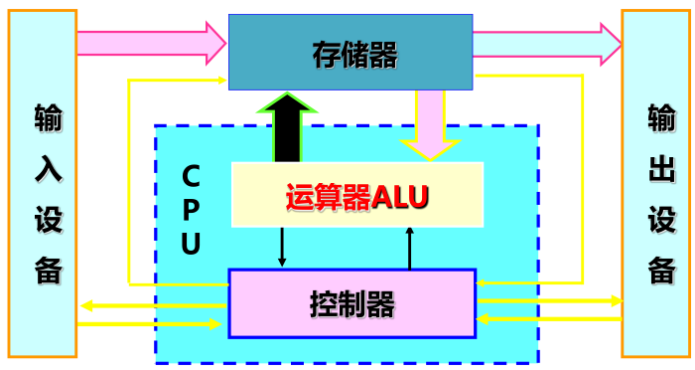
《计算机科学导论》知识点梳理

Powered by Arisu Shiro

第一章 计算机科学概论

1. 硬件

输入设备：如键盘、鼠标；

存储设备：如内存，硬盘；

运算控制设备：如中央处理器CPU；

输出设备：如显示器。

1. 输入数据在硬件上的逻辑流动过程中，涉及到的数据传输都是通过总线完成的。
2. 硬件无法自我完成和实现用户的需求

CPU是计算机的大脑，计算机语言控制CPU按步骤执行任务

指示CPU进行操作的语言称为程序语言

人们预定的安排是通过一个指令序列来表达的，这个指令序列称为程序（或软件）

在程序执行的这个黑匣子中，软件描述了用户的需求，硬件则实现了用户的需求。

1. 软件并不能独立的控制硬件进行工作。

计算机结构中，还需要一个层次来衔接软件与硬件、控制硬件工作、为软件提供服务等。这个特殊的层次就是操作系统（OS）。操作系统其实也是一组程序。

1. 计算机系统分为三个层次：硬件层、操作系统层、软件层。

（1）硬件层包括计算机的各个部件，控制器、运算器、存储器、输入设备和输出设备。这个存储程序计算机架构（冯·诺依曼结构）早在1964年，由美籍匈牙利科学家冯·诺依曼提出。目前，绝大多数计算机采用冯·诺依曼结构，冯·诺依曼也因此被人们称为“计算机之父”。

【重要】冯·诺依曼结构特点：

* 计算机的数制采用二进制，数据和指令均采用1和0组成的二进制代码表示。
* 计算机指令和数据存储在同一记忆装置中，即存储器。
* 计算机按照预先编制的程序顺序执行，即程序控制。

（2）软件层包括由汇编以及高级语言（C/C++，Java，Python等）等开发出的应用程序。对通用型计算机而言，功能的实现需要软硬件的无缝配合。要使硬件CPU发挥计算功能，需要控制指令来完成。

（3）操作系统层是连接硬件和软件的中间桥梁，操作系统的种类繁多，根据不同的应用要求，有不同的操作系统。生活中最常见的操作系统有微软的Windows系列产品、Ubuntu、Fedora等Linux系统，苹果Mac OS系列产品，以及智能手机中所使用的Android、IOS系统等等。

1. 操作系统的主要职能

* 管理文件系统，管理各种硬件资源，例如U盘、网络、键盘等
* 管理程序共享的资源，例如CPU、主存等
* 管理和调度多个程序的执行
* 提供程序和硬件的衔接，提供各种系统的服务和接口
* 设法维护系统的安全，尽量防止病毒（恶意软件）有意或无意的侵入

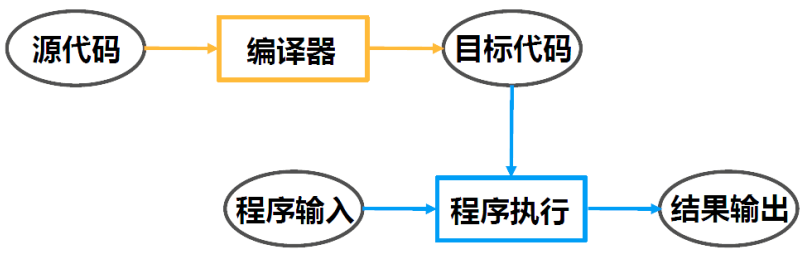
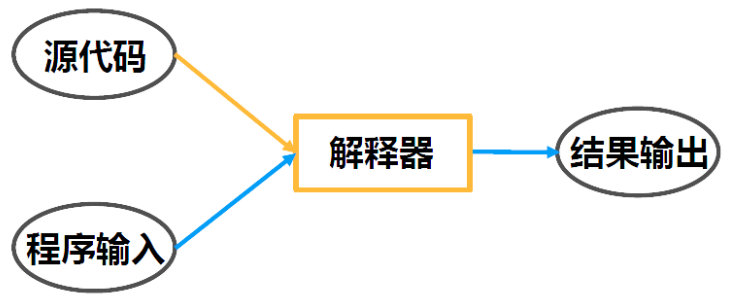
1. 软件层的实现主要由程序设计语言完成。程序设计语言是计算机能够理解和识别用户操作意图的一种交互体系，它按照特定规则组织计算机指令，使计算机能够自动进行各种运算处理。按照程序设计语言规则组织起来的一组计算机指令称为计算机程序。程序设计语言包括：机器语言、汇编语言和高级语言。
2. 机器语言：一种二进制语言，它直接使用二进制代码表达指令，是计算机硬件可以直接识别和执行的程序设计语言。
3. 汇编语言：使用助记符与机器语言中的指令进行一一对应，在计算机发展早期帮助程序员提高编程效率。

机器语言和汇编语言都直接操作计算机硬件并基于此设计，所以它们统称为低级语言。

1. 高级语言：高级语言区别于低级语言在于，高级语言是接近自然语言的一种计算机程序设计语言，更容易地描述计算问题并利用计算机解决计算问题。第一个广泛应用的高级语言是诞生于1972年的C语言。
2. 计算机执行源程序的两种方式：编译和解释

* 源代码：采用某种编程语言编写的计算机程序，人类可读。例如：result = 2 + 3
* 目标代码：计算机可直接执行，人类不可读 (专家除外)。例如：11010010 00111011

1. 编译：将源代码一次性转换成目标代码的过程。执行编译过程的程序叫作编译器。
2. 解释：将源代码逐条转换成目标代码同时逐条运行的过程。执行解释过程的程序叫作解释器。

1. 计算机分类
2. 通用型计算机：常用的台式计算机、笔记本电脑、平板电脑等，或服务器、超级电脑等。
3. 专用型计算机：为特定应用量身打造，内部的程序一般不能被改动。常被称为“嵌入式系统”。
4. 电子计算机从诞生起，经过了电子管、晶体管、集成电路（IC）和超大规模集成电路（VLSI）四个阶段的发展。（摩尔定律）集成电路上可以容纳的晶体管数目在大约每经过18个月到24个月便会增加一倍。
5. 第一代计算机：体积较大；运算速度较低；存储容量不大；而且价格昂贵；使用也不方便。
6. 第二代计算机：全部采用晶体管作为电子器件，其运算速度比第一代计算机的提高了近百倍，体积为原来的几十分之一。在软件方面，开始使用计算机算法语言。
7. 第三代计算机：主要特征是以中、小规模集成电路为电子器件。重大突破是出现了操作系统。
8. 第四代计算机：采用大规模集成电路（LSI）和超大规模集成电路（VLSI）为主要电子器件。另一个重要分支是以大规模、超大规模集成电路为基础发展起来的微处理器和微型计算机。
9. 数据是指所有能输入到计算机并被计算机程序处理的，具有一定意义的数字、字母、符号和模拟量等的通称。计算机利用模数转换器（ADC，Analog to Digital Converter），将模拟信号（即真实世界的连续信号）转换为数字信号（即用数值表示的离散信号），即0和1进行传输。

