# 计算机组成与体系结构

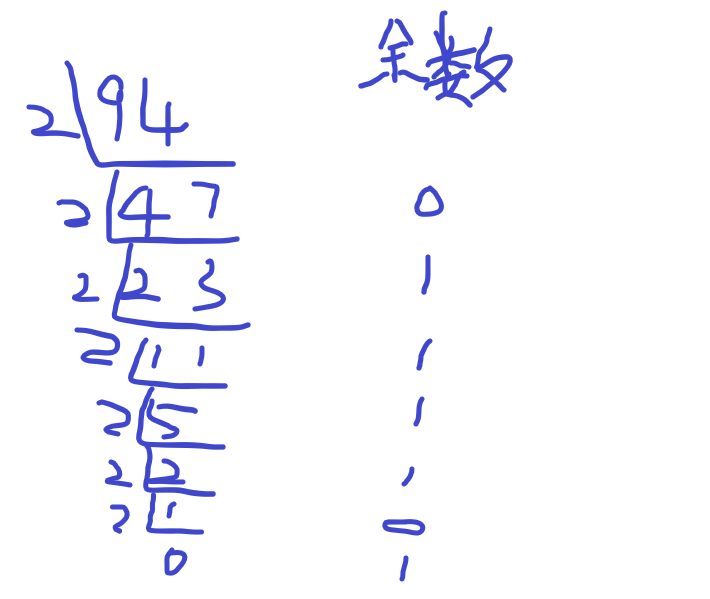
## 数据的表示

R进制转十进制使用按权展开法，其具体操作方式为：将R进制数的每一位数值用Rk形式标识，即幂的底数是R，指数为k，k与该位和小数点之间的距离有关。当该位位于小数点左边，k值是该位和小数点之间数码的个数，而当该位位于小数点右边，k值是负值，其绝对值是该位和小数点之间数码的个数加1。

例如：二进制10100.01 = 1\*24 + 1\*22 + 1\*2-2

例如：七进制604.01 = 6\*72 + 4\*70 + 1\*7-2

十进制转R进制使用短除法。例如：将94转换为二进制数



结果为1011110

二进制转八进制与十六进制数

10001110 转为8进制，不够三位可在最左边补0

2 1 6

10001110 转为16进制，不够四位可以在最左边补0

8 E

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 最高位为符号位 | 数值1 | 数值-1 | 1-1（前面两列运算） |  |
| 原码（最高位为符号位，0为正数，1位负数） | 0000 0001 | 1000 0001 | 1000 0010 | -2，结果不对 |
| 反码(正数不变，负数符号位不变，按位取反) | 0000 0001 | 1111 1110 | 1111 1111 | 在转换为真值时，若符号位为0，数位不变，若符号位为1，应将结果求反才是真值 |
| 补码(正数不变，负数为反码+1) | 0000 0001 | 1111 1111 | 0000 0000 | 0，结果正确 |
| 移码（补码的基础上首位取反，其他位不变 ） | 1000 0001 | 0111 1111 | 1000 0000 | 移码首位为1表示正数，结果为0，正确 |

数值表示范围

|  |  |  |
| --- | --- | --- |
| n位 | 正数 | 8位为例 |
| 原码 | -(2n-1-1)~(2n-1 -1) | -127~127 |
| 反码 | -(2n-1-1)~2n-1-1 |  |
| 补码(+0和-0的的补码都为00000000，少占了一个数) | -2n-1~2n-1-1 | -128~127 |

浮点数运算

浮点数的表示：

N = M\*Re

其中M成为尾数，e是指数，R为基数

结果格式化

尾数计算

对阶

如：1000 = 1.0\*103  119 = 1.19\*102 1.0 + 1.19不行

对阶：将1.19\*102转换为0.119\*103 一般往高阶转换

结果格式化：假设结果为0.119\*103则应该格式化为1.19\*102，

假设结果为11.9\*102则应该格式化为1.19\*103

十进制小数转二进制小数：乘以2取整，一直到小数部分为零为止，如果永远不能为0，就和十进制四舍五入一样，按照要求保留多少位小数时，就根据后面一位是0还是1，取舍，如果是0舍掉，如果是1，向入一位。就是0舍1入，读数要从前面的整数读到后面的整数，顺序排列（乘2取整法）

