软件设计师考前 20 问

第 1 问 地址码为什么不存入程序计数器（PC）中？

答：首先要区分地址码和地址，地址码属于指令的构成部分，而地址一般是指存储的指令地址。其次需要弄清楚 PC 和 IR 的区别，程序计数器 PC 是存放下一条指令的地址，而指令寄存器是存放正在执行的指令。指令实质上是由操作码和地址码两部分进行组成，自热对应整个指令部分，故都存放在指令寄存器（IR）中。

第 2 问 流水线技术中的吞吐率怎么算？

答：首先流水线的吞吐率有实际吞吐率和最大吞吐率之分。

（1）实际吞吐率：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。公式：吞吐率=指令条数/流水线执行时间。例如 10 条指令，流水线执行时间为 35，那吞吐率=10/35。

（2）最大吞吐率：为流水线周期（指令分段执行中时间最长的一段）的倒数。例如 10 条指令，流水线周期为 5，那它的最大吞吐率=1/5（跟指令条数没有关系）。

第 3 问 CISC 和 RISC 的各自特点是什么？

答：（1）CISC（复杂指令集）的特点：指令数量多，指令频率差别大，变长，多种寻址方式，使用微码（微程序）实现。

（2）RISC（精简指令集）的特点：指令数量少，频率接近，定长，单周期，多寄存器寻址，多通用寄存器，硬布线逻辑控制，适用于流水线。有效支持高级程序语言，优化编译。

第 4 问 二叉树的特性有哪些？

答：（1）在二叉树的第 i 层上最多有 2i-1 个结点（i≥1）；

（2）深度为 k 的二叉树最多有 2k -1 个结点（k≥1）；

（3）对任何一棵二叉树，如果其叶子结点数为 n0，度为 2 的结点数为 n2，则 n0=n2+1；

（4）如果对一棵有 n 个结点的完全二叉树的结点按层序编号（从第 1 层到$\lfloor log_2 n \rfloor + 1$层，每层从左到右），则对任一结点 i（1≤i≤n），有：

如果 i=1，则结点 i 无父结点，是二叉树的根；如果 i>1，则父结点是⌊i/2⌋；

如果 2i>n，则结点 i 为叶子结点，无左子结点；否则，其左子结点是结点 2i；

如果 2i+1>n，则结点 i 无右子叶点，否则，其右子结点是结点 2i+1。

第 5 问 图的遍历方式有几种？

答：（1）深度优先遍历：首先访问出发顶点 V；依次从 V 出发搜索 V 的任意一个邻接点 W；W 未访问过，则从该点出发继续深度优先遍历。

（2）广度优先遍历：首先访问出发顶点 V；然后访问与顶点 V 邻接的全部未访问顶点 W、X、Y...；然后再依次访问 W、X、Y...邻接的未访问的顶点。

两者区别在于一个不断的去遍历临近的 1 个结点，直达临近结点遍历完成，才会有返回的过程。而另外一个呢，一次性遍历该结点的所有临近结点，依次类推。回溯法对应深度优先，而分支限界法对应广度优先。

第 6 问 常见的常见算法策略其各个特点是什么？

答：

算法名称	关键点	特征	典型问题
分治法	递归技术	把一个问题拆分成多个小规模的相同子问题，一般可用递归解决。	归并排序、快速排序、二分搜索
贪心法	一般用于求满意解，特殊情况可求最优解（部分背包）	局部最优，但整体不见得最优。每步有明确的、既定的策略。	背包问题（如装箱）、多机调度、找零钱问题
动态规划法	最优子结构和递归式	划分子问题（最优子结构），并把子问题结果使用数组存储，利用查询子问题结果构造最终问题结果。	矩阵乘法、背包问题、LCS 最长公共子序列

回溯法	探索和回退	系统地搜索一个问题的所有解或任一解。有试探和回退的过程。	N 皇后问题、迷宫、背包问题

第 7 问 常见的排序算法有哪些，其特点、时间复杂度、空间复杂度是多少？

答：（1）直接插入排序：即当插入第 i 个记录时，R1，R2，…，Ri-1 均已排好序，因此，将第 i 个记录 Ri 依次与 Ri-1，…，R2，R1 进行比较，找到合适的位置插入。

（2）希尔排序：先取一个小于 n 的整数 d1 作为第一个增量，把文件的全部记录分成 d1 个组。所有距离为 d1 的倍数的记录放在同一个组中。先在各组内进行直接插入排序；然后，取第二个增量 d2<d1 重复上述的分组和排序，直至所取的增量 dt=1（dt<dt-1<…<d2<d1），即所有记录放在同一组中进行直接插入排序为止。