第二章 数字系统和信息编码

1. 十进制（Decimal)、二进制（Binary）、八进制（Octonary）与十六进制（Hexadecimal）

2. 基数（Base）与位权（Weight）

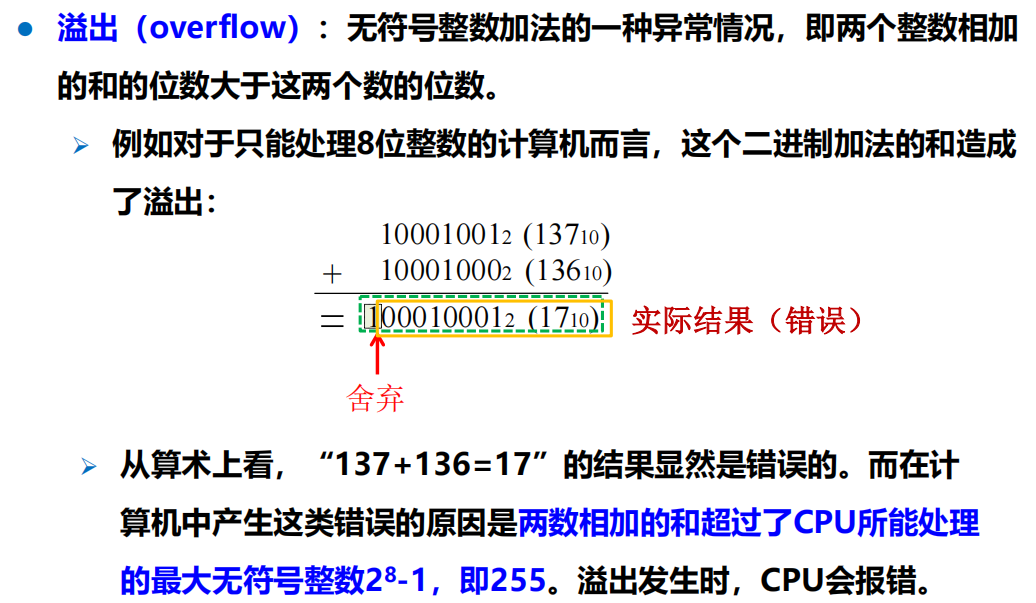
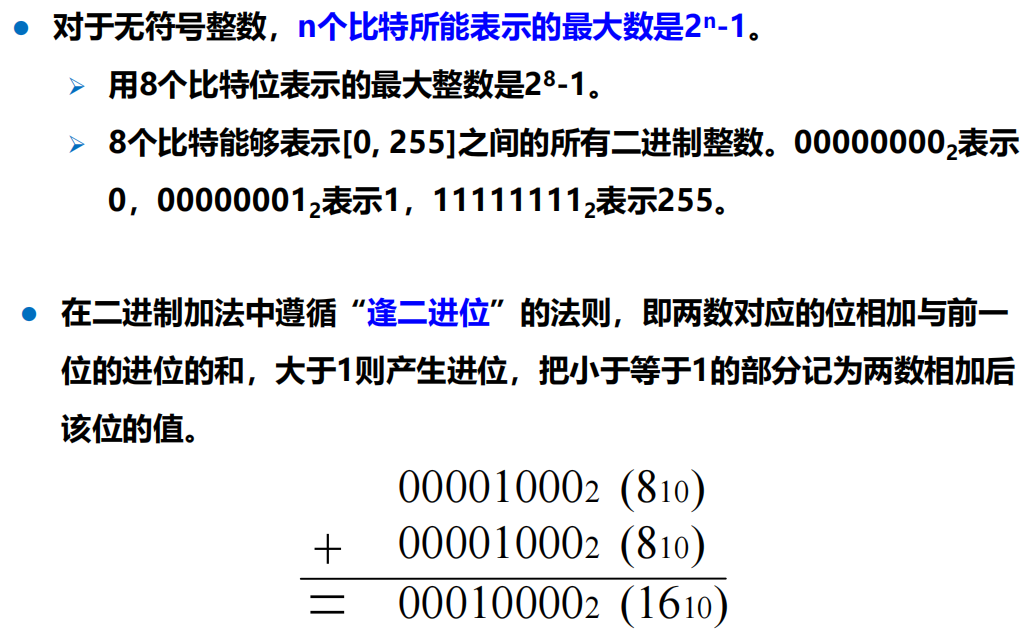
3. 不同进制间的转换

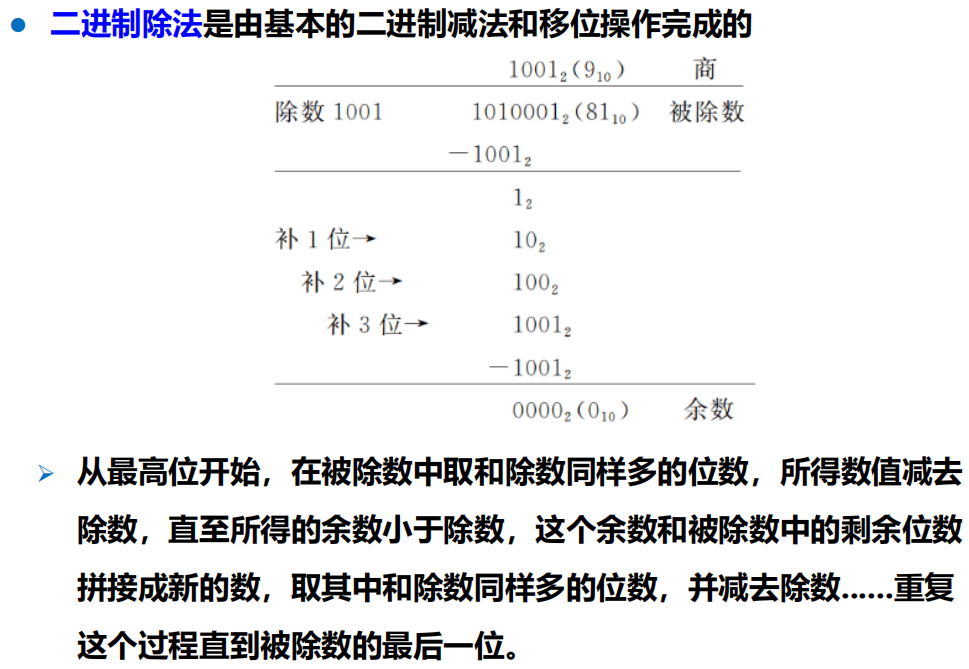
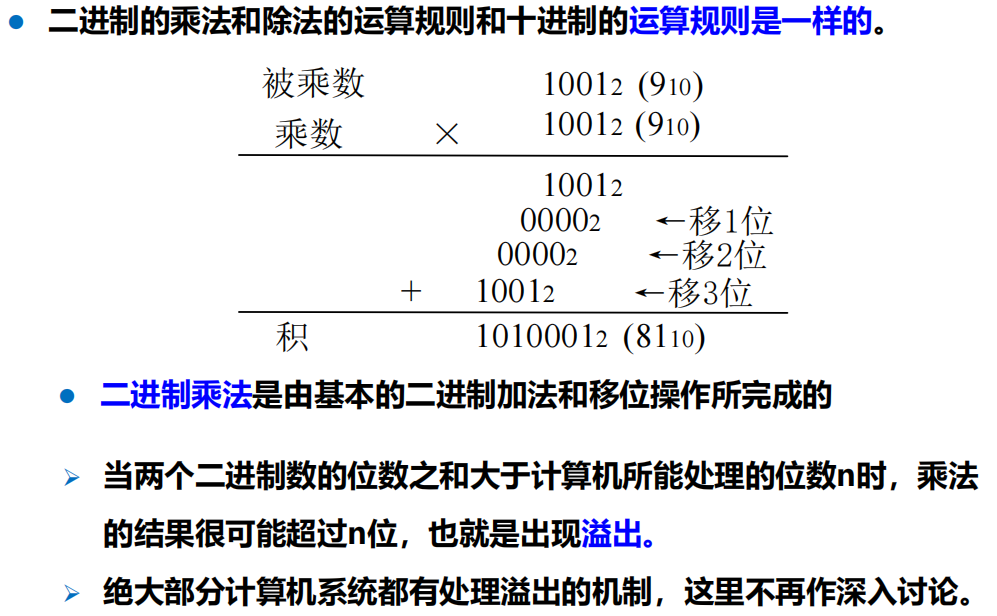
* R进制数转换为十进制数时，将各位数与它的位权乘积相累加，即一个二进制数anan-1...a1a0在十进制中的值A=an×Rn+an-1×Rn-1+…+a1×R1+a0×R0。
* 十进制整数转换成R进制整数：可用十进制整数连续地除以R，每次除法获得的余数即为相应R进制数一位，最后按逆序输出结果。此方法称为“除R取余法”。
* 十进制小数转换成R进制小数：可用十进制的小数连续地乘以R，用得到的整数部分组成R进制的小数，最后按顺序输出结果。此法称为“乘R取整法”。
* 对于二进制数和八进制数、十六进制数之间的转换，有简便快速的“三位一并法”和“四位一并法”。

1. 中央处理器(Central Processing Unit，CPU)是进行各种运算的硬件，是一个非常小的集成电路芯片，通过引脚(pin)与外部连接并交换数据。

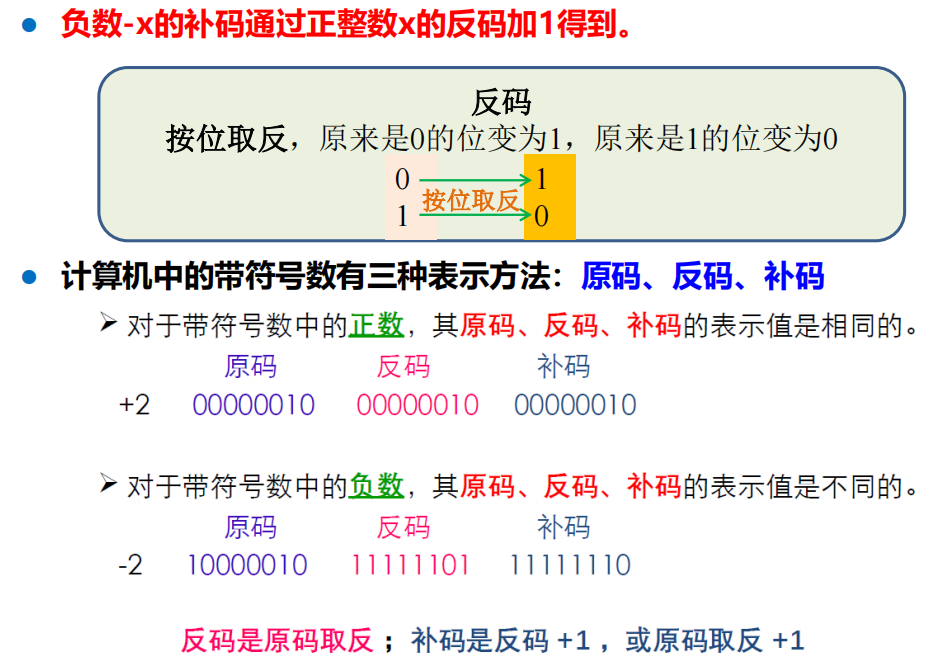
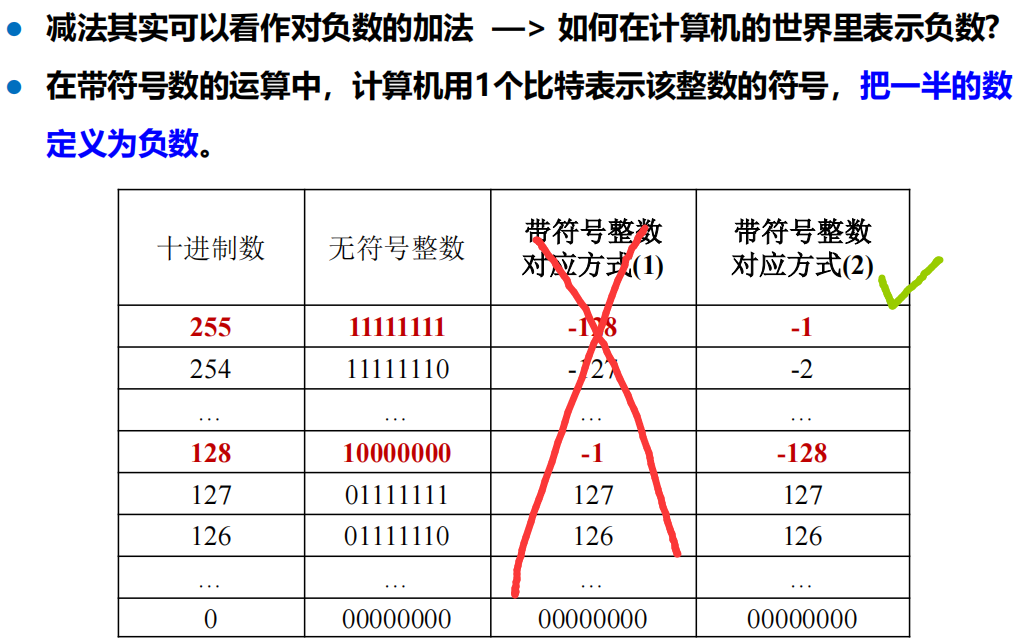
CPU一次只能够处理有限数位的二进制数。现在的计算机一般能一次能处理32个或64个比特的数据，计算机能直接处理的最大的二进制整数是232或264。计算机通常把整数分为两类：