例如：0.125 => 0.001

0.125 \*2 = 0.25 —— 0

0.25\*2 = 0.50 —— 0

0.50\*2 = 1.00 —— 1

## 计算机结构

主机

主机

CPU

运算器

控制器

主存储器

1. 算术逻辑单元ALU
2. 累加寄存器AC
3. 数据缓冲寄存器DR
4. 状态条件寄存器PSW
5. 程序计数器PC
6. 指令寄存器IR
7. 指令译码器
8. 时序部件

## Flynn分类法

### 计算机体系结构分类——Flynn

|  |  |  |  |
| --- | --- | --- | --- |
| 体系结构类型 | 结构 | 关键特性 | 代表 |
| 单指令流单数据流SISD | 控制部分：一个  处理器：一个  主存模块：一个 |  | 单处理器系统  386、486 |
| 单指令流多数据流SIMD | 控制部分：一个  处理器：多个  主存模块：多个 | 各处理器以异步的形式执行同一条指令 | 并行处理机  阵列处理机  超级向量处理机 |
| 多指令流单数据流MISD | 控制部分：多个  处理器：一个  主存模块：多个 | 被证明不可能，至少是不实际 | 目前没有，有文献称流水线计算机为此类 |
| 多指令流多数据流MIMD | 控制部分：多个  处理器：多个  主存模块：多个 | 能够实现作业、任务、指令等各级全面并行 | 多处理机系统  多计算机 |

## CISC与RISC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令系统类型 | 指令 | 寻址方式 | 实现方式 | 其他 |
| CISC（复杂Complex） | 数量多，使用频率差别大，可变长格式 | 支持多种 | 微程序控制技术（微码） | 研制周期长 |
| RISC（精简Reduced） | 数量少，使用频率接近，定长格式，大部分为单周期指令，操作寄存器，只有load/store操作内存 | 支持方式少 | 增加了通用寄存器；硬布线逻辑控制为主，适合采用流水线 | 优化编译，有效支持高级语言 |

## 流水线技术

### 概念

流水线是指在程序执行时多条指令重叠进行操作的一种准并行处理实现技术。各种部件同时处理是针对不同指令而言的，它们可同时为多条指令的不同部分进行工作，以提高各部件的利用率和指令的平均执行速度

取指

分析

执行

### 流水线计算

流水线周期：为执行时间最长的一段

流水线计算公式为：

1条指令执行时间 + （指令条数-1）\* 流水线周期

1. 理论公式：（t1 + t2 + .. + tk）+ (n-1)\*△t
2. 实践公式：(k + n - 1)\* △t

例：若指令流水线把一条指令分为取指、分析和执行三部分，且三部分的时间分别是取指2ns，分析2ns，执行1ns，那么流水线周期是多少？100条指令全部执行完毕需要的时间是多少？

