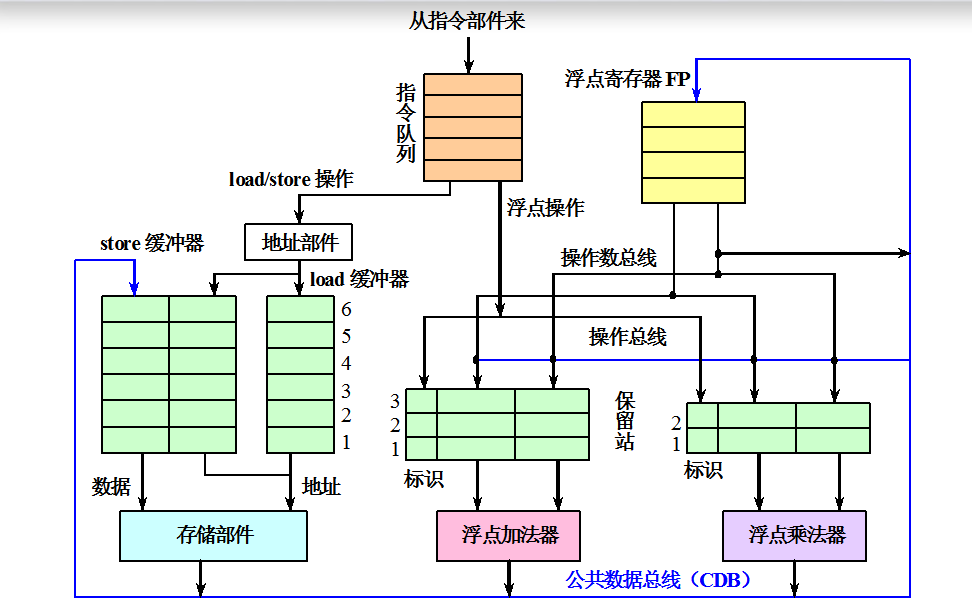
4.1

1. 指令级并行（ILP）：几乎所有的处理机都利用流水线来使指令重叠并行执行，以达到提高性能的目的。这种指令之间存在的潜在并行性称为指令级并行。
2. 两类开发ILP的方法：基于硬件的动态开发以及基于软件的静态开发。
3. 流水线处理机的实际CPI： CPI流水线 = CPI理想 + 停顿结构冲突 + 停顿数据冲突 + 停顿控制冲突
4. 理想CPI是衡量流水线最高性能的一个指标。
5. IPC：每个时钟周期完成的指令条数
6. 基本程序块：一段除了入口和出口以外不包含其他分支指令和转入点的线性代码段。
7. 循环级并行：使一个循环中的不同循环体并行执行。（每一次循环都可以与其他的循环重叠并行执行；但在每一次循环的内部，却没有任何的并行性）
8. 最基本的开发循环级并行的技术：循环展开技术、采用向量指令和向量数据表示。
9. 相关有三种类型：数据相关、名相关、控制相关
10. 流水线冲突：对于具体的流水线来说，由于相关的存在，使得指令流中的下一条指令不能在指定的时钟周期执行。
11. 流水线冲突有三种类型：结构冲突（硬件资源冲突）、数据冲突（数据相关和名相关）、控制冲突（控制相关）
12. 解决相关问题方法：保持相关，但避免发生冲突如指令调度）； 通过代码变换，消除相关
13. 程序顺序：由源程序确定的在完全串行方式下指令的执行顺序。（各种技术目标是尽可能开发并行线，只有在可能会导致错误的情况下，才保持程序顺序）
14. 对于正确地执行程序来说，必须保持的最关键的两个属性是：数据流（数据值从其产生者指令到其消费者指令的实际流动）和异常行为。控制相关并不是一个必须严格保持的关键属性。
15. 保持异常行为是指：无论怎么改变指令的执行顺序，都不能改变程序中异常的发生情况。即原来程序中是怎么发生的，改变执行顺序后还是怎么发生。弱化为：指令执行顺序的改变不能导致程序中发生新的异常
16. 如果我们能做到保持程序的数据相关和控制相关，就能保持程序的数据流和异常行为。