* 无符号整数(unsigned integer)，表示的是非负整数，n位计算机能表示[0, 2n-1]范围内的所有整数；
* 带符号整数(signed integer)，可以表示正整数、负整数和0，因此需要占用一个比特位来表示整数的正负符号，所能表示的正整数范围就会变小。





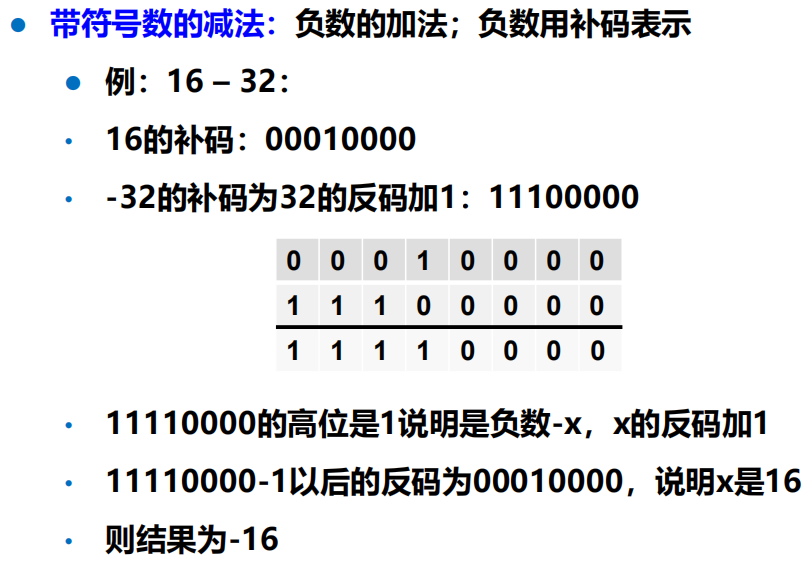
1. 【重要】计算机用第(2)种对应方式表示负数，也就是所谓的补码。



8位二进制无符号数的表示范围为：0~255

8位二进制带符号数的补码表示范围为：-128~127

在用补码方式表示n位带符号整数时，最大数是2n-1-1，最小数是-2n-1



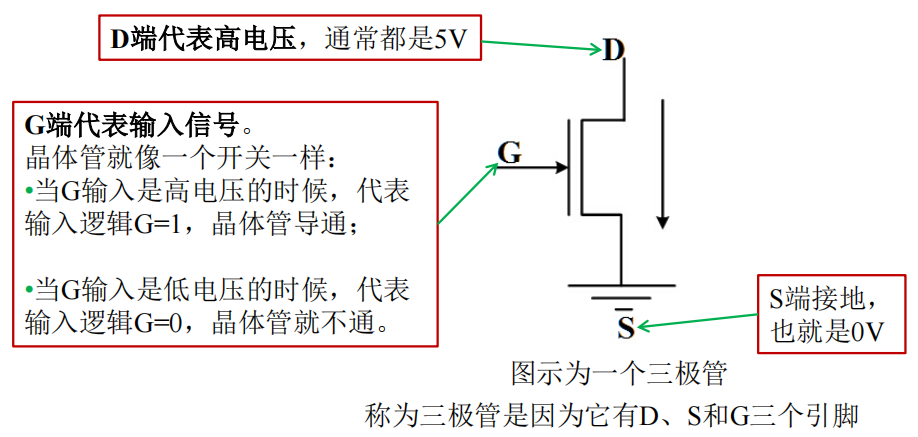
1. 逻辑运算是对逻辑变量和逻辑运算符号的组合序列所作的逻辑推理。

逻辑运算的变量只有两个，它们代表两种对立的逻辑状态。（真—假、是—否、有—无 → 1 — 0）

逻辑运算的基本运算：与（AND）“∧”，或（OR）“∨”，非（NOT）“﹁”

1. 计算机电路中用晶体管实现逻辑。晶体管是以半导体材料为基础的元件，例如各种半导体材料制成的二极管、三极管、场效应管、可控硅等。现在计算机芯片制造者已经可以用大规模生产的方式在几个平方毫米的半导体晶片上实现数百万个晶体管。

每个晶体管可以在不改变自身内部结构的情况下，根据外部电源的变化而展现不同的状态。我们可以通过控制晶体管的电源来控制它们开或关的状态。

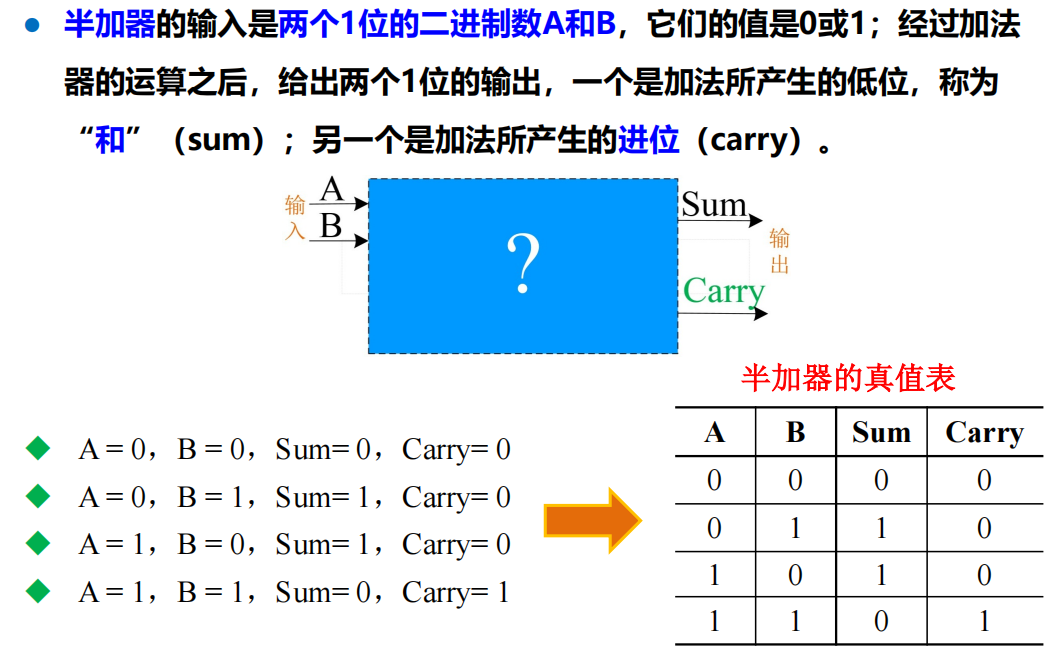
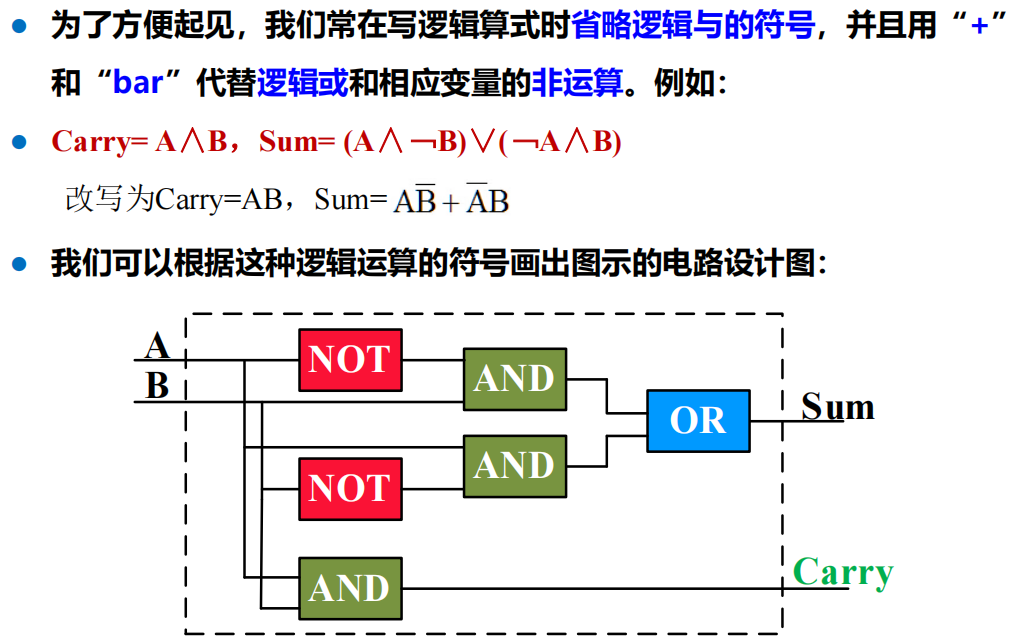