周期为：2ns

100条指令全部执行完毕需要：理论公式计算为：（2+2+1）+99\*2=203

实践公式计算：应该以每一段一个周期来计算：(3+100-1)\*2=204。

可以简单这样理解，如果第三段以1ns计算，则下一条指令可能还未到执行阶段。

### 流水线吞吐率计算

流水线的吞吐率（Through Put rate，TP）是指在单位时间内流水线所完成的任务数量或输出的结果数量。计算流水线吞吐率的最基本公式如下：

TP = 指令条数/流水线执行时间

流水线最大吞吐率：

TPmax = limn->∞(n/((k+n-1) △t)) = 1/△t

以前一个例子为例：

理论结果为100条的时间为203ns，则TP = 100/203，TPmax为极限情况，忽略流水线的建立时间（输出第一个结果的时间），所以TPmax = 1/2

### 流水线的加速比

完成同样一批任务，不使用流水线所使用的时间与使用流水线所使用的时间之比。基本公式如下：

S=不使用流水线执行时间/使用流水线执行时间

以前面100条指令的题目为例：

不使用流水线的时间为 （2+2+1）\*100

使用流水线的时间为：203

S = 500/203

### 流水线的效率

流水线的效率是指流水线的设备利用率。在时空图上，流水线的效率定义为n个任务占用的时空区与k个流水段总的时空区之比

入->s1->s2->s3->s4->出

## 存储系统

### 层次化存储结结构

快

速

度

速度快

慢快

CPU

Cache

内存（主存）

外存（辅存）

寄存器

按内容存取

硬盘、光盘、U盘等

容量

小

大

按内容存取：存信息的时候考虑信息的内容，不同内容存到不同区域，读取信息的时候根据内容就可以知道存储的区域，这种存储器又叫关联（相联）存储器，比普通按地址存储的存储器速度快。

按内容存取2：关联存取器（又译作相联存储器）也称为按内容访问存储器或简称为TLB；它是一种不根据地址而是根据存取内容来进行存取的存储器，可以实现快速地查找块表。

原理:写入信息的时候按顺序写入，不需要地址。读出时，要求中央处理单元给出一个相关联字，用它和存储器中所有单元中的一部分信息进行比较，若它们相等，则将此单元中余下的信息读出。这是实现存储器并行操作的一种有效途径，特别适合于信息的检索和更新。

### Cache-概念

Cache的功能：提高CPU数据输入输出的速率，突破冯诺依曼瓶颈，即CPU与存储系统之间数据传送带宽限制。

在计算机的存储体系中，Cache是访问速度最快的层次（有寄存器则为寄存器，否则为cache）

使用Cache改善系统性能的依据是程序的局部性原理。

如果以h代表对cache的访问命中率，t1表示cache的周期时间，t2表示主存储器周期时间，以读操作为例，使用“cache+主存储器”的系统的平均周期为t3，则：

t3 = h\*t1+(1-h)\*t2

其中，（1-h）又称为失效率（未命中率）

### 局部性原理

概念：局部性原理是指CPU访问存储器时，无论是存取指令还是存取数据，所访问的存储单元都趋于聚集在一个较小的连续区域中。

时间局部性（Temporal Locality）：如果一个信息项正在被访问，那么在近期它很可能还会被再次访问，程序循环，堆栈等是产生时间局部性的原因

空间局部性（Spatial Locality）：在最近的将来将用到的信息很可能与现在正在使用的信息在空间地址上是临近的。

顺序局部性（Order Locality）：在典型程序中，除转移类指令外，大部分指令是顺序执行的。顺序执行和非顺序执行的比例大致是5：1.此外，对大型数组访问也是顺序的。指令的顺序执行、数组的连续存放等是产生顺序局部性的原因。

工作集理论：根据程序局部性理论Denning提出了工作集理论。工作集是进程运算时被频繁访问的页面集合。

### 进程工作集

一个进程当前使用的页面的集合叫做它的工作集（working set）。如果整个工作集都在内存中，在进入下一个运行阶段之前进程的运行不会引起很多页面故障。如果内存太小无法容纳整个工作集，进程运行将引起大量的页面故障并且速度十分缓慢。一个每隔几条指令就发生一次页面故障的程序被称为是在颠簸。

现代计算机系统中内存的访问速度远远高于外存的访问速度。如果系统中不产生缺页中断，则访问数据的时间约等于内存访问时间。但是如果发生缺页中断，则需要从外存中将该页面调入，因此会大大降低系统性能。通过对缺页率的长期研究，Denning提出了工作集理论。由于程序在运行时对页面的访问是不均匀的（即局部性），如果能够预知程序在某段时间内要访问哪些页面，并将它们提前调入内存，这将降低缺页率，提高CPU利用率。

进程工作集：因为频繁的调页操作引起的磁盘I/O会大大降低程序的运行效率，因此对每一个进程，虚拟内存管理器都会将一定量的内存页驻留在物理内存中。并跟踪其执行的性能指标，动态调整这个数量。Win32中驻留在物理内存中的内存页称为进程的“工作集（working set）”，进程的工作集可以通过任务管理器查看，其中“内存使用”列即为工作集的大小。

