务所需时间为：，其中为运行完单条指令需要的时间，△tj表示最长的子任务执行时间。

最大吞吐率为：TPmax＝1/max{△ti}。

实际的吞吐率为：，

加速比为：Sp=n\*/

* 流水线瓶颈子过程

当子过程所用时间不一样时，需要在子过程之间设置接口锁存器，暂时存储上一子过程的数据，并且需要让各锁存器再接受同一时钟脉冲同步，时钟脉冲的周期应该定义为max{△tj }，则流水线中经过时间最长的子过程称为瓶颈子过程。

#### 流水线的控制

* 同步流动方式：也称为顺序流动方式，指让指令流出流水线的顺序保持与流入流水线的顺序一致。
* 异步流动方式：指让指令流出流水线的顺序保持与流入流水线的顺序不一致。

###### 历年试题

●(2018年下)下列关于流水线方式执行指令的叙述中，不正确的是(6) 。

(6)A.流水线方式可提高单条指令的执行速度 B.流水线方式下可同时执行多条指令

C.流水线方式提高了各部件的利用率 D.流水线方式提高了系统的吞吐率

●(2018年上)流水线的吞吐率是指单位时间流水线处理的任务数，如果各段流水的操作时间不同，则流水线的吞吐率是(6)的倒数。

(6)A.最短流水段操作时间 B.各段流水的操作时间总和

C.最长流水段操作时间 D.流水段数乘以最长流水段操作时间

●(2017年下)某四级指令流水线分别完成取指、取数、运算、保存结果四步操作。若完成上述操作的时间依次为8ns、9ns、4ns、8ns，则流水线的操作周期应至少为 (2) ns。

(2)A.4 B.8 C.9 D.33

●(2016年下)将一条指令的执行过程分解为取指、分析和执行三步，按照流水方式执行，若取指时间=4、分析时间=2、执行时间=3，则执行完100条指令，需要的时间为\_\_(5)\_\_。

(5)A.200 B.300 C.400 D.405

●(2014年上)通常可以将计算机系统中执行一条指令的过程分为取指令、分析和执行指令3步，若取指令时间为4△t，分析时间为2△t，执行时间为3△t，按顺序方式从头到尾执行完600条指令所需时间为 (4)△t，若按照执行第i条、分析第i+l条、读取第i+2条重叠的流水线方式执行指令，则从头到尾执行完600条指令所需时间为 (5) △t。

(4)A.2400 B.3000 C.3600 D.5400

(5)A.2400 B.2405 C.3000 D.3009

●(2013年下)若指令流水线由4段组成，各段所需要的时间如下图所示。连续输入8条指令时的吞吐率(单位时间内流水线索完成的任务数或输出的结果数)为 (4) 。



(4)A.8/56Δt B.8/32Δt C.8/28Δt D.8/24Δt

●(2007年上)指令流水线将一条指令的执行过程分为四步，其中第1、2和4步的经过时间为△t，如下图所示。若该流水线顺序执行50条指令共用153△t，并且不考虑相关问题，则该流水线的瓶颈第3步的时间为 (5) △t 。



(5)A.2 B.3 C.4 D.5

●(2015年上)以下关于指令流水线性能度量的叙述中，错误的是 (6) 。

(6)A.最大吞吐率取决于流水线中最慢一段所需的时间 B.如果流水线出现断流，加速比会明显下降

C.要使加速比和效率最大化应该对流水线各级采用相同的运行时间D.流水线采用异步控制会明显提高其性能

## 总线结构

总线是连接计算机有关部件的一组信号线，是计算机中用来传送信息代码的公共通道。**采用总线结构主要有以下优点：**简化系统结构，便于系统设计制造；大大减少了连线数目，便于布线，减小体积，提高系统的可靠性；便于接口设计，所有与总线连接的设备均采用类似的接口；便于系统的扩充、更新与灵活配置，易于实现系统的模块化；便于设备的软件设计，所有接口的软件就是对不同的口地址进行操作；便于故障诊断和维修，同时也降低了成本。