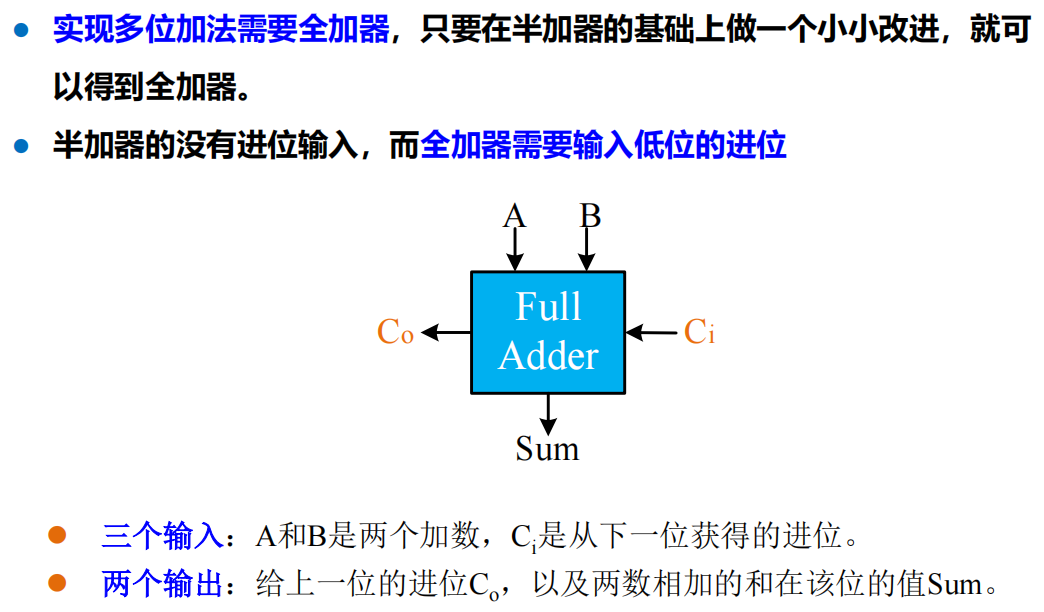
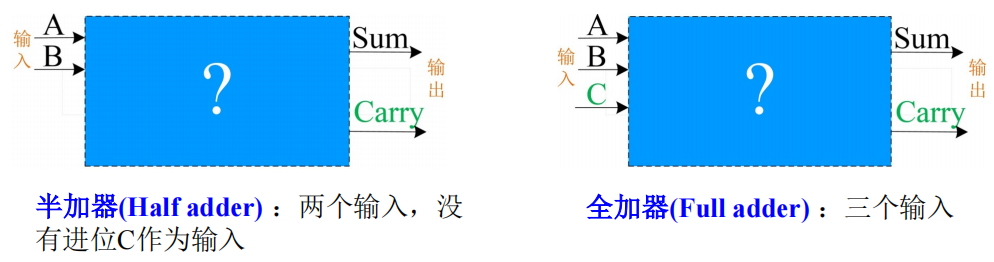
1. 二进制数的加法在计算机里是由每一个比特位的加法组成的

而每一位的的加法都需要3个输入，并产生2个输出

3个输入分别是两个相加位和一个由相邻低位产生的进位

2个输出分别是一个相加得到的二进制数位和一个进位

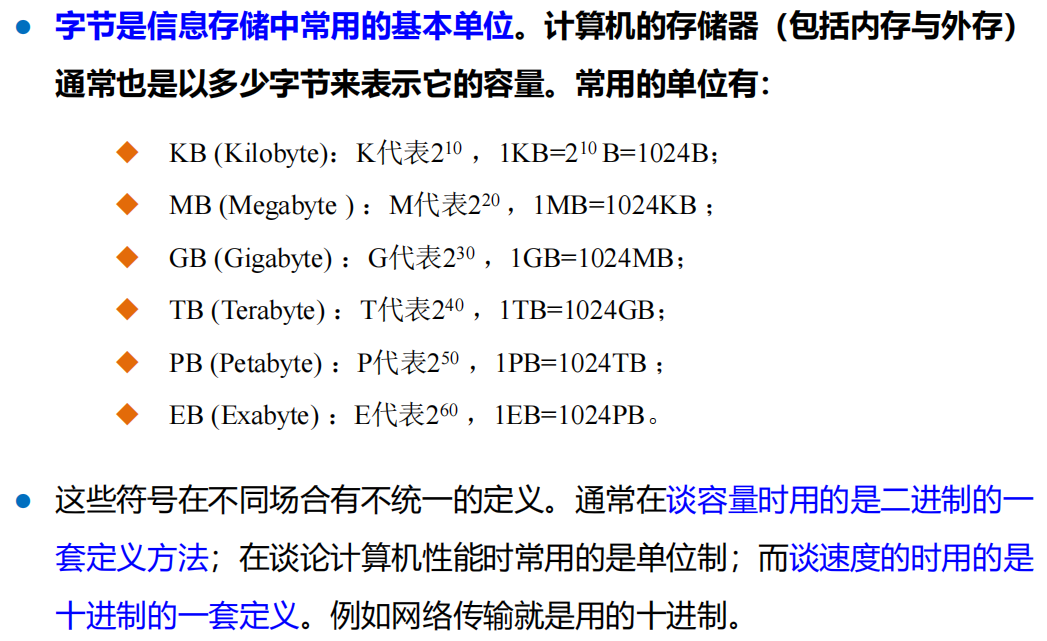
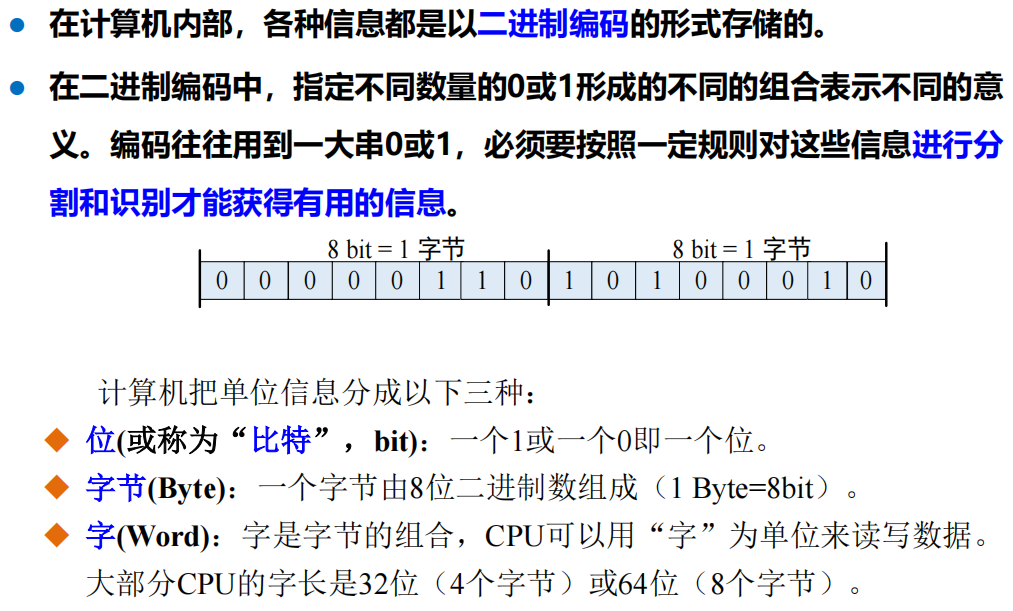
 