工作集是会动态变化的，进程初始时只有很少的代码页和数据页被调入内存。当执行到未被调入内存的代码或者访问到尚未被调入内存的数据时，这些代码页或者数据页会被调入物理内存，工作集页随之增长。但工作集不能无限增长，系统为每个进程都定义了一个默认的最小工作集（根据系统物理内存大小，此值可能为20~50MB）和最大工作集（根据系统物理内存大小，此值可能为45~345MB）。当工作集达到最大工作集，即进程需要再次调入新页到物理内存中时，虚拟内存管理器会将其原来的工作集中的某些页先置换出内存，然后将需要调入的新页调入内存。

因为工作集的页驻留在物理内存中，因此对这些页的访问不涉及磁盘I/O，相对而言非常快；反之，如果执行的代码或者访问的数据不再工作集中，则会引发额外的磁盘I/O，从而降低程序的运行效率。一个极端的情况就是所谓的颠簸或抖动（thrashing），即程序的大部分执行时间都花在了调页操作上，而不是代码执行上。

如前所述，虚拟内存管理器在调页时，不仅仅只是调入需要的页，同时还将其附近的页也一起调入内存中。综合这些知识，对开发人员来说，如果想提高程序的运行效率，应该考虑一下两个因素：

1. 对代码来说，尽量编写紧凑代码，这样最理想的情况就是工作集从不会达到最大阈值。在每次调入新页时，也就不需要置换已经载入内存的页。因为根据Locality特性，以前执行的代码和访问的数据在后面有很大可能会被再次执行或访问。这样程序程序执行时，发生的缺页错误数就会大大降低，即减少了磁盘I/O。即使不能达到这种理想情形，紧凑代码页往往意味着接下来执行的代码更大可能就在相同页或相邻页。根据时间Locality特性，程序80%的时间花在20%的代码上。如果能将这20%的代码尽量紧凑且排在一起，无疑会大大提高程序的整体运行性能。
2. 对数据来说，尽量将那些会一起访问的数据（比如链表）放在一起。这样当访问这些数据时，因为他们在同一页或相邻页，只需一次调页操作即可完成；反之，如果这些数据分散在多个页（更糟的情况是这些页还不相邻），那么每次对这些数据的整体访问都会引发大量的缺页错误，从而降低系统性能。利用Win32提供的预留和提交两步机制，可以为这些会一同访问的数据预留出一大块空间。此时并没有分配实际存储空间，然后在后续执行过程中生成这些数据时随需为他们提交内存。这样既不浪费真正的物理内存（包括调页文件的磁盘空间和物理内存空间），又能利用Locality特性。另外内存池机制也是基于类似的考虑。此外，工作集和驻留集不同，驻留集指当前在主存中的页面集，而工作集指不久后所需要的页面集。

Resident set, In a virtual memory system, a process’ resident set is that part of a process’ address space which is currently in main memory. If this does not include all of the process’ working set, the system may thrash.

Working set, the working set of a program or system is that memory or set of address which is will use in the near future. This term is generally used when discussing miss rates at some storage level; the time scale of “near future” depends upon the cost of a miss. The working set should fit in the storage level; otherwise the system may thrash.

例：

int i,j,s=0;

for(i=1;i<1000;i++){

for(j=1;j<1000;j++){

s+=j;

}

}

printf(“结果是:%d”,s);

