数据库系统工程师易混淆知识点

CISC 与 RISC

●实现的基本思想

复杂指令集计算机（CISC）的基本思想：

进一步增强原有指令的功能，用更为复杂的新指令取代原先由软件子程序完成的功能，实现软件功能的硬化，导致机器的指令系统越来越庞大而复杂。

精简指令集计算机（RISC）的基本思想：

通过减少指令总数和简化指令功能，降低硬件设计的复杂度，使指令能单周期执行，并通过优化编译，提高指令的执行速度，采用硬线控制逻辑，优化编译程序。

计算机相关的周期概念

●指令周期（InstructionCycle）

取出并执行一条指令的时间。

●总线周期（BUSCycle）

也就是一个访问存储器或 I/O 端口操作所用的时间。

●时钟周期（ClockCycle）

又称震荡周期，是处理操作的最基本单位。（晶振频率的倒数）

●指令周期、总线周期和时钟周期之间的关系

一个指令周期由若干个总线周期组成，而一个总线周期时间又包含有若干个时钟周期。

一个总线周期包含一个（只有取址周期）或多个机器周期

CPU 响应 DMA 是在一个机器周期结束时

●机器周期

完成一个基本操作的时间单元，如取指周期、取数周期

输入输出

●内存与接口地址独立编制方式

内存地址和接口地址完全独立的两个地址空间，它们是完全独立的并且是相互隔离的。访问数据时所使用的指令也完全不同，用于接口的指令只用于接口读写，其余的指令全都是用于内存的。因此，在编程序或读程序中很容易使用和辨认

●内存与接口统一编址方法

内存地址和接口地址统一在一个公共的地址空间里，即内存单元和接口共用地址空间。在这些地址空间里划出一部分地址分配给接口使用，其余地址归内存单元使用。分配给内存的地址区间只能用于内存单元，接口绝不允许使用。同样，分配给接口的地区间内存单元也绝不能再用。这种编址方法的优点是原则上用于内存的指令全部都可以用于接口，其中一部分分配给接口使用，剩余的为内存所用，这经常会导致内存地址不连续。当用于内存的指令和用于接口的指令是完全一样的，维护程序时就需根据参数定义表仔细辨认。

寻址方式

●立即寻址

是一种特殊的寻址方式，指令中在操作码字段后面的部分不是通常意义上的操作数地址，而是操作数本身，也就是说数据就包含在指令中，只要取出指令，也就取出了可以立即使用的操作数。