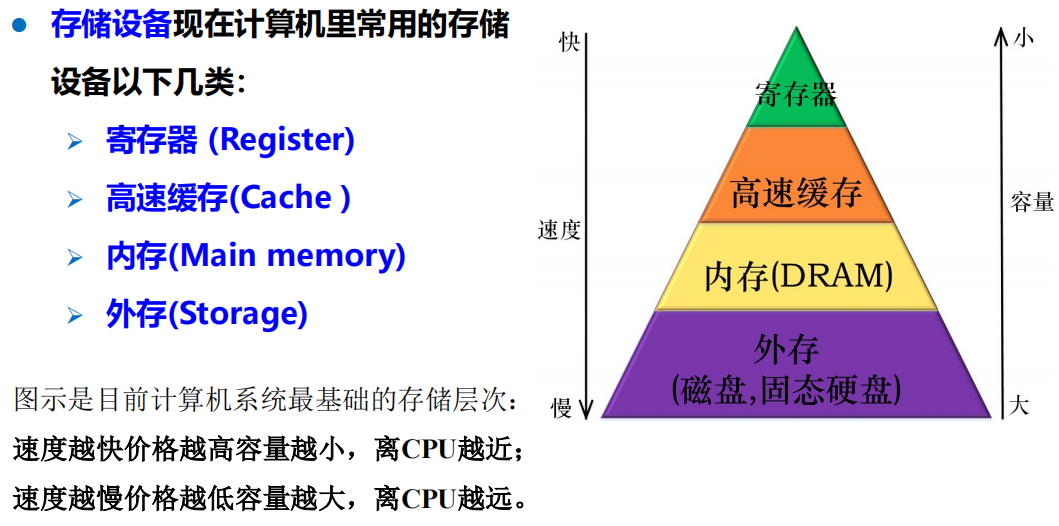
1. 数据的存储形式

计算机用二进制数的组合来表达所有需要保持的信息，这些二进制数的组合按照一定的规则存放就形成了计算机里的数据。

二进制数的数值0或1的存储方式跟随着物理介质的特性不同而不同，基本是利用物理材料的电信号、磁信号之类的状态来存储0或1。这些载体就称为存储介质。存储介质和辅助数据存储和数据读写的电路、设备等组合在一起，构成了存储设备，例如我们常用的内存、磁盘、U盘等。



1. 存储设备 – 存储层次

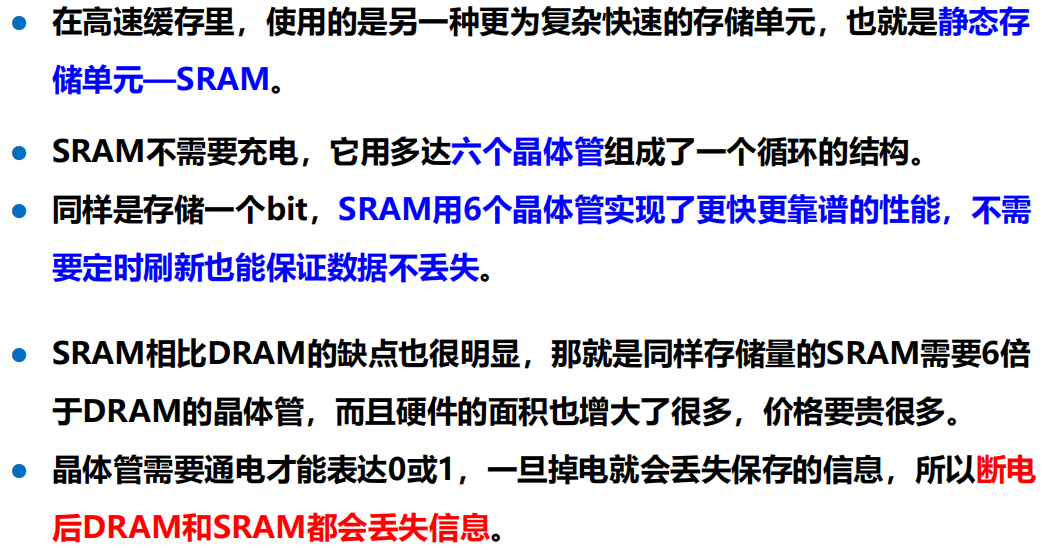
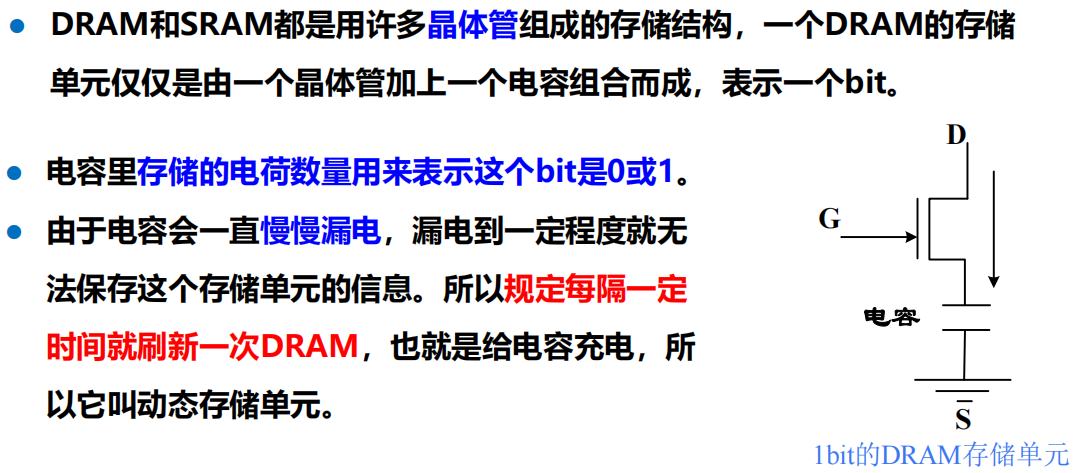


普通运算时需要将数据先放到CPU中的寄存器里，然后ALU对寄存器的值做计算，再存回寄存器里。

程序普通执行时的信息，包括程序指令和很多用到的数据都存放在内存中。

外存的一个显著特征是数据断电后不丢失，而当前以上三种存储在断电后都会丢失数据。

1. 存储设备 – 用晶体管制成DRAM和SRAM



1. 存储设备 – 掉电也能用的存储介质

磁盘、闪存（比如U盘、固态硬盘）。

闪存使用的是一种改进的晶体管。

第三章 程序是如何执行的

1. 计算机中有两个核心部件，分别是CPU和主存（内存）。CPU是做运算的，主存存储程序和相关的变量，每一条程序语句和每一个变量在内存中都有相应的内存地址。

2. CPU中的核心部件

（1）语句地址的存储——程序计数器PC(Program Counter)

（2）CPU中存储程序语句——指令寄存器IR(Instruction Register)

（3）执行运算——算术逻辑单元ALU(Arithmetic Logic Unit)

3. 【重要】汇编指令

* “读取a到R”操作——load指令

格式：load R1, (address) 例如：load R1, (1000) (address)表示address这个地址内存储的**值**。

* “R赋值”操作——mov指令

格式1：mov R1, constant 例如：mov R1, 0Ah 前一个是寄存器，后一个是十六进制常数。

格式2：mov R2, R1 （后赋前）

* 加法指令add

格式1：add R2, R1, constant 即为R2=R1+constant. 例如：add R2,R1,01h

格式2：add R1, R1, R2 即为R1=R1+R2.

* 减法指令 sub

“sub R2, R1，constant”，代表R2 =R1 – constant.

* 左移位指令shiftl

“shiftl R3,R1,05h”代表寄存器R1的二进制数左移5位，移出的那5位填0。将最终值存入R3。

‘05h’也可以用一个寄存器表示，例如“shiftl R3, R1, R2”代表R1的二进制值向左移R2位，存入R3。

左移指令就相当于将R1做乘法。R1左移一位，R1值相当于乘2，R1左移2位，R1值相当于乘4 。

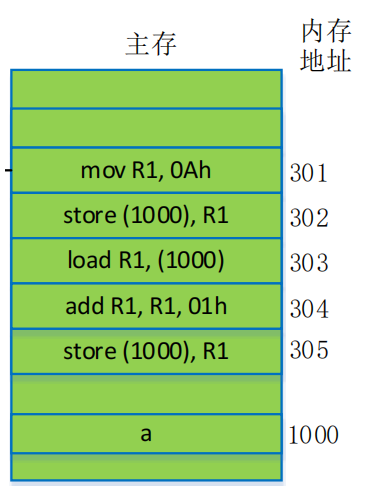
* 右移位指令shiftr

向右移位，移完后的那些最高位填0。

* “将R存回a”操作——store指令

格式：store (address), R1

注：address是内存地址，(address)表示该地址要存回的值，R1是寄存器。也就是(address)=R1。



【演示】a=a+1的完整执行过程

程序开始执行时，变量a存储在主存地址1000处。a=10, a=a+1程序语句有五条汇编指令，从地址301处开始顺序存储每条指令。程序开始执行时，PC指向汇编程序的首地址301处。

* “比较x是否小于y”——slt指令

格式：slt R4, R1, R2

即比较寄存器R1中的数值是否小于R2中的数值，如果小于，则将寄存器R4置1，否则置0。

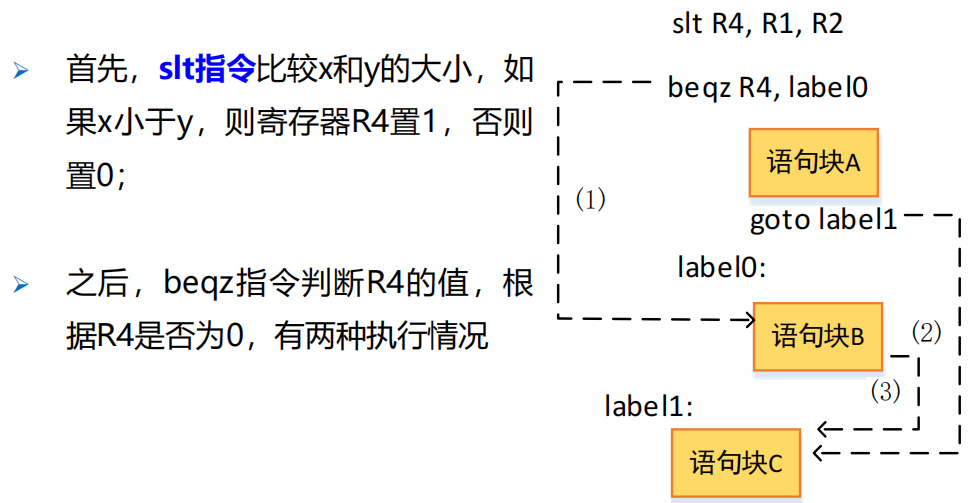
* 判断小于或等于指令——sle指令 （同上）
* “选择跳转语句块”操作——beqz指令