4.2

1. 静态调度：依靠编译器对代码进行静态调度，以减少相关和冲突。它不是在程序执行的过程中、而是在编译期间进行代码调度和优化。通过把相关的指令拉开距离来减少可能产生的停顿。
2. 动态调度：在程序的执行过程中，依靠专门硬件对代码进行调度，减少数据相关导致的停顿。
3. 优点：能够处理一些在编译时情况不明的相关（比如涉及到存储器访问的相关），并简化了编译器；能够使本来是面向某一流水线优化编译的代码在其他的流水线（动态调度）上也能高效地执行。
4. 缺点：以硬件复杂性的显著增加为代价
5. 我们所使用流水线的最大的局限性:指令必须按序流出和执行
6. 在之前的基本流水线中，译码（ID）检测结构冲突和数据冲突，一旦一条指令受阻，其后的指令都将停顿。解决方法为：乱序执行
7. 为了允许乱序执行，我们将5段流水线的译码阶段再分为两个阶段：流出（Issue，IS）：指令译码，检查是否存在结构冲突；读操作数（RO）：等待数据冲突消失，然后读操作数。
8. 乱序执行可能导致：WAR WAW冲突
9. 指令乱序完成带来的最大问题:异常处理比较复杂
10. 动态调度另一问题：不精确异常：当执行指令i导致发生异常时，处理机的现场（状态）与严格按程序顺序执行时指令i的现场不同。
11. Tomasulo算法基本思想：记录和检测指令相关，操作数一旦就绪就立即执行，把发生RAW冲突的可能性减少到最小；通过寄存器换名来消除WAR冲突和WAW冲突。
12. P113图



13.保留站：每个保留站中保存一条已经流出并等待到本功能部件执行的指令（相关信息）。包括：操作码、操作数以及用于检测和解决冲突的信息

14.保留站设置在运算部件入口。

15.公共数据总线CDB：所有功能部件的计算结果都是送到CDB上，由它把这些结果直接送到（播送到）各个需要该结果的地方。

在具有多个执行部件且采用多流出（即每个时钟周期流出多条指令）的流水线中，需要采用多条CDB。

16.load缓冲器和store缓冲器：存放读/写存储器的数据或地址

17.load缓冲器的作用有3个：

存放用于计算有效地址的分量；

记录正在进行的load访存，等待存储器的响应；

保存已经完成了的load的结果（即从存储器取来的数据），等待CDB传输。

18.store缓冲器的作用有3个：

存放用于计算有效地址的分量；

保存正在进行的store访存的目标地址，该store正在等待存储数据的到达；

保存该store的地址和数据，直到存储部件接收。

1. 浮点寄存器FP：它们通过一对总线连接到功能部件，并通过CDB连接到store缓冲器。
2. 指令队列：指令部件送来的指令放入指令队列；指令队列中的指令按先进先出的顺序流出
3. Tomasulo算法中，寄存器换名是通过保留站和流出逻辑来共同完成的：