（3）冒泡排序：通过相邻元素之间的比较和交换，将排序码较小的元素逐渐从底部移向顶部。

（4）快速排序：采用分治法，其基本思想是将原问题分解成若干个规模更小但结构与原问题相似的子问题。通过递归解决这些子问题，然后再将这些子问题的解组合成原问题的解。

（5）选择排序：首先在所有记录中选出排序码最小的记录，把它与第 1 个记录交换，然后在其余的记录内选出排序码最小的记录，与第 2 个记录交换，依次类推，直到所有记录排完为止。

（6）堆排序：基本步骤先开始初建堆，其次取出堆顶元素，然后重建堆，最后再完成排序。

（7）归并排序：归并也称为合并，是将两个或两个以上的有序子表合并成一个新的有序表。若将两个有序表合并成一个有序表，则称为二路合并。

（8）基数排序：基数排序是一种借助多关键字排序思想对单逻辑关键字进行排序的方法。基数排序不是基于关键字比较的排序方法，它适合于元素很多而关键字较少的序列。

类别	排序方法	时间复杂度		空间复杂度	稳定性
		平均情况	特殊情况	辅助存储	
插入排序	直接插入	O（n^2）	基本有序最优 O（n）	O（1）	稳定
	Shell 排序	O（$n^{1.3}$）	-	O（1）	不稳定
选择排序	直接选择	O（n^2）	-	O（1）	不稳定

	堆排序	O（$nlog_2n$）	-	O（1）	不稳定
交换排序	冒泡排序	O（n^2）	基本有序最优 O（n）	O（1）	稳定
	快速排序	O（$nlog_2n$）	基本有序最差（n^2）	O（1）	不稳定
归并排序		O（$nlog_2n$）	--	O（n）	稳定
基数排序		O（d（n+rd））	--	O（rd）	稳定

第 8 问 PV 操作都必须要成对出现吗？

答：PV 一定是成对出现的，PV 可以简单理解为加锁和解锁，如果只加锁也就是只有 P 操作，可能会形成死锁。如果没有加锁，只有解锁也就是 V 操作，那么相当于系统中资源无端增多，也就没有意义了。

以打印机为例，如果对打印机加上 PV 操作，P 操作相当于，每个人用之前都锁定一次打印机（S=S-1），此时其他人去打印时进行申请，会发现资源不足（S<0），此时就进入阻塞队列去排队了；而 V 操作相当于，能够使用打印机的人，用完之后，之前占有的打印机就会释放给其他人用（S=S+1），此时如果发现有人排队（S<=0），就会通知排队的人，可以使用了。

如果只占有不释放，很明显，排队的人只会持续暴增；如果只释放，那么信号量 S 无端增加 1，而不会减少，也就没有意义了。

第 9 问 编译程序和解释程序各自的特点和区别是什么？

答：

		编译型语言	解释型语言
共同点		高级语言程序	
		词法分析、语法分析、语义分析	
不同点	翻译程序	编译器	解释器
	是否生成目标代码	生成	不会生成
	目标程序直接执行	直接执行	边解释边执行

	翻译是否参与	不参与执行	解释器参与执行
	执行效率	高	低
	灵活性与可移植性	灵活性差，可移植性差	灵活性好，可移植性强

第 10 问 在数据流图中父图与子图之间平衡是什么意思？

答：父图中某个加工的输入输出数据流必须与其子图的输入输出流在数量和名字上相同，父图的一个输入（或输出）数据流对应子图中几个输入（或输出）数据流，而子图中组成的这些数据流的数据项全体正好是父图中的这一个数据流。

第 11 问 UML 类图中的关系有几种以及其各自含义？

答：（1）依赖关系：一个事物发生变化影响另一个事物，例如：y=x+1，我们就说 y 依赖于 x，y 的值随着 x 的值变化而变化。

（2）泛化关系：父子关系，一般与特殊的关系，例如动物类和猫类。

（3）关联关系：两者之间用链进行连接，一般表示两个类进行通信。

（4）聚合和组合关系：强调部分与整体关系，前者强调生命周期不同，后者强调生命周期相同，比如：大雁和翅膀属于组合关系，大雁和雁群属于聚合关系。

（5）实现关系：是接口和类的关系。

第 12 问 如何区分泛化关系？

答：泛化关系一般可以体现为 is-a 或者 has-a，也就是说，泛化关系可以理解为典型的父类-子类关系，其次还可以理解为 A 有 n 种 B 的情况下，前者是后者的泛化也就是父类。泛化关系中，父类是抽象用例，抽象后，必须且只能选择一种子类用例去执行。