### 总线的定义与分类

#### 总线分类

广义地讲，任何连接两个以上电子元器件的导线都可称为总线。通常分为三类总线。

##### 内部总线。用于芯片一级的互连，分为芯片内总线和元件级总线。

* 芯片内总线，用于集成电路芯片内部各部分的连接，如CPU内部各寄存器的连接
* 元件级总线，用于同一块电路板内各元器件的连接。

##### 系统总线。用于插件板一级的互连，用于构成计算机各组成部分(CPU、内存、接口等)的连接，

##### 外总线。又称通信总线，用于设备一级的互连，是计算机与外设或计算机与计算机的连接或通信线路，如计算机和打印机之间的连线。

#### 总线特性.

##### 物理特性：指总线的物理连接方式，包括总线的根数，总线的插头插座的形状，引脚线的排列方式等。

##### 功能特性：描述总线中每一根线的功能。

* 地址总线的宽度指明了总线能够直接访问的存储器地址空间范围。
  + - 地址总线(Address Bus，AB)是在计算机系统各个部件之间传输地址信息的信号线，用来规定数据总线上的数据来自何处或将被送往何处。
    - 地址总线是单向的。
    - 如果CPU要从存储器中读取一个信息，将要读取的信息的存储器地址放到地址总线上，从给定的存储器地址中取出所需要的信息。
    - 地址总线的宽度决定了计算机系统能够使用的最大的存储器容量。
* 数据总线的宽度指明了访问一次存储器或外设所能交换数据的位数。
  + - 数据总线(Data Bus，DB)是在计算机系统各部件之间传输数据信息的信号线，数据总线是双向的。
    - 数据线的根数称为数据总线的宽度，通常为8、16、32或64，由于每根数据线每次传送1位二进制数，所以数据线的根数决定了每次能同时传送的二进制的位数，如果数据总线的宽度为8位，而指令长度为16位，那么在每个指令周期中需要两次访问存储器才能取回完整的16位指令。
* 控制总线包括CPU发出的各种控制命令(如存储器读/写、I/O读/写等)，请求信号与仲裁信号，外设与CPU的时序同步信号，中断信号，DMA控制信号等等。
  + - 作用是对数据总线、地址总线的访问及使用实施控制。
    - 控制线中每一根线都是单向的。
    - 用来指明数据传送的方向、中断请求和定时控制等。
    - 控制总线上传输的控制信息，其作用就是在计算机系统各个部件之间发送操作命令和定时信息。
    - 命令信息规定了要执行的具体操作。
    - 定时信息则规定了数据信息和地址信息的时效性。

##### 电气特性：定义每一根线上信号的传递方向及有效电平范围，送入CPU的信号叫输入信号(IN)，从CPU发出的信号叫输出信号(OUT)，例如，IBM PC/XT总线的A0～A19是地址输出线，D0～D7是双向数据线，既可作为数据输入线又可作为数据输出线，总线的电平都符合TTL电平的定义。

##### 时间特性：定义每根线在什么时间有效，即规定总线上各信号有效的时序关系。

#### 总线的主要参数

##### **总线宽度：**总线能同时传送的数据的二进制位(bit)数，如16位总线、32位总线指的就是总线具有16位或32位的数据传输能力。

##### **总线频率：**总线的实际工作频率，也就是一秒钟传输数据的次数，是总线工作速度的一个重要参数，工作频率越高，速度越快，总线频率通常用MHz表示：如33MHz、100 MHz、400 MHz、800 MHz等，1Hz = 1/s

##### **总线带宽：**总线本身所能达到的最高数据传输速率，单位是兆字节每秒(MB/s)，

总线带宽与总线宽度和总线频率的关系：

例题：某总线在一个总线周期中并行传送32位数据，假设一个总线周期等于一个总线时钟周期，总线时钟频率为33MHz，总线带宽是多少?

【解】：⑴总线带宽 =(32bit/8bit/B)×33MHz=(32bit/8bit/B)×33M/s=132MB/s

#### 不同时期总线的内部结构

##### 早期总线的内部结构

总线是处理器芯片引脚的延伸，是处理器与I/O设备适配器的通道，这种简单的总线一般由50~100根信号线组成，按功能特性可分为：数据(总)线、地址(总)线、控制(总)线，如图2.25所示。



图2.25早期系统总线概图

随着计算机技术的发展，这种简单总线结构暴露出一些缺点：

第一，CPU是总线上的惟一主控者，即使后来增加了具有简单仲裁逻辑的DMA控制器以支持DMA传送，但是仍不能满足多CPU环境的要求；

第二，总线信号是CPU引脚信号的延伸，所以总线结构与CPU紧密相关，通用性较差。

##### 当代总线是一些标准总线



图2.26当代系统总线概图

如图2.26所示，CPU与Cache作为一个模块与总线相连，系统中允许存在多个这样的处理器模块；总线控制器负责在几个总线请求者之间进行协调与仲裁；整个总线结构分成四个部分：数据传送总线；仲裁总线；中断和同步总线；公用线。

* 数据传送总线：由地址线、数据线、控制线组成，结构与早期总线类似，有32条地址线，32或64条数据线
* 仲裁总线：包括总线请求线和总线授权线
* 中断和同步总线：处理带优先级的中断操作，包括中断请求线和中断认可线
* 公用线：包括时钟信号线、电源线、地线、系统复位线以及加电或断电的时序信号线等

#### 总线的连接方式

根据连接方式的不同，单机系统中采用的总线结构可分成三种基本类型：单总线结构、双总线结构、三总线结构；

##### 单总线结构：在许多单处理器的计算机中，使用一条单一的系统总线来连接CPU、主存和I/O设备。



图2.27单总线结构

如图2.27所示，在单总线系统中，对输入/输出设备的操作与主存的操作方法完全一样，当CPU把指令的地址字段送到总线上时，如果该地址字段对应的地址是主存地址，则主存予以响应，在CPU和主存间发生数据传送，数据传送的方向由指令操作码决定。

如果该地址字段对应的地址是外围设备地址，则外围设备予以响应，在CPU和对应的外围设备间发生数据传送，数据传送的方向也由指令操作码决定。

* 单总线结构的优点：容易扩展成多CPU系统，只要在系统总线上挂接多个CPU即可。
* 单总线结构的缺点：由于所有逻辑部件都挂在同一个总线上，因此总线只能分时工作，即某一个时间只能允许一对部件之间传送数据，这就使信息传送的吞吐量受到限制。

##### 双总线系统结构

如图2.28所示，在CPU和主存间设置了一组高速的存储总线，存储总线与存储器换信息，减轻系统总线的负担，同时主存仍可通过系统总线与外设进行DMA操作，而不必经过CPU，当然，这种双总线系统是以增加硬件为代价的。



图2.28双总线系统结构

##### 三总线系统结构

如图2.29所示，在双总线系统基础上增加I/O总线形成的，系统总线是CPU、主存 和通道(IOP)之间进行数据传送的公共通路，I/O总线是多个外围设备与通道之间进行数据传送的公共通路。