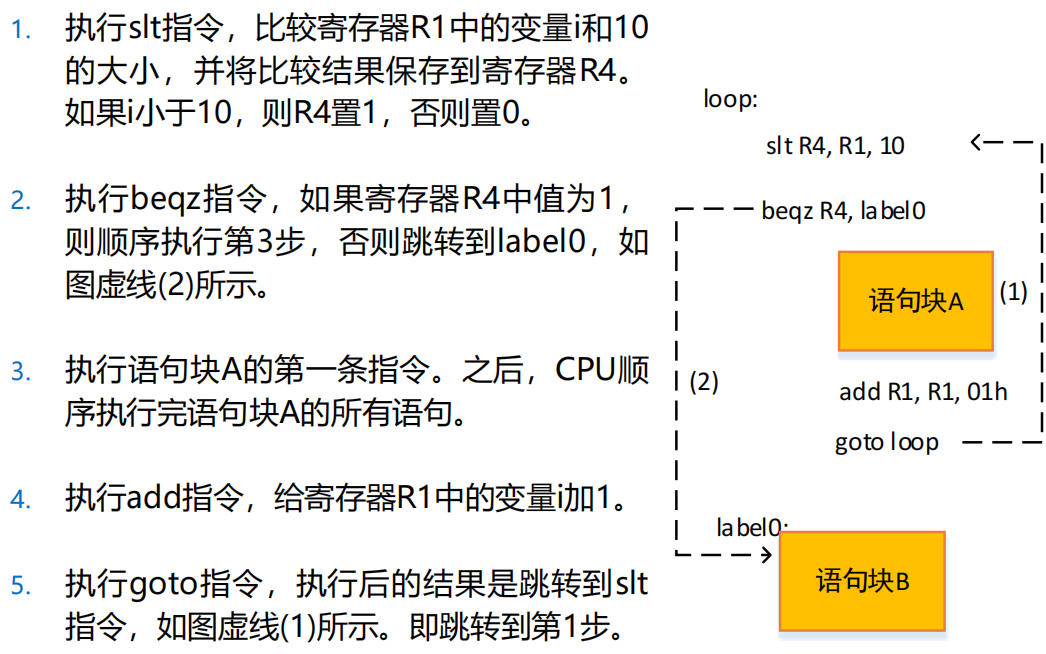
beqz R4, label2表示：如果寄存器R4中的数值为零，则跳转到标签label2标记的指令块处。

* “直接跳转到语句块”操作——goto指令

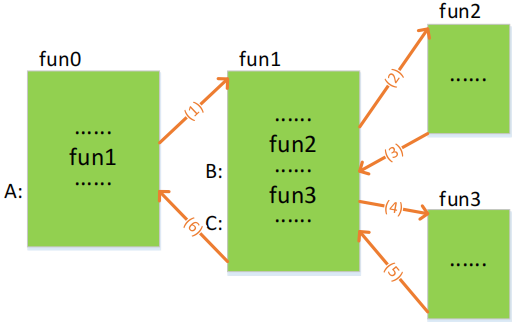
只有一个操作数，即“标签”label。goto label3表示：跳转到标签label3标记的指令处执行。

【应用】if-else分支语句；while循环语句；for循环语句





1. 关于Python的函数调用
2. 函数调用过程的分析

栈：按照先进后出的原则存储数据

主调函数是指调用其他函数的函数；被调函数是指被其他函数调用的函数。一个函数很可能既调用别的函数，又被另外的函数调用。函数调用后返回时，程序会返回到主调函数中调用函数的语句的后一条语句继续执行。

函数调用的特点是：越早被调用的函数，越晚返回。

所以：采用“栈”来保存返回地址。

局部变量如同返回地址般也是存在栈里。

当函数执行时，这个函数的每一个局部变量就会在栈里有一个空间。在栈中存放此函数的局部变量和返回地址的这一块区域叫做此函数的栈帧(frame)。当此函数结束时，这一块栈帧就会被弹出。

第五章 计算思维的核心——算法

1. 递归

动态规划、分治法、贪心算法都是基于递归概念的方法。

* 例：平面划分问题、汉诺塔问题。

递归是一个过程或函数在它的定义或说明中又直接或间接调用它自己的一种方法。

递归本质是把一个复杂的大问题层层转化为一个与原问题相似的小问题，利用小问题的解来构筑大问题的解。学习用递归解决问题的关键就是找到问题的递归式。

在使用递归解决问题时要特别注意，一定要有一个明确的递归结束条件，否则就会陷入无限循环中。

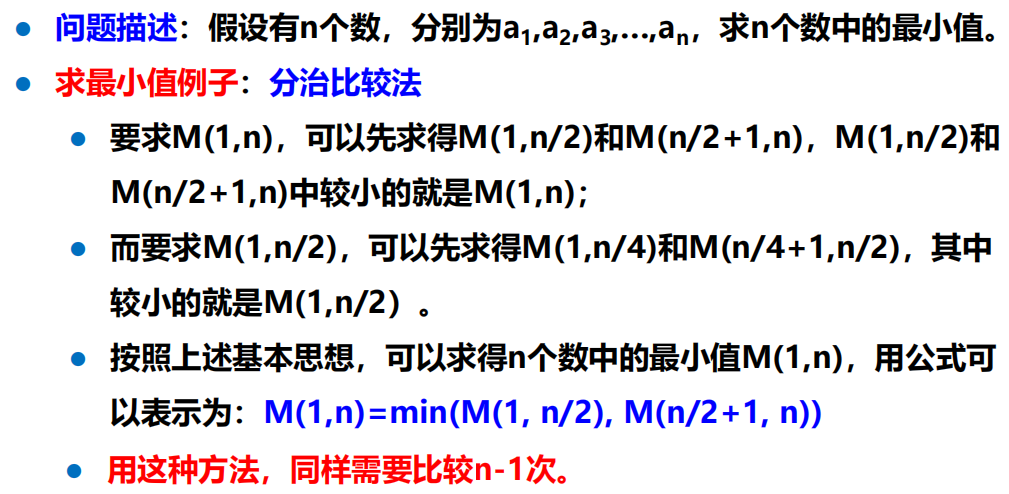
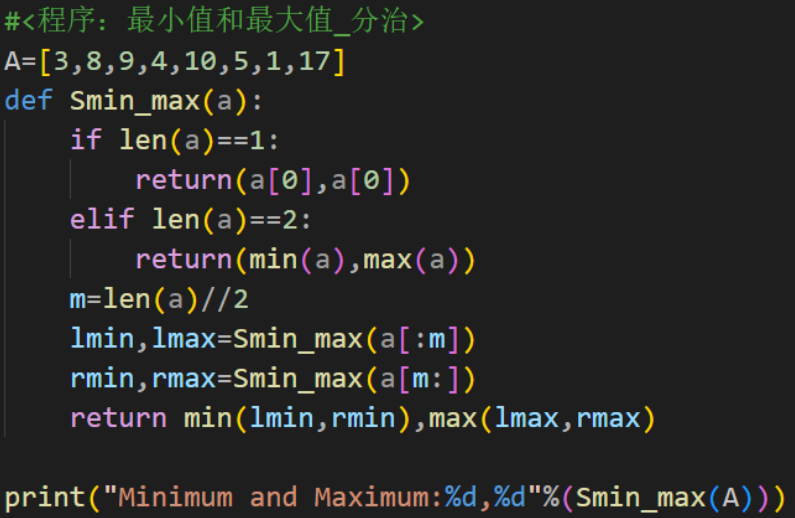
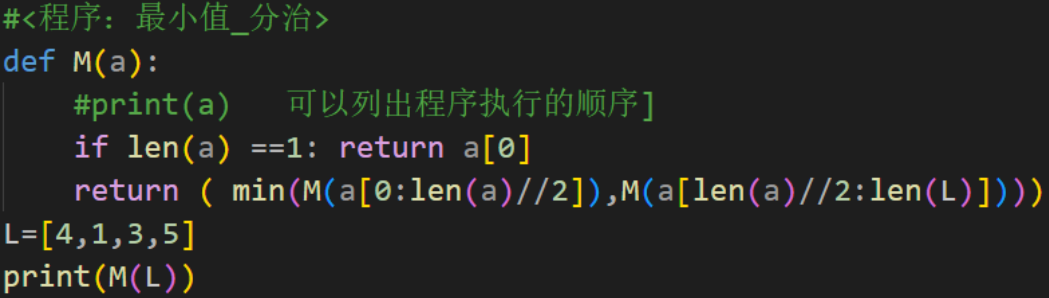
写递归程序的诀窍就是：（1）怎么分，怎么合；（2）怎么终止。