### 主存-分类

随机存取存储器

只读存储器

DRAM（Dynamic RAM，动态RAM）-SDRAM

SRAM（Static RAM，静态）

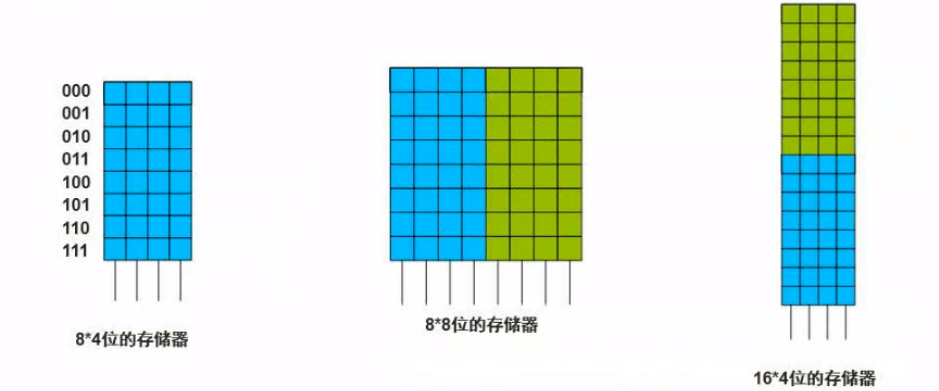
MROM（Mask ROM，掩模式ROM）

PROM（Programmable ROM，一次可编程ROM）

EPROM（Erasable PROM，可插除的PROM）

闪速存储器（flash memory，闪存）

### 主存-编址



内存地址从AC000H到C7FFFH，共有（112）K个地址单元，如果该内存地址按字（16bit）编址，由28片存储器芯片构成。已知构成此内存的芯片每片有16K个存储单元，则该芯片每个存储单元存储（4）位。

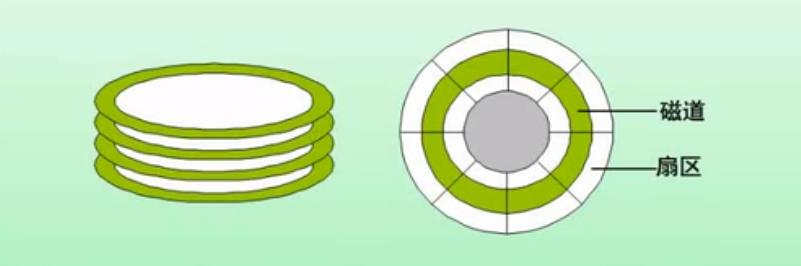
答：C7FFFH – AC000H + 1 （类似于0~9有10个数，所以要加1）= C8000H – AC000H = 1C000H 个地址空间，转换为多少K：1C000/210 = 112

1C000转换为10进制 = （1\*164 + 12\*163）/210 = + ((24)4 + 12\*(24)3)/210 =( 216 + 12\*212 )/210

=26 + 12\*22=64 + 48 =112

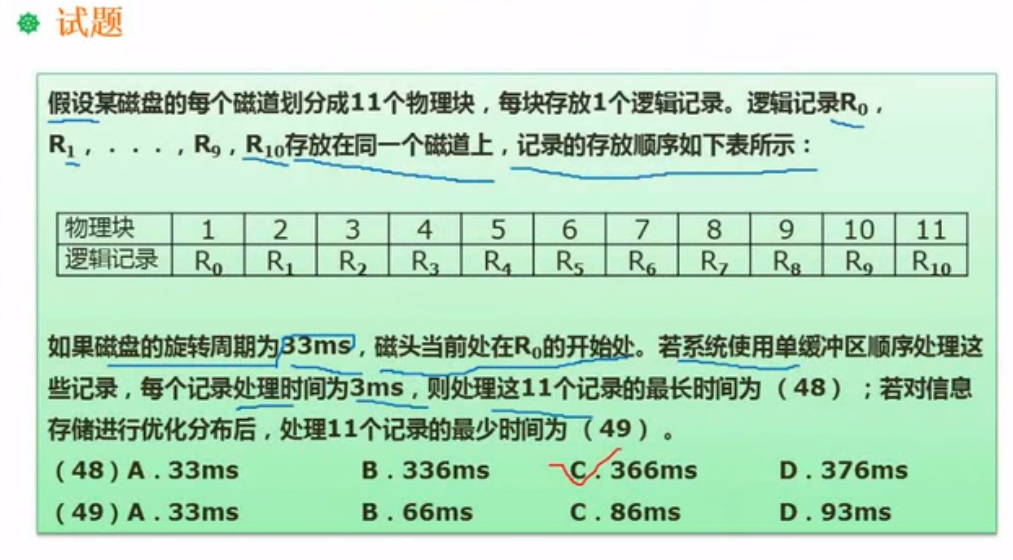
总容量为（112K \* 16）/(28 \* 16K\*x) = 1 所以x = 4