图2.29三总线系统结构

DMA方式中，外设与主存间直接交换数据而不经过CPU，从而减轻了CPU对数据输入输出的控制，通道实际上是一台具有特殊功能的处理器，又称为IOP(I/O Processor，I/O处理器)，分担CPU的一部分功能，实现对外设的统一管理，完成外设与主存之间的数据传送

### 系统总线标准

常见的系统总线如下：

* ISA总线：是工业标准总线，包括24个地址线，16条数据线，控制总线(内存读写、接口读写、中断响应等)。
* EISA总线：是在ISA基础上发展起来的32位总线。该总线定义32位地址线，32位数据线，以及其他控制信号。
* PCI总线：是目前微型机上广泛采用的内总线。PCI总线的工作与处理器的工作相互独立。PCI上设备是即插即用的。
* CPU总线：也称CPU-存储器总线，是一个64位数据线和32位地址线的总线，连接有内存、缓存和他们的控制器，CPU是这条线的主控者。

### 外总线标准

外总线标准多样，仅介绍以下几种。

* RS－232C总线：是一条串行外总线。
* SCSI总线：小型计算机系统接口是一条并行外总线，广泛用于连接软硬磁盘、光盘等。该总线上最多可接63种外设。
* USB：通用串行总线。由4条信号线组成，两条用于传送数据，另外两条传送电流。支持即插即用和热插拔。
* IEEE－1394：串行外总线，由6条信号线组成。两条传送数据，两条传送控制信号，2条传送电流。支持即插即用和热插拔。
* 如用于连接并行打印机的Centronics总线；
* 用于硬磁盘接口的IDE等。

●(2016年上)以下关于总线的叙述中，不正确的是(6)。

(6)A.并行总线适合近距离高速数据传输 B.串行总线适合长距离数据传输

C.单总线结构在一个总线上适应不同种类的设备，设计简单且性能很高

D.专用总线在设计上可以与连接设备实现最佳匹配

●(2015年上)总线宽度为32bit，时钟频率为200MHz,若总线上每5个时钟周期传送一个32bit的字，则该总线的带宽为(5)MB/S。

(5)A.40 B.80 C.160 D.200

●(2014年下)三总线结构的计算机总线系统由 (1) 组成。

(1)A.CPU总线、内存总线和IO总线 B.数据总线、地址总线和控制总线

C.系统总线、内部总线和外部总线 D.串行总线、并行总线和PCI总线

● (6) 不属于系统总线。

(6)A.ISA B.EISA C.SCSI D.PCI

●(2011年下)总线复用方式可以 (4) 。

(4)A.提高总线的传输带宽 B.增加总线的功能 C.减少总线中信号线的数量 D.提高CPU利用率

●(2011年上)在计算机系统中采用总线结构，便于实现系统的积木化构造，同时可以 (3)。

(3)A.提高数据传输速度 B.提高数据传输量 C.减少信息传输线的数量 D.减少指令系统的复杂性

●(2009年上)处理机主要由处理器、存储器和总线组成，总线包括访(4 )。

(4)A.数据总线、地址总线、控制总线 B.并行总线、串行总线、逻辑总线

C.单工总线、双工总线、外部总线 D.逻辑总线、物理总线、内部总线

## 输入输出系统

#### 输入输出设备的编址方式

计算机程序要访问外设，需要通过地址总线访问各个接口，外设接口的编址方法最常见的有以下两种：

##### 独立编址方式：外设接口地址和主存地址是分开的，是完全独立的两个地址空间。访问外设指令和内存指令不同，依靠指令来确定访问对象。外设访问指令常常以汇编指令的方式编写，和CPU有密切的关系，对程序的可移植性造成了影响。

##### 统一编址方式：内存地址和外设接口地址在一个公共的地址空间内。依靠不同的内存地址代码来确定访问对象，而不需要单独编写指令，降低了软件编写的复杂度。

#### 输入输出设备的控制方式

##### 程序查询方式：最简单的方式，也是数据传输速度最低的方式。这种方式CPU定时查询外设的状态，发现外设就绪，就开始和外设进行输入输出操作和处理。缺点：当输入/输出控制器和外设交换数据时，CPU必须等待。

##### 中断方式：在I/O系统中，当CPU执行到I/O请求时，向输入输出控制器发出相应的指令后，CPU并不等待，而是继续执行其他操作，由输入输出控制器和外设进行通信，当数据从寄存器写到外设或从外设写入寄存器完毕后，输入输出控制器向CPU发出中断请求，CPU响应中断，并进行相应的处理。由于CPU无须等待输入输出控制器和外设的数据交换，提高了系统的效率。

虽然中断驱动I/O 比程序I/O 方式更有效，但它仍是以字(节)为单位进行I/O的，每当完成一个字(节)的I/O 时，控制器便要向CPU 请求一次中断。换言之，采用中断驱动I/O 方式时的CPU是以字(节)为单位进行干预的。如果将这种方式用于块设备的I/O，例如，为了从磁盘中读出1 KB的数据块，需要中断CPU 1K 次。为了进一步减少CPU 对I/O 的干预而引入了直接存储器访问方式DMA(Direct Memory Access)。程序I/O和中断驱动方式的流程如图2.30所示。