1. 分治法

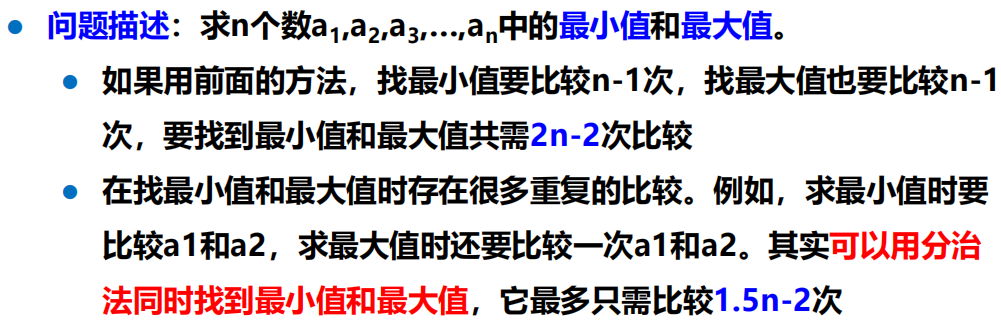
* 例：求n个数中的最小值

循环比较法：(n-1)次；递归比较法：(n-1)次；

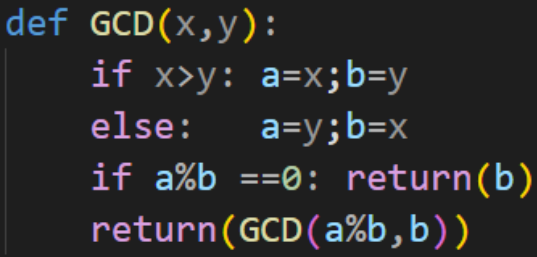
分治比较法：(n-1)次。但是这种方法可能在多核的情况下要比前两种方法的效率高。



* 例：同时求min和max：(1.5n-2)次



1. 贪心算法（贪婪算法）：用来求解最优化问题的一种方法

一般来说，求解最优化问题的过程就是做一系列决定从而实现最优值的过程。最优解就是实现最优值的这些决定。贪心算法考虑局部最优，每次都做当前看起来最优的决定，得到的解不一定是全局最优解，但是有些问题能够用贪心算法求得最优解。

* 例：求两个正整数x和y的最大公约数

重要性质：如果a是x和y的最大公约数并且**x>y**，那么a也是**x-y和y的最大公约数**。（15和10的最大公约数是5，15-10=5，5和10的最大公约数也是5。）

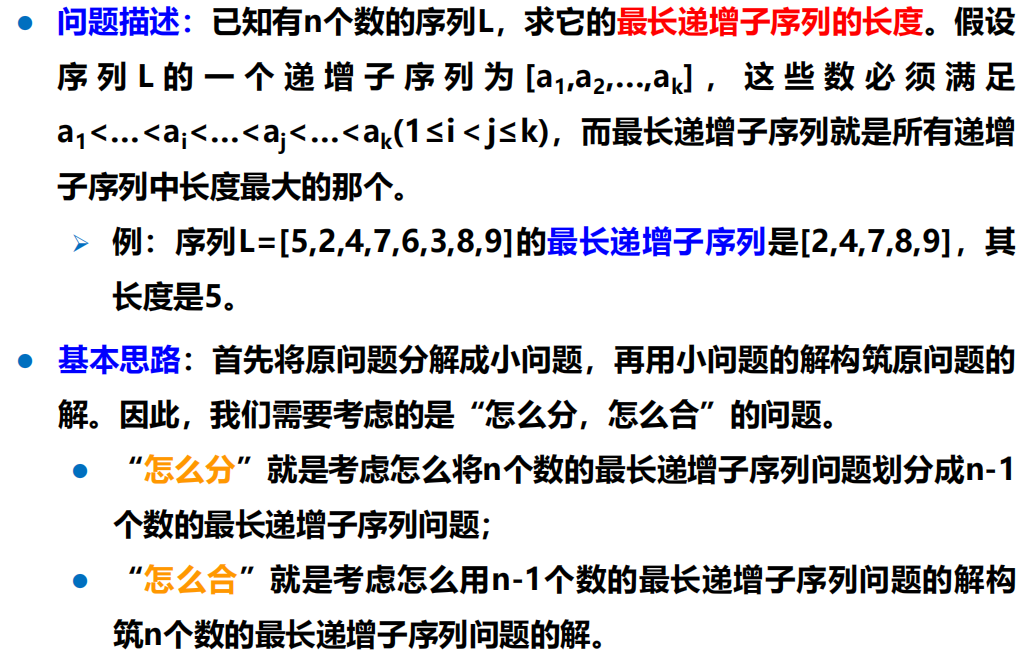
两个整数的最大公约数等于其中较小的数和两数相除余数的最大公约数。

* 对比：

贪心算法考虑局部最优，每次都做当前看起来最优的决定，得到的解不一定是全局最优解。

动态规划考虑全局最优，得到的解一定是最优解。

1. 动态规划

动态规划与分治法类似，其基本思想也是将待求解问题分解成若干个子问题，先求解子问题，然后从这些子问题的解得到原问题的解。

与分治法不同的是，适合于用动态规划求解的问题，经分解得到子问题往往不是互相独立的，即子问题之间具有重叠的部分。在这种情况下，如果用分治法求解就会重复的求解这些重叠的部分。

动态规划只会对这些重叠的部分求解一次并用表格保存这些解，如此一来就可以避免大量的重复计算。

* 例：最长递增子序列问题

1. 算法效率的度量：用依据该算法编制的程序在计算机上执行所消耗的时间来度量。

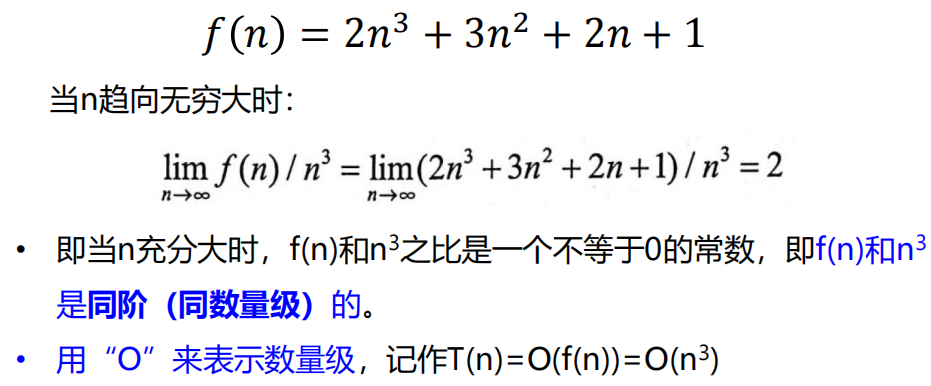
（1）事后统计：利用计算机内的计时功能，不同算法的程序可以用一组或多组相同的统计数据区分

（2）事前分析估计：依据的算法选用何种策略、问题的规模、程序语言、编译程序产生机器代码质量、机器执行指令速度。

1. 时间复杂度的**渐进表示法**

只需要考虑当问题规模充分大时，算法中基本语句的执行次数在渐近意义下的阶。

算法的时间复杂度：算法中基本语句重复执行的次数是问题规模n的某个函数f(n),算法的时间量度记作：T(n)=O(f(n)). 表示随着n的增大，算法执行的时间的增长率和f(n)的增长率相同，称渐近时间复杂度。