当指令流出时，如果其操作数还没有计算出来，则将该指令中相应的寄存器号换名为将产生这个操作数的保留站的标识。

指令流出到保留站后，其操作数寄存器号或者换成了数据本身（如果该数据已经就绪），或者换成了保留站的标识，不再与寄存器有关系。

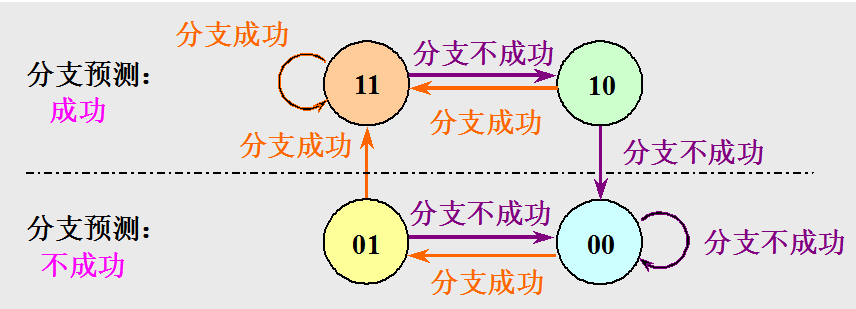
1. Tomasulo算法采用分布的保留站，具有以下两个特点：

冲突检测和指令执行控制是分布的。

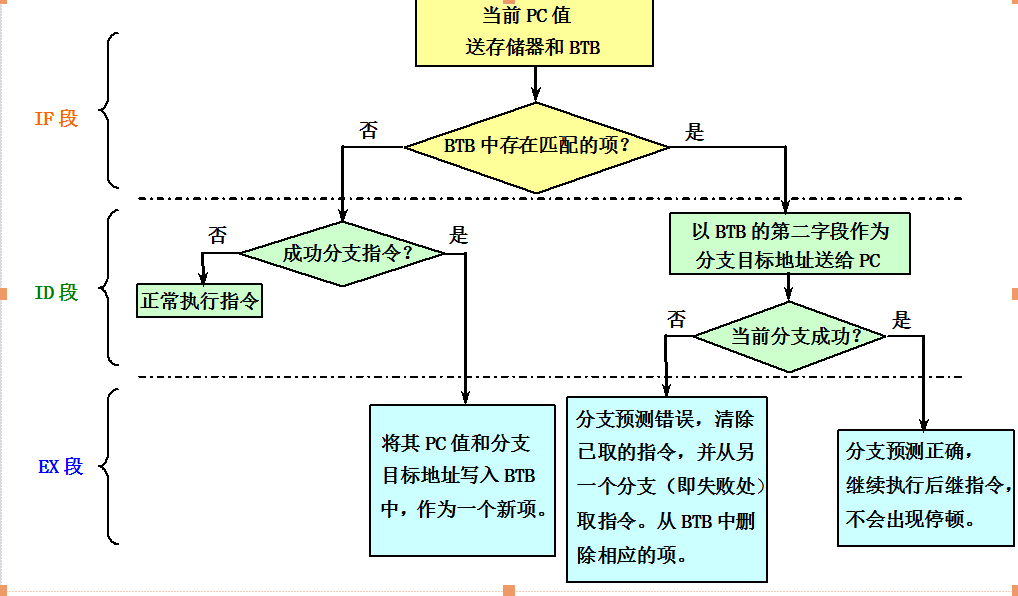
计算结果通过CDB直接从产生它的保留站传送到所有需要它的功能部件，而不用经过寄存器。

4.3

1. 所开发的ILP越多，控制相关的制约就越大，分支预测就要有更高的准确度。
2. 动态分支预测：在程序运行时，根据分支指令过去的表现来预测其将来的行为。如果分支行为发生了变化，预测结果也跟着改变。有更好的预测准确度和适应性。
3. 最简单的动态分支预测方法：分支历史表BHT。用BHT来记录分支指令最近一次或几次的执行情况（成功或不成功），并据此进行预测。
4. 采用两位分支预测位的BHT状态转换



1. 两位分支预测中的操作有两个步骤：分支预测。状态修改。
2. BHT方法只在以下情况下才有用：判定分支是否成功所需的时间大于确定分支目标地址所需的时间。
3. 分支目标缓冲器BTB：将分支成功的分支指令的地址和它的分支目标地址都放到一个缓冲区中保存起来，缓冲区以分支指令的地址作为标识。这个缓冲区就是分支目标缓冲器（Branch-Target Buffer，简记为BTB。在指令IF段就可得到信息，将分支开销降到0
4. 采用BTB时所进行的处理步骤





1. BTB的另一种形式：在分支目标缓冲器中存放一条或者多条分支目标处的指令。

有三个潜在的好处：

更快地获得分支目标处的指令；

可以一次提供分支目标处的多条指令，这对于多流出处理器是很有必要的；

使我们可以进行称为分支折叠（branch folding）的优化。

1. 前瞻执行的基本思想：对分支指令的结果进行猜测，并假设这个猜测总是对的，然后按这个猜测结果继续取、流出和执行后续的指令。只是执行指令的结果不是写回到寄存器或存储器，而是放到一个称为ROB的缓冲器中。等到相应的指令得到“确认”（即确实是应该执行的）之后，才将结果写入寄存器或存储器
2. 基于硬件的前瞻执行结合了三种思想：