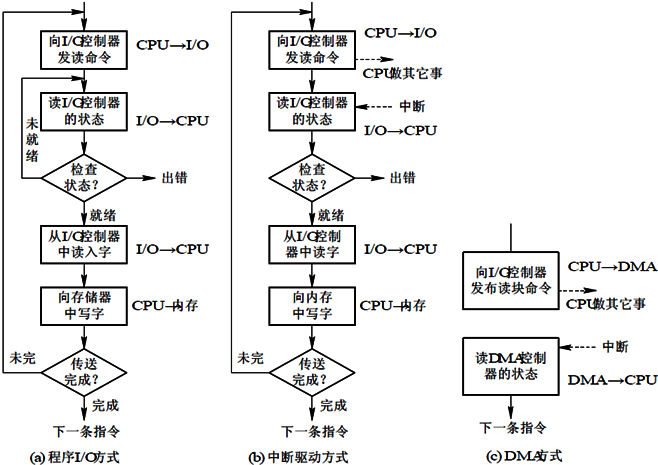


图2.30程序I/O和中断驱动方式的流程

##### DMA(Direct Memory Access)方式：是指内存和I/O设备间传送一个数据块的过程中，不需要CPU的任何干涉，是一种快速传递大量数据常用的技术。

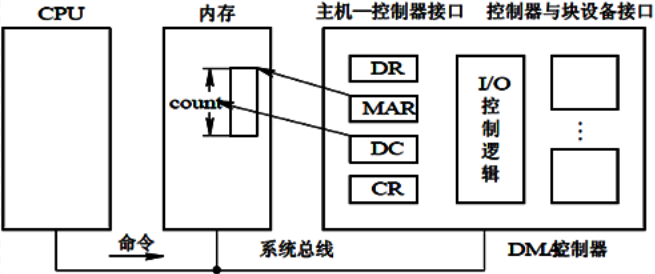


图2.31DMA控制器的组成

为了实现在主机与控制器之间成块数据的直接交换，必须在DMA控制器中设置如下四类寄存器：

* 命令/状态寄存器(CR)。用于接收从CPU 发来的I/O 命令，或有关控制信息，或设备的状态。
* 内存地址寄存器(MAR)。在输入时，它存放把数据从设备传送到内存的起始目标地址；在输出时，它存放由内存到设备的内存源地址。
* 数据寄存器(DR)。用于暂存从设备到内存，或从内存到设备的数据。
* 数据计数器(DC)。存放本次CPU要读或写的字(节)数。

工作过程大致如下所述：

1. 外设向CPU申请DMA传送的请求；
2. 获CPU允许后，CPU放弃对总线的控制权，并停止访问内存，DMA控制器接管系统总线的控制权；
3. 在DMA控制器的控制下，存储器和外设之间直接进行数据传送，在传送过程中不需要CPU的参与。仅在开始时提供要传送的数据的起始地址和数据长度；
4. 传送结束后，向CPU返回DMA操作完成信号，CPU重新获取系统总线的控制权。

##### 通道方式：通道方式是DMA方式的发展，它可进一步减少CPU的干预，即把对一个数据块的读(或写)为单位的干预减少为对一组数据块的读(或写)及有关的控制和管理为单位的干预。同时，又可实现CPU、通道和I/O 设备三者的并行操作设置通道后，CPU只需向通道发出I/O命令，通道收到命令后，从主存中取出本次I/O要执行的通道程序，并执行，仅当通道完成了I/O任务后，才向CPU发出中断信号。

###### 历年试题

●(2019年上)DMA控制方式是在 (2) 之间直接建立数据通路进行数据的交换处理。

(2)A.CPU与主存 B.CPU与外设 C.主存与外设 D.外设与外设

●(2017年下)计算机运行过程中，CPU需要与外设进行数据交换。采用(6)控制技术时，CPU与外设可并行工作。

(6)A.程序查询方式和中断方式 B.中断方式和DMA方式

C.程序查询方式和DMA方式 D.程序查询方式、中断方式和DMA方式

●(2017年上)计算机系统中常用的输入/输出控制方式有无条件传送、中断、程序查询和 DMA方式等。当采用( 3 )方式时，不需要 CPU 执行程序指令来传送数据。