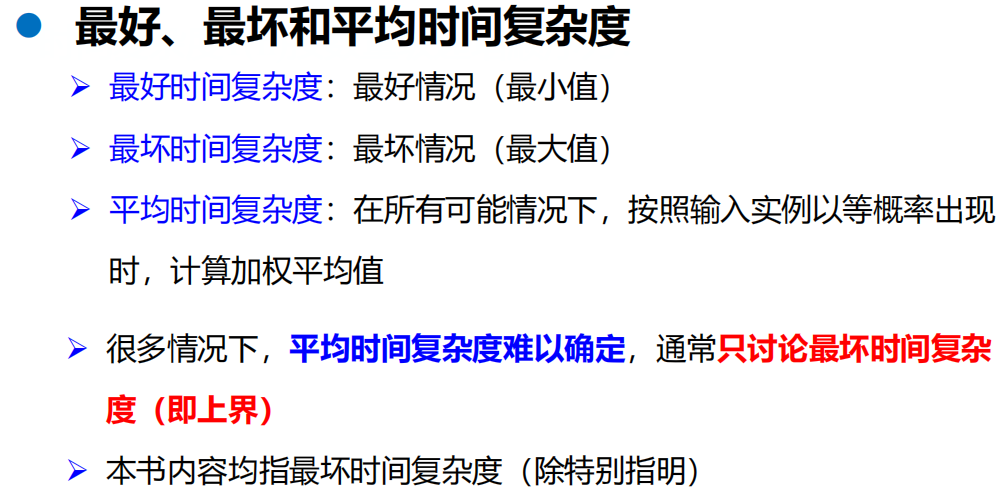
【重要】分析算法时间复杂度的基本方法

→找出语句频度最大的那条语句作为基本语句

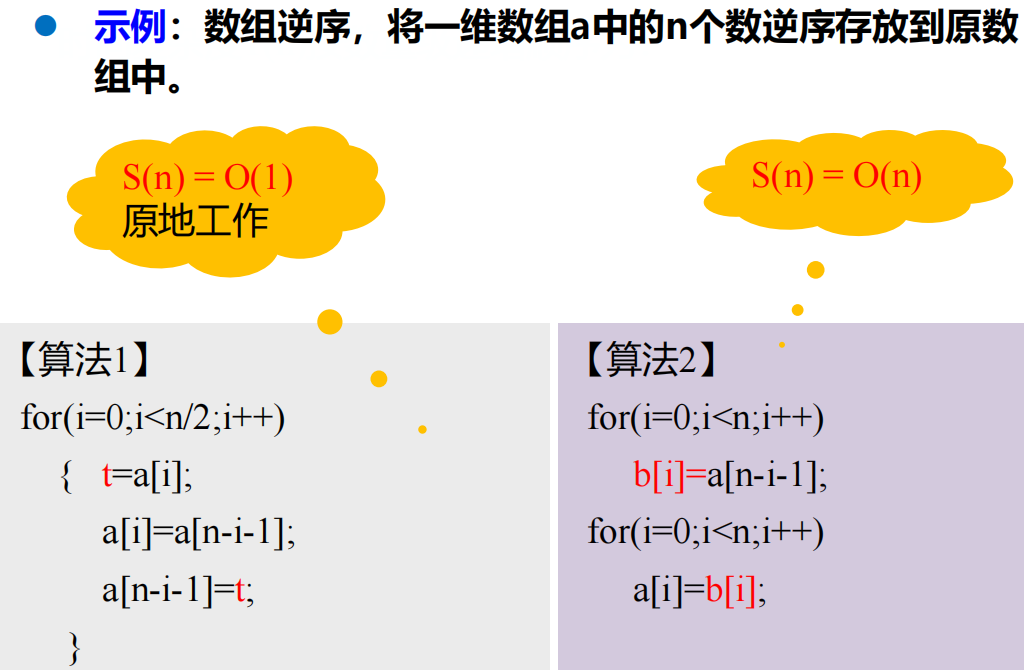
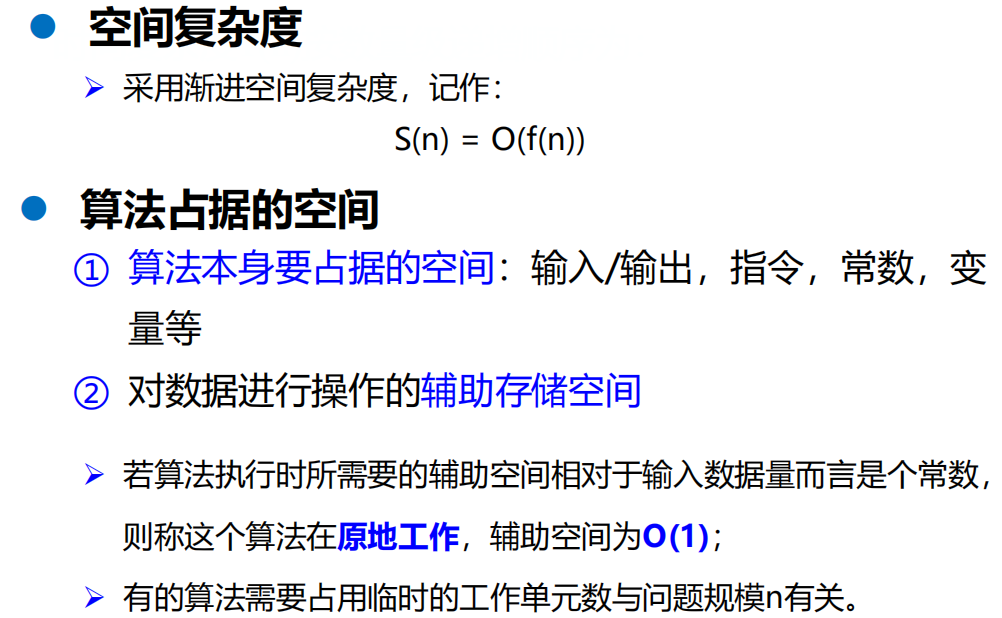
→计算基本语句的频度得到问题规模n的某个函数f(n)

→取其数量级用符号“O”表示

时间复杂度是由嵌套最深层语句的频度决定的。



1. 空间复杂度



第六章 操作系统简介

1. BIOS

所有设备在开机启动过程中都包括三个阶段：启动自检阶段、初始化启动阶段、启动加载阶段。

这三个阶段主要由BIOS (Basic Input Output System)来完成的。

BIOS是一组程序，包括基本输入输出程序、系统设置信息、开机后自检程序和系统自启动程序。

（1）启动自检阶段

电脑刚接通电源，将读取BIOS程序，并对硬件进行检测。

这个检测过程也叫做加电自检（Power On Self Test, POST）。功能是检查电脑整体状态是否良好。

（2）初始化启动阶段

根据BIOS设定的启动顺序，找到优先启动的设备.

（3）启动加载阶段

BIOS会指定启动的设备来读取硬盘中的操作系统核心文件。

由于不同的操作系统具有不同的文件系统格式（如FAT32，NTFS，EXT4等等），因此需要一个启动管理程序来处理核心文件的加载，这个启动管理程序就被称为Boot Loader。

（4）内核装载阶段

操作系统利用内核程序，开始测试并驱动各个外围设备，包括存储装置、CPU、网卡、声卡等。

（5）登录阶段

计算机主要完成以下两项任务：启动所有需要自动启动的Windows服务；显示登录界面。

1. 操作系统（计算机管理控制程序）

操作系统（OS）是管理计算机硬件与软件资源的计算机程序，是软件与硬件的中间接口。

操作系统很喜欢sleep！所以要叫醒它！叫醒操作系统的方式叫做“中断”。中断的来源有三种。

【重要】中断的来源

* 硬件中断（Hardware Interrupt）：指由硬件发出的中断，包括I/O设备发出的数据交换请求、时钟中断等等。
* 软件中断（Software Interrupt）：指由应用程序触发的中断，就是正在执行的软件需要操作系统提出服务。软件中断主要包括各种系统调用（system calls），为应用程序提供不同的服务。
* 异常（Exception）：指当系统运行过程中出现了一些非正常事务，需要操作系统进行处理。例如用户程序读写一个地址，而这地址被保护起来，是不能被用户程序读写的，这也会发生异常中断。但是，异常并不全是错误。

1. 操作系统要管理的硬件资源最主要包括：各种各样的I/O设备、计算资源、存储资源。

（1）I/O设备：可以与计算机进行数据传输的硬件。指键盘，显示器，U盘等这些常用的设备，操作系统需要统一对这些硬件进行管理。

（2）计算资源：主要指CPU。CPU通常使用轮询和硬件中断两种方式检测设备的工作状态。

轮询方式的三个弊端：1）检测中断速度慢；2）可能存在设备处于“饥饿”状态，某一个设备有请求却一直得不到CPU的响应；3）系统处理中断事务不灵活。

（3）存储资源：通常包括内存和外存，内存是CPU直接通过系统总线来访问的，而外存是通过标准的I/O来管理的。

1. 驱动程序

任何新硬件如果要连接到计算机，必须提供驱动程序（Device Driver）。

1. 内核态（Kernel Mode）与用户态（User Mode）

绝对不允许用户程序直接访问硬件设备。必须要CPU提供硬件的支持。

禁止用户程序直接访问硬件设备的基本思想是CPU将指令集分为需要特权的指令（Privileged）和一般的指令(Non-Privileged)。而所有的I/O指令都是属于需要特权的指令。一般用户不能执行这类privileged指令，必须是系统内核才能执行这类privileged 指令。

在CPU有个特殊的寄存器叫做状态寄存器（Status Register），显示现在的CPU是在内核态还是用户态。假如CPU是在用户态，那么任何的privileged 指令都不可以执行，一旦执行了，CPU就发生异常错误。

CPU如何从用户态转成内核态，这是现代操作系统的一个重要的技术：必须要使用“中断”方式，只有中断，CPU才会进入内核态。

1. 系统调用（System Call）—— 软件中断

操作系统中设置了一组用于实现系统功能的子程序，称为系统调用函数。

系统调用函数的操作**一定运行于内核态**，而普通的函数调用由函数库或用户自己提供，运行于用户态。

（3）常用系统调用

① 进程控制，如：fork()，exit()等。

② 文件系统操作控制，如：read()，write()等。

③ 系统控制，如：ioctl()，time()等。

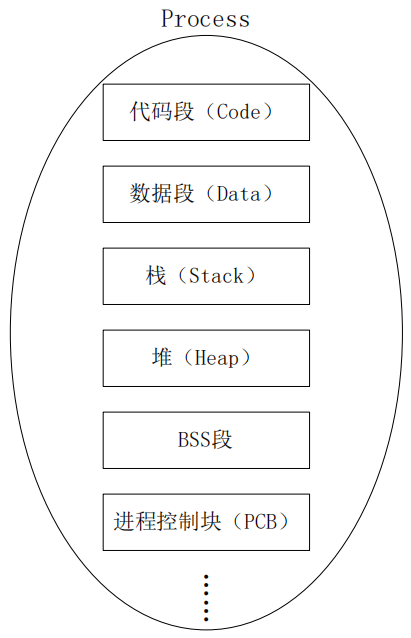
④ 内存管理，如：mmap()，mprotect()等。

⑤ 网络管理控制，如：sethostname()，socket()，bind()等。

⑥ 用户管理，如：getuid()，getgid()等。

⑦ 进程间通信，如：signal()，kill()等。

在今后的学习工作中，文件操作是最为常用，包括文件的打开、创建、读、写等系统调用。

1. 进程

进程调度：在系统进行任务的换入，换出操作时，需要确定哪些任务需要换入CPU中执行，而哪些任务需要继续等待。

进程是一个程序的一次执行，包含了其执行时所有的环境信息。程序运行首先需要将源代码转化为可执行程序，其次还需要操作系统为其提供一个运行环境，而这个运行环境就是进程。

一个进程包含了代码段、数据段、栈、堆、BSS段以及进程控制块等部分。

* 代码段通常是指用来存放程序执行代码的一块内存区域。
* 数据段通常是指用来存放程序中已经初始化的全局变量的一块内存区域。
* 栈（Stack）是用户存放程序临时创建的局部变量区域。
* 堆（Heap）是用于存放进程运行中动态分配的内存段，大小并不固定，可根据需要动态扩张或缩减。
* BSS段（Block Started by Symbol）通常是指用来存放程序中未初始化的全局变量的一块内存区域。
* 为了统一管理进程，进程控制块（Process Control Block，PCB），用来记录进程的特征信息，描述进程运动变化的过程。PCB是操作系统感知进程存在的唯一标识，进程与PCB是一一对应的。

1. 进程状态——“三状态模型”