●直接寻址

在直接寻址中，指令中地址码字段给出的地址 A 就是操作数的有效地址，即形式地址等于有效地址。

●间接寻址

间接寻址意味着指令中给出的地址 A 不是操作数的地址，而是存放操作数地址的主存单元的地址，简称操作数地址的地址。

●寄存器寻址

寄存器寻址指令的地址码部分给出了某一个通用寄存器的编号 Ri，这个指定的寄存器中存放着操作数。

●寄存器间接寻址

在寄存器间接寻址方式中，寄存器内存放的是操作数的地址，而不是操作数本身，即操作数是通过寄存器间接得到的。

●变址寻址

变址寻址就是把变址寄存器 Rx 的内容与指令中给出的形式地址 A 相加，形成操作数有效地址，即 EA=（Rx）+A。

●基址寻址

基址寻址是将基址寄存器 Rb 的内容与指令中给出的位移量 D 相加，形成操作数有效地址，即 EA=(Rb)+D。

●相对寻址

相对寻址是基址寻址的一种变通，由程序计数器提供基准地址，指令中的地址码字段作为位移量 D，两者相加后得到操作数的有效地址，即 EA=（PC）+D。

校验码

●奇偶校验

是一种简单有效的校验方法

通过在编码中增加一位校验位来使编码中的 1 的个数为奇数（奇校验）或者为偶数（偶校验），从而使码距变为 2

●CRC

利用生成多项式为 K 个数据位产生 r 个校验位来进行编码

其编码长度为：k+r

●海明码

在数据位之间插入 K 个校验位，通过扩大码距来实现检查和纠错；

设数据位是 n 位，校验位是 k 位，则 n 和 k 必须满足 以下关系：

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

进程控制相关概念

●同步/互斥

同步：

是合作进程间的直接制约问题。

互斥：

是申请临界资源进程间的间接制约问题。

●PV 操作

是实现进程同步和互斥的常用方法，P 操作和 V 操作是低级通信原语，在执行期间不可分割；其中 P 操

作表示申请一个资源，V 操作表示释放一个资源。

P 操作的定义：

S：=S-1，若 S>=0，则执行 P 操作的进程继续执行；若 S<0，则将该进程设为阻塞状态（因为无可用资源），并将其插入阻塞队列。

V 操作的定义：

S：=S+1，若 S>0，则执行 V 操作的进程继续执行；若 S<=0，则从阻塞状态唤醒一个进程，并将其插入就绪队列，然后执行 V 操作的进程继续。

磁盘容量

非格式化容量=面数×（磁道数/面）×内圆周长×最大位密度

格式化容量=面数×（磁道数/面）×（扇区数/道）×（字节数/扇区）

解释程序/编译程序

●解释程序

也称解释器；直接解释执行源程序，或者将源程序翻译成某种中间代码后再加以执行。

●编译程序

也称编译器；将源程序翻译成目标语言程序，然后在计算机上运行目标程序。

●两者的根本区别

编译方式下，机器上运行的是与源程序等价的目标程序，源程序和编译程序都不再参与目标程序的执行过程。

解释方式下，解释程序和源程序（或某种等价表示）要参与到程序的运行过程中，运行程序的控制权在

解释程序。

即：解释方式，翻译程序不生成独立的目标程序，而编译方式则生成独立保持的目标程序。

编译器工作的过程

●词法分析阶段

是编译过程的第一阶段，其任务是对源程序从前到后（从左到右）逐个字符扫描，从中识别出一个个“单词”符号。

词法分析过程的依据是语言的词法规则，即描述“单词”结构的规则。

●语法分析阶段

其任务是在词法分析的基础上，根据语言的语法规则将单词符号序列分解成各类语法单位。

通常语法分析是确定整个输入串是否构成一个语法上正确的程序。

一般来说，通过编译的程序，不存在语法上的错误。

●语义分析阶段

其任务主要检查源程序是否包含静态语义错误，并收集类型信息供后面的代码生成阶段使用。

语义分析的一个主要工作是进行类型分析和检查。

●中间代码生成

其任务是根据语义分析的输出生成中间代码。

●目标代码生成

是编译器工作的最后一个阶段。其任务是把中间代码变换成特定机器上的绝对指令代码、可重定位的指令代码或汇编指令代码。本阶段与具体机器密切相关。

●符号表管理

符号表的作用是记录源程序中各个符号的必要信息，以辅助语义的正确性检查和代码生成，在编译过程中需要对符号表进行快速有效地查找、插入、修改和删除等操作。

形参/实参

●形参

在过程（或函数）首部声明的参数成为形式参数，简称形参

●实参

过程（或函数）调用时的参数称为实际参数，简称实参。

传值/传址（引用）调用

●传值调用

形参取的是实参的值，形参的改变不会导致调用点所传的实参的值发生改变。

●引用（传址）调用

形参取的是实参的地址，即相当于实参存储单元的地址引用，因此其值的改变同时就改变了实参的值。

传值/传址（引用）调用

●传值调用

形参取的是实参的值，形参的改变不会导致调用点所传的实参的值发生改变。

●引用（传址）调用

形参取的是实参的地址，即相当于实参存储单元的地址引用，因此其值的改变同时就改变了实参的值。

二叉树的遍历

●前序遍历

先访问根结点，再依次按前序遍历的方式访问根结点的左子树、右子树。

●中序遍历

先中序遍历根结点的左子树，再访问根结点，再中序遍历根结点的右子树。

●后序遍历

先中序遍历根结点的左子树，再中序遍历根结点的右子树，再访问根结点。

●层次遍历

先访问第一层的根结点，然后从左到右依次访问第二层上的所有结点，再以同样的方式访问下一层，直到访问到树中最低层的所有结点。

TCP 与 UDP 的区别

●TCP

面向连接；建立连接必须经过 3 次握手；连接断开：4 次断开；传输可靠性：可靠；适合传输大量数据

●UDP

无连接；传输可靠性：不可靠；适合传输少量数据；通信开销小

网络攻击的相关概念

●冒充

就是一个实体假装成一个不同的实体。常与主动攻击形式一起使用，特别是消息的重演与篡改。

●重演

当一个消息或部分消息为了产生非授权效果而被重复时，出现重演。

●消息篡改

数据所传送的内容被改变而未被发觉，并导致非授权后果。

●服务拒绝

当一个实体不能执行它的正常功能，或它的动作妨碍了别的实体执行它们的正常功能的时候，便发生服务拒绝。

数据流图/数据字典

●数据流图（Data Flow Diagram，DFD）

是一种最常用的结构化分析工具，从数据传递和加工的角度，以图形的方式刻画系统内数据的运动情况。

是一种能全面地描述信息系统逻辑模型的主要工具，可以用少数几种符号综合地反映出信息在系统中的流动、处理和存储的情况。

●数据字典

对数据流图的重要补充和说明。

是以特定格式记录下来的、对系统的数据流图中各个基本要素（数据流、处理逻辑、数据存储和外部实体）的内容和特征所做的完整的定义和说明。

聚合&耦合

聚合衡量模块内部各元素结合的紧密程度

●偶然聚合

模块完成的动作之间没有任何关系，或者仅仅是一种非常松散的关系。

●逻辑聚合

模块内部的各个组成在逻辑上具有相似的处理动作，但功能用途上彼此无关。

●时间聚合

模块内部的各个组成部分所包含的处理动作必须在同一时间内执行。

●过程聚合

模块内部各个组成部分所要完成的动作虽然没有关系，但必须按特定的次序执行。

●通信聚合

模块的各个组成部分所完成的动作都使用了同一个数据或产生同一输出数据。

●顺序聚合

模块内部的各个部分，前一部分处理动作的最后输出是后一部分处理动作的输入。

●功能聚合

模块内部各个部分全部属于一个整体，并执行同一功能，且各部分对实现该功能都必不可少

耦合度量不同模块间互相依赖的程度

●非直接耦合

两个模块之间没有直接关系，它们的联系完全是通过主模块的控制和调用来实现的。

●数据耦合

两个模块彼此间通过数据参数交换信息。

●标记耦合

一组模块通过参数表传递记录信息，这个记录是某一个数据结构的子结构，而不是简单变量。

●控制耦合

两个模块彼此间传递的信息中有控制信息。

●外部耦合

一组模块都访问同一全局简单变量而不是同一全局数据结构，而且不是通过参数表传递该全局变量的信息。

●公共耦合

两个模块之间通过一个公共的数据区域传递信息。

●内容耦合

一个模块需要涉及到另一个模块的内部信息。

【制作于 2023 年 11 月 适用于第 4 版教材】