动态分支预测。用来选择后续执行的指令。

在控制相关的结果尚未出来之前，前瞻地执行后续指令。

用动态调度对基本块的各种组合进行跨基本块的调度。

1. 对Tomasulo算法加以扩充，就可以支持前瞻执行。把Tomasulo算法的写结果和指令完成加以区分，分成两个不同的段：写结果，指令确认
2. 写结果段

把前瞻执行的结果写到ROB中；

通过CDB在指令之间传送结果，供需要用到这些结果的指令使用。

指令确认段

在分支指令的结果出来后，对相应指令的前瞻执行给予确认。

如果前面所做的猜测是对的，把在ROB中的结果写到寄存器或存储器。

如果发现前面对分支结果的猜测是错误的，那就不予以确认，并从那条分支指令的另一条路径开始重新执行。

14.实现前瞻的关键思想：允许指令乱序执行，但必须顺序确认。

前瞻执行通过ROB实现了指令的顺序完成。

能够实现精确异常。

很容易地推广到整数寄存器和整数功能单元上。

主要缺点：所需的硬件太复杂。

4.4

1. 多流出处理机有两种基本风格：超标量、超长指令字VLIW
2. 超标量：在每个时钟周期流出的指令条数不固定，依代码的具体情况而定。（有上限）

设这个上限为n，就称该处理机为n流出。可以通过编译器进行静态调度，也可以基于Tomasulo算法进行动态调度。

1. 超长指令字：在每个时钟周期流出的指令条数是固定的，这些指令构成一条长指令或者一个指令包。指令包中，指令之间的并行性是通过指令显式地表示出来的。指令调度是由编译器静态完成的。
2. 基于静态调度的多流出技术：指令按序流出，在流出时进行冲突检测
3. 流出部件检测结构冲突或者数据冲突。一般分两阶段实现：

第一阶段：进行流出包内的冲突检测，选出初步判定可以流出的指令。

第二阶段：检测所选出的指令与正在执行的指令是否有冲突。

1. 静态调度的多流出技术受到load指令和分支延迟限制
2. 动态调度的多流出技术：指令按顺序流向保留站；将整数所用的表结构与浮点用的表结构分离开，分别进行处理，这样就可以同时地流出一条浮点指令和一条整数指令到各自的保留站。
3. 两种不同的方法可以实现多流出：在半个时钟周期里完成流出步骤，这样一个时钟周期就能处理两条指令；设置一次能同时处理两条指令的逻辑电路。
4. 多流出处理器受到的限制：

程序所固有的指令级并行性。

硬件实现上的困难。

超标量和超长指令字处理器固有的技术限制

1. 超流水线处理机：将每个流水段进一步细分，这样在一个时钟周期内能够分时流出多条指令
2. 对于一台每个时钟周期能流出n条指令的超流水线计算机来说，这n条指令不是同时流出的，而是每隔1/n个时钟周期流出一条指令。

4.5

1. 编译器完成指令调度能力受限于两个特性：程序固有的指令级并行性；流水线功能部件的执行延迟。
2. 循环展开：把循环体的代码复制多次并按顺序排放，然后相应调整循环的结束条件。