(3) A.中断 B.程序查询 C.无条件传送 D.DMA

●(2015年下)CPU是在­­­(1)结束时响应DMA请求的。

(1)A.一条指令执行 B.一段程序 C.一个时钟周期 D.一个总线周期

●(2013年上)DMA工作方式下，在(4)之间建立了直接的数据通路。

(4) A.CPU与外设 B.CPU与主存 C.主存与外设 D.外设与外设

●(2011年下)若某计算机系统的I/O接口与主存采用统一编址，则输入输出操作是通过(2) 指令来完成的。

(2)A.控制 B.中断 C.输入输出 D.访存

●(2010年上)以下关于计算机系统中断概念的叙述中，正确的是 (3) 。

(3)A.由I/O设备提出的中断请求和电源掉电都是可屏蔽中断

B.由I/O设备提出的中断请求和电源掉电都是不可屏蔽中断

C.由I/O设备提出的中断请求是可屏蔽中断，电源掉电是不可屏蔽中断

D.由I/O设备提出的中断请求是不可屏蔽中断，电源掉电是可屏蔽中断

●(2008年下)下面关于在I/O设备与主机间交换数据的叙述，(4)是错误的。

(4)A.中断方式下，CPU需要执行程序来实现数据传送任务

B.中断方式和DMA方式下，CPU与I/O设备都可同步工作

C.中断方式和DMA方式中，快速I/O设备更适合采用中断方式传递数据

D.若同时接到DMA请求和中断请求，CPU优先响应DMA请求

## 多处理机与并行处理

#### 计算机系统结构的分类

* 指令流(Instruction Stream)：机器执行的指令序列。
* 数据流(Data Stream)：由指令流调用的数据序列，包括输入数据和中间结果。

Flynn于1972年提出了计算平台的Flynn分类法，主要根据指令流和数据流来分类，共分为四种类型的计算平台：

* 单指令流单数据流(SISD)：是一种传统的串行计算机，它的硬件不支持任何形式的并行计算，所有的指令都是串行执行。并且在某个时钟周期内，CPU只能处理一个数据流。因此这种机器被称作单指令流单数据流机器。早期的计算机都是SISD机器，如冯诺.依曼架构，如IBM PC机，早期的巨型机和许多8位的家用机等。
* 单指令流多数据流(SIMD)：是采用一个指令流处理多个数据流。这类机器在数字信号处理、图像处理、以及多媒体信息处理等领域非常有效。
* 多指令流单数据流(MISD)：是采用多个指令流来处理单个数据流。由于实际情况中，采用多指令流处理多数据流才是更有效的方法，因此MISD只是作为理论模型出现，没有投入到实际应用之中。
* 多指令流多数据流(MIMD)：MIMD机器可以同时执行多个指令流，这些指令流分别对不同数据流进行操作。最新的多核计算平台就属于MIMD的范畴，例如Intel和AMD的双核处理器等都属于MIMD。

#### 阵列处理机

又称并行处理机，它将重复设置的多个处理单元(PU)按一定方式连接成阵列，在一个控制部件(CU)控制下，执行同样的指令，对分配给自己的不同的数据进行同步处理，并行地完成一条指令所规定的操作。特别适应特定的运算，比如矩阵。阵列处理机是一种单指令流多数据流(SIMD)计算机。通过资源重复实现并行性。阵列处理机如图2.32所示。

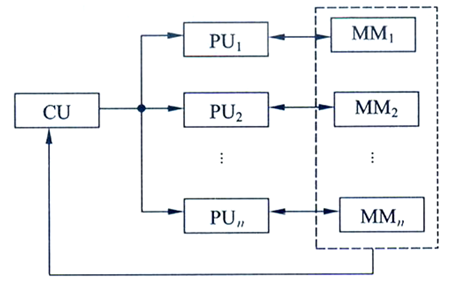


图2.32SIMD模型

#### 多处理机

多处理机系统是多台处理机组成的系统，每台处理机有属于自己的控制部件，可以执行独立的程序，同时对不同的数据进行不同的处理，指令流和数据流都存在并行，它是多指令流和多数据流(MIMD)计算机。

#### 并行处理机

并行处理机与采用流水线结构的单机系统都是单指令流多数据流计算机，区别是并行处理机采用资源重复技术，而流水线结构的单机系统采用时间重叠技术。

●(2008年上)利用高速通信网络将多台高性能工作站或微型机互连构成机群系统，其[系统结构](http://www.educity.cn/incsearch/search.asp?key=%CF%B5%CD%B3%BD%E1%B9%B9)形式属于 (5) 计算机。

(5)A.单指令流单数据流(SISD) B.多指令流单数据流(MISD)C.单指令流多数据流(SIMD) D.多指令流多数据流(MIMD)

# 计算机可靠性与系统性能基础知识

## 计算机可靠性与系统性能

#### 计算机可靠性概述

计算机系统的可靠性是指从它开始运行(t＝0)到某时刻t这段时间内能正常运行的概率，用R(t)表示。

* 失效率是指单位时间内失效的元件数与元件总数的比例，即器件的失效概率，用lambad表示。当为常数时，可靠性与失效率的关系为：R(t)=e-t。
* 平均无故障时间是指两次故障之间系统能正常工作的平均时间，用MTBF表示：MTBF＝1/
* 平均修复时间是指从故障发生到及其修复平均所需时间，用MTRF表示。
* 计算机的可用性指计算机的使用效率，它以系统在执行任务的任意时刻能正常工作的概率A表示：A=平均无故障时间/(平均无故障时间+平均修复时间)= MTBF/(MTBF+MTRF)。

#### 计算机可靠性模型

##### 串联系统



图3.1串联模型

如图3.1所示，设各个子系统的可靠性分别为R1、R2、……、RN，系统的可靠性R＝R1×R2×…×RN。

设各个子系统的失效率为1，2，…，N，系统的失效率为：＝1+2+…+N。

系统的平均无故障时间为：MTBF＝1/。

##### 并联系统

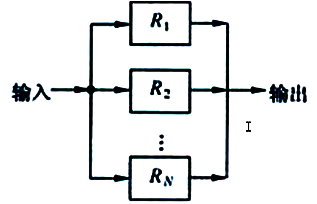


图3.2并联模型

如图3.2所示，设各个子系统的可靠性分别为R1、R2、……、RN，整个系统的可靠性：R＝1-(1-R1)×(1-R2)×….×(1-RN)。

设各个子系统的失效率为，整个系统的失效率为：。

系统的平均无故障时间为：MTBF＝1/＝

#### 计算机系统的性能评价

计算机系统性能常用的概念如下：

* 时钟周期：为了使计算机各部件同步工作，计算机中有一个脉冲源，通常是晶振，这个脉冲源产生主振脉冲，主振脉冲的时间间隔称为时钟周期。
* 时钟频率：指CPU晶振所产生脉冲的频率，一秒钟产生的主振次数。
* CPU周期：CPU从内存中读一个指令的最短时间称为CPU周期，一般由若干个时钟周期组成。
* 指令周期：CPU执行指令的时间称为指令周期，由于指令有不同的复杂程度，所以每种指令的指令周期可能不同，指令周期一般由若干个CPU周期组成。