### 磁盘结构与参数

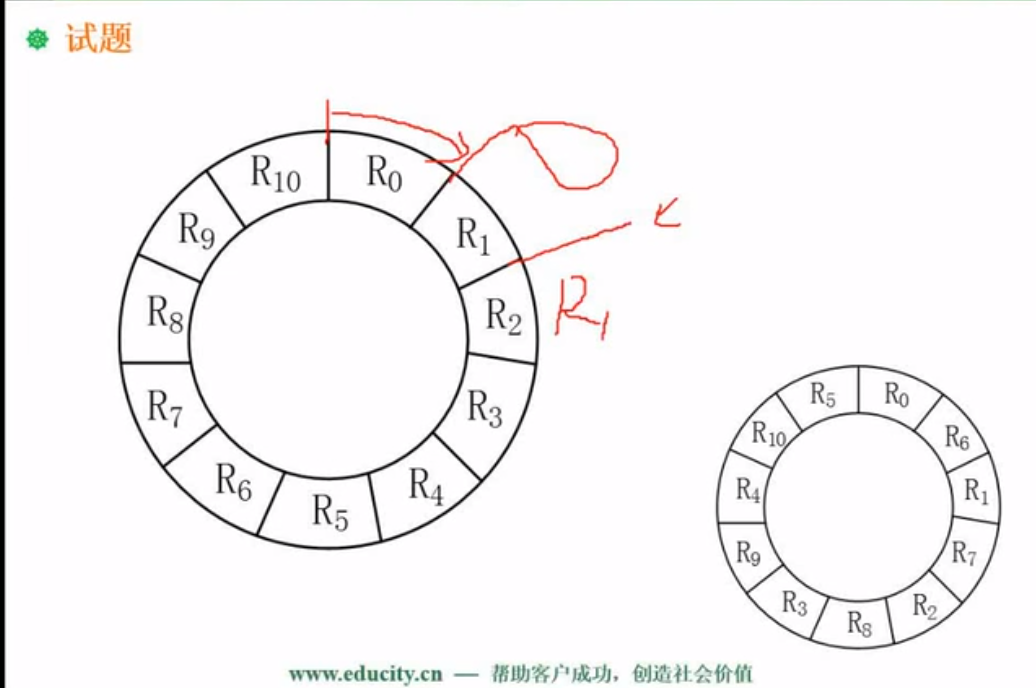


存取时间 = 寻道时间+等待时间（平均定位时间+转动延迟）

注意：寻道时间是指磁头移动到磁道所需的时间；等待时间为等待读写的扇区转到磁头下方所用的时间。



(48)为366ms，(49)为66ms



## 总线系统

 根据总线所处的位置不同，总线通常被分成三种类型：

内部总线

系统总线：

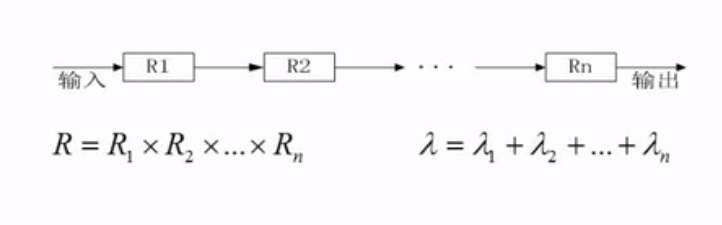
数据总线，地址总线，控制总线

外部总线

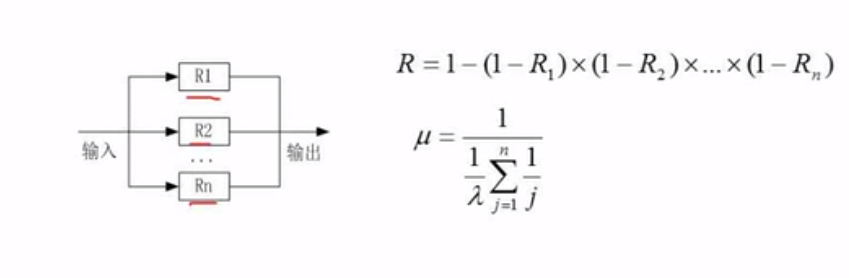
## 可靠性

### 系统可靠性分析-串联系统与并联系统

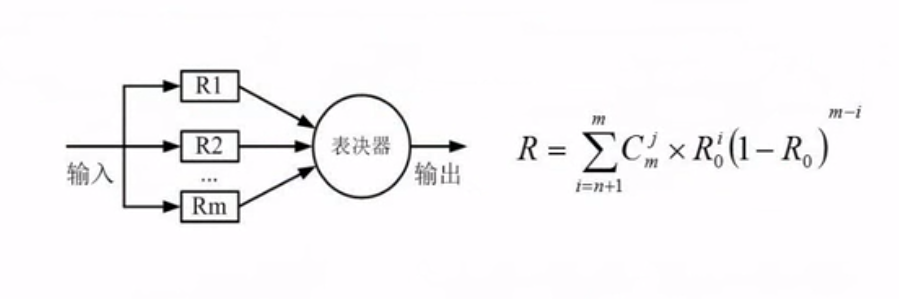