对比来看，用例之间的关系（用例之间存在 3 种关系--包含、扩展、泛化）

包含关系<<include>>：将用例中的一部分行为抽取出来作为单独的用例，对于被包含的用例，是必须选择的用例；

扩展关系<<extend>>：将某个条件下可能的行为作为扩展用例，是某个条件下可以选择的扩展；

泛化关系：将用例中的一部分公共部分抽象出来作为父用例，对于这些子用例，必须选择其中一个用例。（注意与包含关系的区分）

第 13 问 23 种设计模式的分类和定义？

答：

	创建型	结构型	行为型	
类	factory method 工厂方法模式	adapter 适配器模式（类和对象）	template method 模板方法模式	interpreter 解释器模式
对象	abstract factory 抽象工厂模式 prototype 原型模式 singleton 单例模式 builder 构建器模式	bridge 桥接模式 composite 组合模式 decorator 装饰模式 facade 外观模式 flyweight 享元模式 proxy 代理模式	chain of responsibility 职责链模式 command 命令模式 iterator 迭代器模式 mediator 中介者模式	memento 备忘录模式 observer 观察者模式 state 状态模式 strategy 策略模式 visitor 访问者模式

- 工厂方法模式（Factory Method）：定义一个创建对象的接口，但由子类决定需要实例化哪一个类。工厂方法使得子类实例化的过程推迟。
- 抽象工厂模式（Abstract Factory）：提供一个接口，可以创建一系列相关或相互依赖的对象，而无需指定它们具体的类。

- 原型模式（Prototype）：用原型实例指定创建对象的类型，并且通过拷贝这个原型来创建新的对象。
- 单例模式（Singleton）：保证一个类只有一个实例，并提供一个访问它的全局访问点。
- 构建器模式（Builder）：将一个复杂类的表示与其构造相分离，使得相同的构建过程能够得出不同的表示。
- 适配器模式（Adapter）：将一个类的接口转换成用户希望得到的另一种接口。它使原本不相容的接口得以协同工作。
- 桥接模式（Bridge）：将类的抽象部分和它的实现部分分离开来，使它们可以独立地变化。
- 组合模式（Composite）：将对象组合成树型结构以表示“整体-部分”的层次结构，使得用户对单个对象和组合对象的使用具有一致性。
- 装饰模式（Decorator）：动态地给一个对象添加一些额外的职责。它提供了用子类扩展功能的一个灵活的替代，比派生一个子类更加灵活。
- 外观模式（Facade）：定义一个高层接口，为子系统中的一组接口提供一个一致的外观，从而简化了该子系统的使用。
- 享元模式（Flyweight）：提供支持大量细粒度对象共享的有效方法。
- 代理模式（Proxy）：为其他对象提供一种代理以控制这个对象的访问。
- 职责链模式（Chain of Responsibility）：通过给多个对象处理请求的机会，减少请求的发送者与接收者之间的耦合。将接收对象链接起来，在链中传递请求，直到有一个对象处理这个请求。
- 命令模式（Command）：将一个请求封装为一个对象，从而可用不同的请求对客户进行参数化，将请求排队或记录请求日志，支持可撤销的操作。
- 解释器模式（Interpreter）：给定一种语言，定义它的文法表示，并定义一个解释器，该解释器用来根据文法表示来解释语言中的句子。
- 迭代器模式（Iterator）：提供一种方法顺序访问一个聚合对象中的各个元素，而不需要暴露该对象的内部表示。
- 中介者模式（Mediator）：用一个中介对象来封装一系列的对象交互。它使各对象不需要显式地相互调用，从而达到低耦合，还可以独立地改变对象间的交互。
- 备忘录模式（Memento）：在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态，从而可以在以后将该对象恢复到原先保存的状态。
- 观察者模式（Observer）：定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并自动更新。
- 状态模式（State）：允许一个对象在其内部状态改变时改变它的行为。

- 策略模式（Strategy）：定义一系列算法，把它们一个个封装起来，并且使它们之间可互相替换，从而让算法可以独立于使用它的用户而变化。
- 模板方法模式（Template Method）：定义一个操作中的算法骨架，而将一些步骤延迟到子类中，使得子类可以不改变一个算法的结构即可重新定义算法的某些特定步骤。
- 访问者模式（Visitor）：表示一个作用于某对象结构中的各元素的操作，使得在不改变各元素的类的前提下定义作用于这些元素的新操作。