评价计算机系统的性能好坏常用如下几个指标：

##### 时钟频率

##### 指令执行速度：每秒执行指令的数量，KIPS(千条指令每秒)，MIPS(百万条指令每秒)。

##### 等效指令速度法：统计各类指令在程序中所占比例，并进行折算。设某类指令i在程序中所占比例为Wi，该指令执行时间为Ti，即所需时钟数CPI(Cycles Per Instruction)，则等效指令的执行时间：

T=，等效指令执行的速度＝时钟频率/T。

如果采用(1)、(2)两种方式来评价计算机系统的性能，则具有片面性，通常采用第(3)中方式来对计算机性能进行评价。

###### 历年试题

●(2019年上)某系统由3个部件构成，每个部件的千小时可靠度都为R，该系统的千小时可靠度为(1-(1-R)²)R,则该系统的构成方式是 (4) 。

(4)A.3个部件串联 B.3个部件并联

C.前两个部件并联后与第三个部件串联 D.第一个部件与后两个部件并联构成的子系统串联

●(2018年下)某系统由下图所示的部件构成，每个部件的千小时可靠度都为R，该系统的千小时可靠度为(4) 。



(4)A.(3R+2R)/2 B.R/3+R/2 C.(1-(1-R)3)(1-(1 -R)2) D.(1-(1-R)3-(1-R)2)

●(2008年上)某数据处理软件包括 2 个完全相同的数据处理部件和 1 个数据[存储](http://www.educity.cn/incsearch/search.asp?key=%B4%E6%B4%A2)部件，且采用下图给出的容错方案。当数据处理部件的可靠性为 0.6 时，为使整个软件系统的可靠性不小于0.66，则数据存储部件的可靠性至少应为 (31) 。



(31)A.0.6 B.0.66 C.0.79 D.1.0

# 

# 差错控制技术

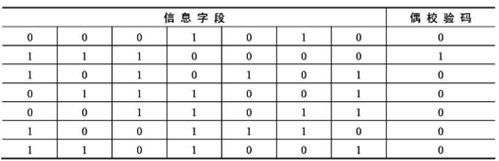
## 奇偶校验码

最简单应用最广泛的检错码。原理：发送方发送数据时，每组均增加一位校验位，使编码中1的个数保持为奇数(奇校验)或者为偶数(偶校验)；接收方收到数据时，检测每组数据中1的个数是否依然保持为奇数(奇校验)或者偶数(偶校验)，如果异常，则认为传输出错，奇偶校验方式简单，但只能发现奇数个错误，不能发现偶数个错误。

校验位的添加方法可以有如下三种：

水平奇偶校验码。对每一个数据的编码添加校验位。如表0.1所示。

表0.1水平奇偶校验码例表



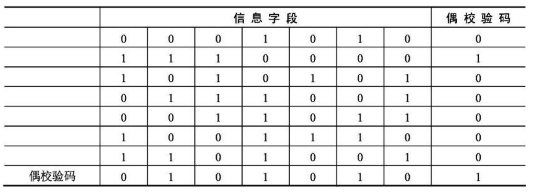
垂直奇偶校验码。校验位的添加不是针对单个数据，而是针对一组数据的相同位进行校验，添加校验位，形成该组数据的一个奇偶校验码，如表0.2所示。

表0.2垂直奇偶校验码例表



水平垂直奇偶校验码。先对一组数据进行垂直奇偶校验，所得结果(连同该添加的垂直校验位)再添加一位水平校验位。水平垂直奇偶校验码不仅能检测许多形式的错误，并且在给定的行或者列中产生孤立错误时，还可以对该错误进行纠正。如表0.3所示。

表0.3水平垂直奇偶校验码例表



虽然水平垂直奇偶校验码校验能力有所提高，但对于连续错误，该校验方式依然有漏洞存在。

●(2018年下)以下关于采用一位奇校验方法的叙述中，正确的是 (5) 。

(5) A.若所有奇数位出错，则可以检测出该错误但无法纠正错误

B.若所有偶数位出错，则可以检测出该错误并加以纠正

C.若有奇数个数据位出错，则可以检测出该错误但无法纠正错误

D.若有偶数个数据位出错，则可以检测出该错误并加以纠正

## 海明码

#### 海明距离

一个编码系统中任意两个合法编码(码字)之间不同的二进制数位的个数叫做这两个码字之间的海明距离，也叫码距。如ASCII中1的编码为0110001，2的编码为0110010，3的码字为0110011，则1和2之间的海明距离为2，1与3的海明距离为1，整个编码系统中任意两个码字的最小距离就是该编码系统的海明距离。要检测出d个错，则海明距离至少为d+1，要纠正d个错，海明距离至少2d+1。ASCII不具备检错和纠错距离。

#### 海明编码规则

对于给定的m位数据，DmDm-1….D1 加入K位校验位，PkPK-1….P1 形成n位码字，n=m+K，HNHN-1….H1,Pi放在整个编码的第2i-1位置，即Hj=Pi，j=2i-1，数据位则依顺序从低到高占据海明码剩下的位置，k的位数需满足m+K+1≤2k。