串联

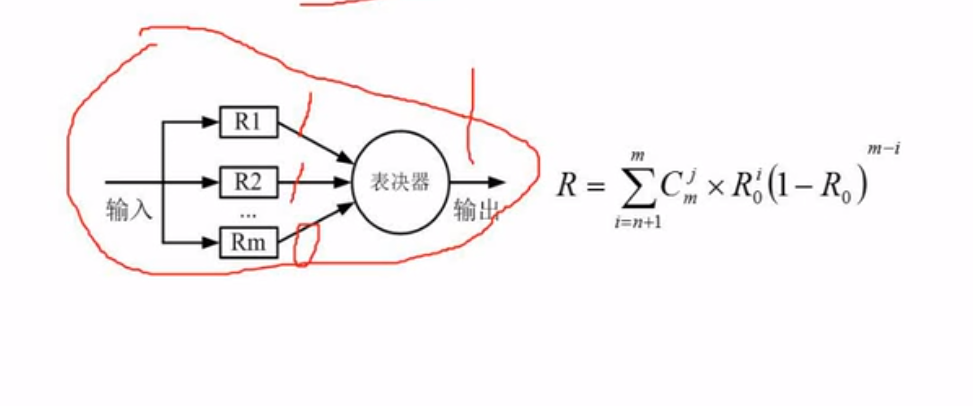


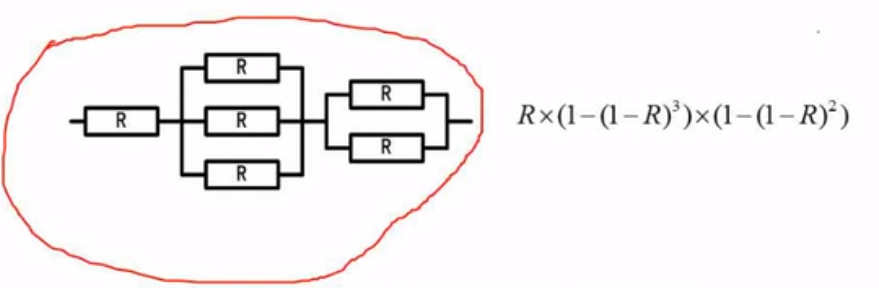
并联



### 系统可靠性分析-模冗余系统与混合系统







## 校验码

### 差错控制-CRC与海明校验码

### 什么是检错和纠错

检错是找出错误，纠错是找到并纠正错误。

### 什么是码距？

一个编码系统的码距是整个编码系统中任意（所有）两个码字的最小距离（变化多少位得到另一个码字）。

例：

若用1位长度的二进制编码。若A=1，B=0，这样A,B之间的最小码距为1

若用2位长度的二进制编码。若A=11，B=00，则A，B之间的最小码距为2

若用3位长度的二进制编码。可选用111，000作为合法编码，A,B之间的最小码距为3