第 14 问 对于软件设计中模块设计的过程需要遵循的原则，除了“高内聚低耦合”还有哪些？

答：对于模块设计除了“高内聚，低耦合”，还要适当考虑权衡适量的原则，模块大小适中，适宜的系统深度和宽度比例，尽可能减少调用的深度，适度控制模块的扇入扇出，模块的作用域应该在模块之内等。

第 15 问 在著作权中，哪些权利会永久保护？

答：署名权、修改权、保护作品完整权。

第 16 问 在计算中，常见的防御手段有哪些？

答：（1）防火墙技术：主要了解它的机制是防外不防内，对于 DMZ 非军事区主要放置应用服务器（如邮件服务器，WEB 服务器）。

（2）漏洞扫描：入侵者可以利用系统漏洞侵入系统，系统管理员可以通过漏洞扫描技术，及时了解系统存在的安全问题，并采取相应措施来提高系统的安全性。

（3）入侵检测 IDS：基于数据源的分类——审计功能、记录安全性日志。基于检测方法——异常行为检测。

第 17 问 关系模式如何判断其规范化程度？

答：根据范式的判断依据，首先了解清楚一些基本概念：候选键、主属性、非主属性、部分函数依赖、传递函数依赖、函数依赖集合、函数依赖的决定因素。

然后再根据定义一步一步地深入判断：

属性不可再分则满足 1NF；

在 1NF 基础上，如果存在非主属性对候选键的部分函数依赖则当前规范化程度最高只能达到 1NF，如果已经消除了非主属性对候选键的部分函数依赖（候选键只有单属性则必定不存在对候选键的部分函数依赖），则当前规范化程度至少满足 2NF；

在 2NF 基础上，如果存在非主属性对候选键的传递函数依赖则当前规范化程度最高只能达到 2NF，如果已经消除了非主属性对候选键的传递函数依赖，则当前规范化程度至少满足 3NF（如果没有非主属性，则至少满足 3NF）；

在 3NF 基础上，如果函数依赖集合中的所有函数依赖，都满足其左侧决定因素包含候选键，则当前规范化程度至少满足 BCNF，否则最高只能达到 3NF。

第 18 问 数据库中分布式透明包括哪些内容？

答：（1）分片透明：是指用户不必关心数据是如何分片的，它们对数据的操作在全局关系上进行，即如何分片对用户是透明的。

（2）复制透明：用户不用关心数据库在网络中各个节点的复制情况，被复制的数据的更新都由系统自动完成。

（3）位置透明：是指用户不必知道所操作的数据放在何处，即数据分配到哪个或哪些站点存储对用户是透明的。

（4）局部映像透明性（逻辑透明）：是最低层次的透明性，该透明性提供数据到局部数据库的映像，即用户不必关心局部 DBMS 支持哪种数据模型、使用哪种数据操纵语言，数据模型和操纵语言的转换是由系统完成的。因此，局部映像透明性对异构型和同构异质的分布式数据库系统是非常重要的。

第 19 问 什么是数字证书？什么是数字签名？

答：（1）数字证书是由权威机构——CA 证书授权（Certificate Authority）中心发行的，能提供在 Internet 上进行身份验证的一种权威性电子文档，人们可以在因特网交往中用它来证明自己的身份和识别对方的身份。数字证书包含版本、序列号、签名算法标识符、签发人姓名、有效期、主体名和主体公钥信息等并附有 CA 的签名，用户获取网站的数字证书后通过 CA 的公钥验证 CA 的签名，从而确认数字证书的有效性，然后验证网站的真伪。

（2）数字签名技术是对非对称加密技术与信息摘要的综合应用。通常的做法是：先对正文产生信息摘要，之后使用发送者 A 的私钥对该信息摘要进行加密，这就完成了签名。当接收者 B 收到签了名的摘要以后，会对摘要使用发送者 A 的公钥进行解密（认证），若能认证，则表明该信息确实是由 A 发送的。这就是数字签名技术。

第 20 问 对称加密、非对称加密的各自特点、算法有哪些？

答：（1）对称加密技术：对称加密：Ke=Kd；

特点：加密强度不高，但效率高；密钥分发困难。

常见对称密钥（共享密钥）加密算法：DES、AES、 3DES(三重 DES)、 RC-5、IDEA 算法。

（2）非对称加密技术：Ke ≠ Kd；密钥必须成对使用（公钥加密，相应的私钥解密）。

特点：加密速度慢，但强度高。

常见非对称密钥（公开密钥）加密算法： RSA、DSA、ECC。