**例：**求1101001数据传送时，用海明码编码后的码字。解：

##### 确定K值：

因为1101001有7位，则m+K+1≤2k,所以K+8≤2k

* 当K=1时，不行，
* 当K=2时，不行，
* 当K=3时，不行，
* 当K=4时，可以。

##### 确定校验位位置，Hj=Pi，j=2i-1

，位于1、2、4、8的位置。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H11 | H10 | H9 | H8 | H7 | H6 | H5 | H4 | H3 | H2 | H1 |
| D7 | D6 | D5 | P4 | D4 | D3 | D2 | P3 | D1 | P2 | P1 |
| 1 | 1 | 0 | ? | 1 | 0 | 0 | ? | 1 | ? | ? |
|  |  |  | 0 |  |  |  | 1 |  | 0 | 1 |

##### 确定各校验位校验哪几个数据和校验位值

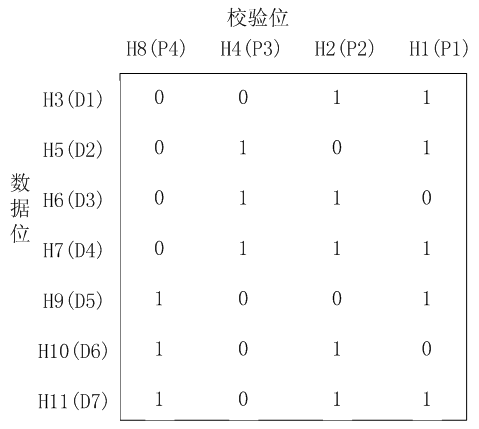


图.1海明编码例子

表0.4校验关系表

|  |  |  |  |
| --- | --- | --- | --- |
| 海明码中数位 | 下标 | 和H8(P4)H4(P3)H2(P2)H1(P1)的对应关系  (该位置为1时才负责监督左侧对应的位置，为0时没有监督责任) | 校验位组 |
| H1(P1) | 1=H1下标 | 0001(只有P1负责监督H1) | P1 |
| H2(P2) | 2=H2下标 | 0010(只有P2负责监督H1) | P2 |
| H3(D1) | 3=1+2=H1+H2下标 | 0011(P2、 P1同时负责监督H3) | P2，P1 |
| H4(P3) | 4 =H4下标 | 0100(略) | P3 |
| H5(D2) | 5=4+1=H4+H1下标 | 0101(略) | P3，P1 |
| H6(D3) | 6=4+2=H4+H2下标 | 0110(略) | P3，P2 |
| H7(D4) | 7=4+2+1= H4+H2+ H1下标 | 0111(P3、P2、 P1同时负责监督H7) | P3，P2，P1 |
| H8(P4) | 8=H8下标 | 1000(略) | P4 |
| H9(D5) | 9=8+1= H8+H1下标 | 1001(略) | P4，P1 |
| H10(D6) | 10=8+2= H8+H2下标 | 1010(P4、P2同时负责监督H10) | P4 ，P2 |
| H11(D7) | 11=8+2+1= H8+H2+ H1下标 | 1011(略) | P4， P2， P1 |

由图知，P1参与了D1、D2、D4、D5、D7的校验。校验公式为：P1 ⊕D1⊕D2⊕D4⊕D5⊕D7=0，也可以表示为：P1 =D1⊕D2⊕D4⊕D5⊕D7

按照偶校验方式进行计算P1值：P1 =1⊕0⊕1⊕0⊕1=1

同理P2、 P3、 P4的校验公式和值分别为：

* P2 =D1⊕D3⊕D4⊕D6⊕D7=1⊕0⊕1⊕1⊕1=0
* P3 =D2⊕D3⊕D4=0⊕0⊕1=1
* P4 =D5⊕D6⊕D7=0⊕1⊕1=0

则得海明编码后，1100100101

#### 海明码的纠错过程

检错需要计算这4个检验公式，若发现有一个公式的计算结果不为0，则说明该公式中某一位存在错误。

例：设上例中D4出错，传输中由1变为0，

则：P1⊕D1⊕D2⊕D4⊕D5⊕D7=1⊕1⊕0⊕0⊕0⊕1=1

P2⊕D1⊕D3⊕D4⊕D6⊕D7=0⊕1⊕0⊕0⊕1⊕1=1

P3⊕D2⊕D3⊕D4=1⊕0⊕0⊕0=1

P4⊕D5⊕D6⊕D7=0⊕0⊕1⊕1=0

则说明参与P4的各数据没有出错，出错的那一位一起参与了P1、P2、P3的校验，只有H7(D4)出错，即由P4P3P2P1=0111=7即可得出为H7出错。

●(2018年上)海明码是一种纠错码,其方法是为需要校验的数据位增加若干校验位,使得校验位的值决定于某些被校位的数据,当被校数据出错时,可根据校验位的值的变化找到出错位,从而纠正错误.对于32位的数据,至少需要增加(4)个校验位才能构成海明码.

以10位数据为例，其海明码表示为中，其中(1≤i≤9)表示数据位，(1≤j≤4)表示校验位，数据位由、和进行校验(从右至左的位序为14,即等于8+4+2,因此用第8位的、第4位的和第2位的校验),数据位由(5)进行校验。

(4)A.3 B.4 C.5 D.6

(5)A. B. C. D.