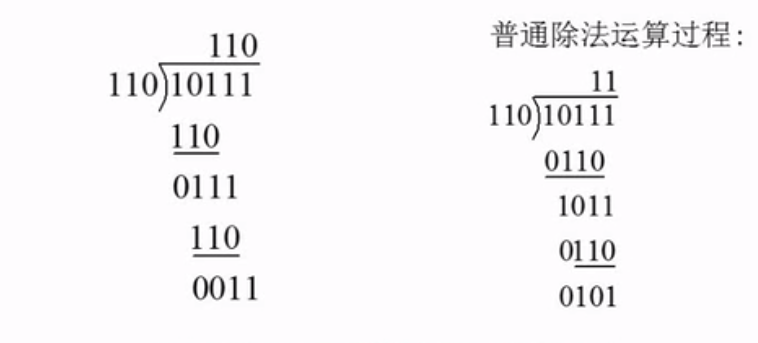
### 码距与检错、纠错有何关系

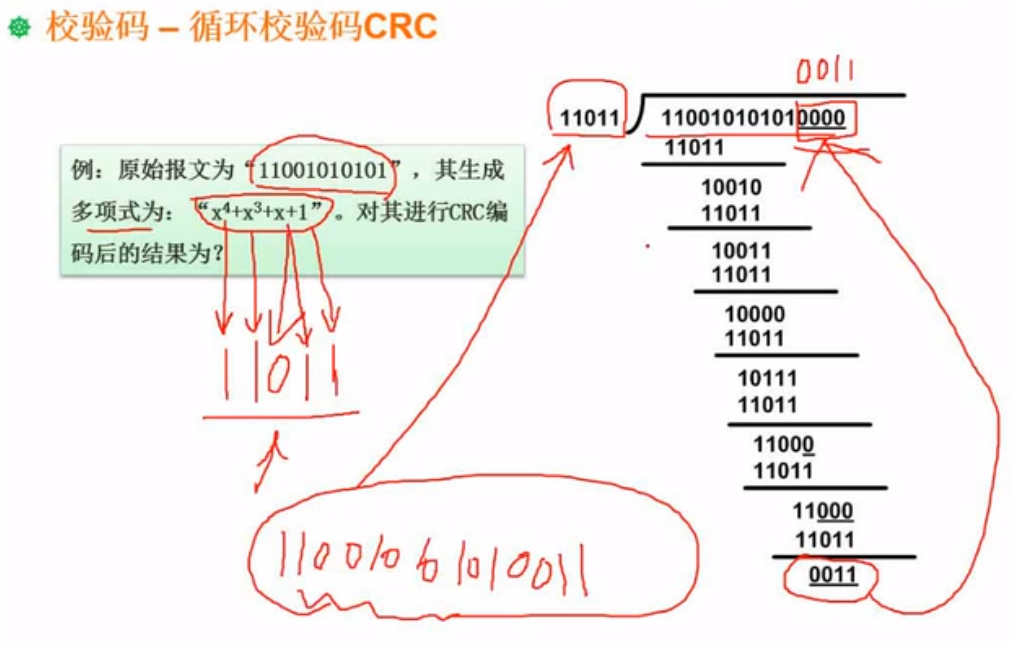
1. 在一个码组内为了检测e个误码，要求最小码距d应该满足：d>=e+1
2. 在一个码组内为了纠正t个误码，要求最小码距d应该满足：d>=2t+1

### 校验码-循环校验码CRC

什么是模2除法，他和普通的除法有什么区别？

模2除法是指在做除法运算的过程中不计其进位的除法。

例如：10111对110进行模2除法为



### 校验码-海明校验码

例：求信息 1011的海明码？

（1）2r>=4+r+1，确定校验码为3位；23>=4+3+1。分别放在20=1，21=2，22=4位（4为信息位的个数，r为校验位的个数2r>=x+r+1）

（2）列出校验公式