●(2017年下)以下关于海明码的叙述中，正确的是 (5) 。

(5)A.海明码利用奇偶性进行检错和纠错 B.海明码的码距为1

C.海明码可以检错但不能纠错 D.海明码中数据位的长度和校验位的长度必须相同

●(2017年上)己知数据信息为 16 位，最少应附加(5)位校验位，才能实现海明码纠错。

(5)A.3 B.4 C.5 D.6

●海明码是一种纠错编码，一对有效码字之间的海明距离是 (14) 。如果信息为6位，要求纠正1位错，按照海明编码规则，需要增加的校验位是 (15)位。

(14)A.两个码字的比特数之和 B.两个码字的比特数之差

C.两个码字之间相同的比特数 D.两个码字之间不同的比特数

(15)A.3 B.4 C.5 D.6

●海明码(Hamming Code)是一种 (17) 。

(17)A.纠错码 B.检错码 C.语音编码 D.压缩编码

●设数据码字为10010011,采用海明码进行校验，则必须加入(20) 比特冗余位才能纠正一位错。

(20) A.2 B.3 C.4 D.5

## CRC码(循环冗余校验码)

在实践中，被采用的是下述的CRC循环冗余校验码。

CRC码计算过程：数据传输时，通信双方事先约定一个生成多项式G(x)，生成多项式的最高位和最低位都是1，发送端用要传送的m位二进制码除以该生成多项式G(x)的各项系数，取r位余数，即校验用的监督码(CRC)码，加在原始数据位后，形成一个新的序列(m+r)位发出去，接收把收到的序列用来除以同一个G(X)的各项系数，若余数为0则收到的编码正确，否则错误。(除法用模二除法)

例：假设使用生成多项式G(x)=**X3+X+1**,对报文**1001**进行CRC编码，求编码后的报文是什么？

##### 求出生成多项式G(x)= **X3+X+1**系数作为除数：**1011**；

##### 在要传送的报文末尾加上多个0，位数等于多项式的最高次数3个；得到新报文1001000

##### 用新报文除以多项式系数，得到余数，余数位数等于多项式的最高次数；

##### 将得到的余数加到原数据报文的最后，即最终以码字**1001110**进行传输；

##### 对方收到报文，同样除以多项式后，若余数为0，则传输未出错，若余数不为0，则认为传输出错。

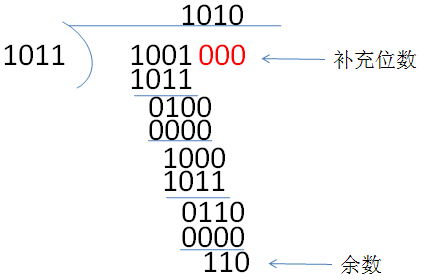
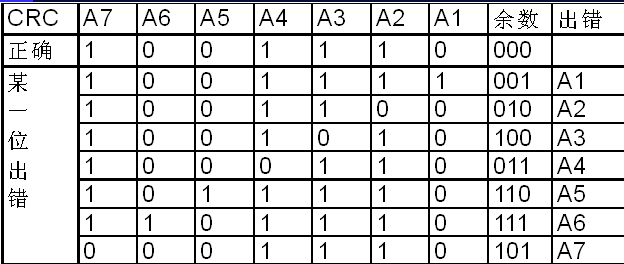


图0.2CRC编码校验和计算过程

#### CRC码的差错表



#### 生成多项式的确定

G(x)是一个约定的除数，用来产生校验码。从检错和纠错的要求出发，它并不是随意选择的，它应满足下列要求：

* 任何一位发生错误都应使余数不为0 ；
* 不同位发生错误应使余数不同；
* 余数继续模2 除，应使余数循环。

CRC特点：电路设计简单，可检测出所有双数位错，奇数位错和突发位错≤r的突发错误。

以下几种已定义为国际标准的CRC生成多项式：

* CRC-12 G(x)=x12+ x11+ x3+ x2+x1+1 通常用来传送6-bit字符串。
* CRC-16 G(x)=x16+ x15+ x2+1 通常传送8-bit字符磁盘读写也使用
* CRC-CCITT G(x)=x16+ x12+ x5+1 通常传送8-bit字符通讯协议X.25的FCS(帧检错序列)使用
* CRC-32 G(x)=x32+ x26+ x23+ x22+ x16+ x12+ x11+ x10+ x8+ x7+ x5+ x4+ x2+ x+1 常以太网中使用，

●(2019年上)在 (5) 校验方法中,采用模2运算来构造校验位。(软/网)

(5)A.水平奇偶 B.垂直奇偶 C.海明码 D.循环冗余

●(2016年下)己知数据信息为16位，最少应附加\_(4)\_位校验位，以实现海明码纠错。

(4)A.3 B.4 C.5 D.6

●(2012年下)循环冗余校验码(CRC)利用生成多项式进行编码。设数据位为k位，校验位为r位，则CRC码的格式为 (2) 。

(2) A.k个数据位之后跟r个校验位 B.r个校验位之后跟k个数据位

C.r个校验位随机加入k个数据位中 D.r个校验位等间隔地加入k个数据位中

●(2014年上)海明码利用奇偶性检错和纠错，通过在n个数据位之间插入k个校验位，扩大数据编码的码距。若n=48，则k应为(3)。

(3)A.4 B.5 C.6 D.7

●(2009年下)以下关于校验码的叙述中，正确的是 (5) 。

(5)A.海明码利用多组数位的奇偶性来检错和纠错 B.海明码的码距必须大于等于1

C.循环冗余校验码具有很强的检错和纠错能力 D.循环冗余校验码的码距必定为1