5.1

1.程序访问的局部性原理：对于绝大多数程序来说，程序所访问的指令和数据在地址上不是均匀分布的，而是相对簇聚的。包括时间局部性和空间局部性。

2.时间局部性：是指程序将要用到的信息很可能就是现在正在使用的信息。

3.空间局部性：是指程序将要用到的信息很可能与现在正在使用的信息在存储空间上是相邻的。

4.存储层次的性能参数：S ── 容量

TA ── 访问时间

C ── 每位价格

5.存储系统的多级层次结构：从CPU来看，该存储系统的速度接近于M1，而容量和每位价格都接近于Mn。存储器越靠近CPU，访问时间越短，容量越小，价格越高。

6.存储系统的平均每位价格C：两级存储层次结构

当S1<<S2,C约等于C2

1. 命中率H：CPU访问存储系统时，在M1中找到所需信息的概率

N1 ── 访问M1的次数

N2 ── 访问M2的次数

8.失效率 ：F＝1－H

9.平均访问时间TA：分两种情况来考虑CPU的一次访存：

当命中时，访问时间即为TA1（命中时间）

当不命中时，情况比较复杂。

不命中时的访问时间为：TA2＋TB＋TA1＝TA1＋TM

TM ＝TA2＋TB

失效开销TM：从向M2发出访问请求到把整个数据块调入M1中所需的时间。

TB：传送一个信息块所需的时间为。

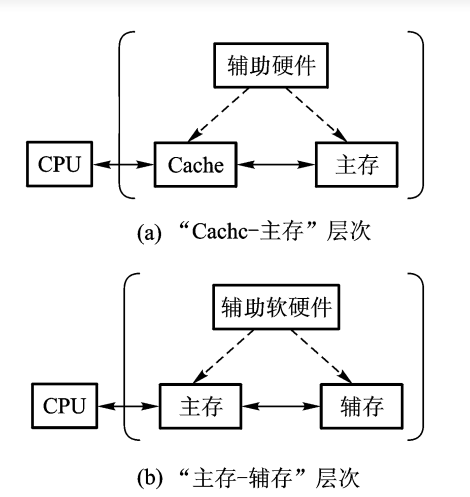
考虑到命中和不命中的概率分布是H和1-H

TA ＝ HTA1＋（1－H）（TA1＋TM）＝ TA1＋（1－H）TM

或 TA＝ TA1＋FTM

1. 三级存储系统：“Cache－主存”和“主存－辅存”层次

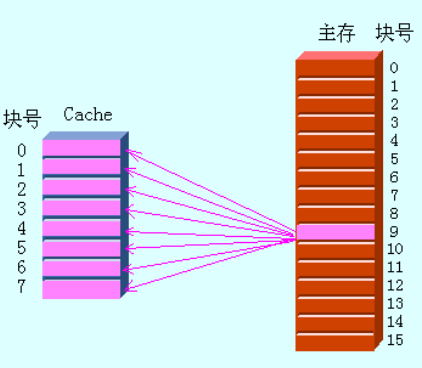
“Cache－主存”层次：弥补主存速度的不足

“主存－辅存”层次： 弥补主存容量的不足

5.2

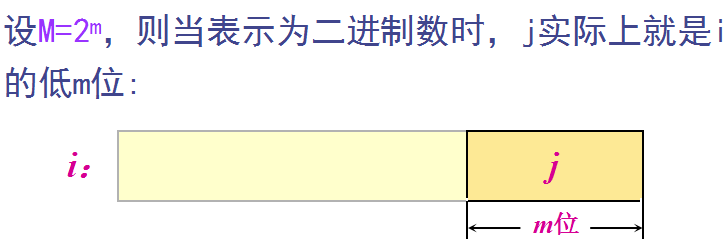
1. Cache:Cache和主存均被分割成大小相同的块，信息以块单位调入Cache。相应的，CPU的访存地址被分割成两部分：块地址和块内位移。
2. 主存块地址用于查找该块在Cache中的位置，块内位移用于确定所访问的数据在该块中的位置。
3. 映像规则：全相联映像、直接映像、组相联映像。

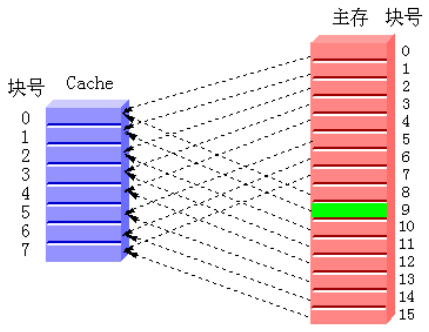
全相联映像：



直接映像：

对于主存的第i 块，若它映象到Cache的第j 块，则j＝i mod (M )

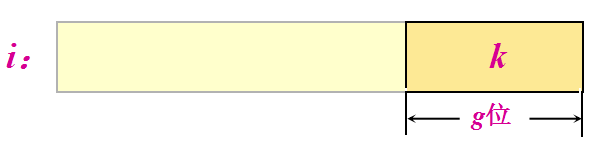


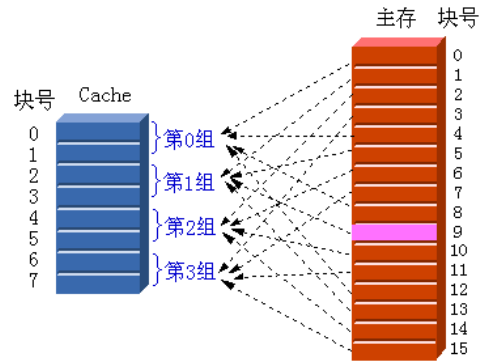


组相联映像：

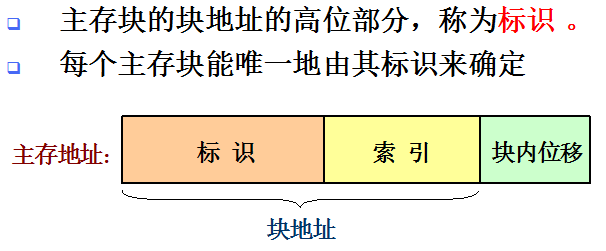
若主存第i 块映象到第k 组，则k＝i mod（G） （G为Cache的组数）

设G＝2g，则当表示为二进制数时，k 实际上就是i 的低 g 位:





1. n 路组相联：每组中有n个块(n＝M/G )。n 称为相联度。相联度越高，Cache空间的利用率就越高，块冲突概率就越低，失效率也就越低。
2. 查找方法：并行查找和顺序查找。通过查找目录表实现。每一个Cache块在该表中都有唯一的一项，用于指出当前该块中存放的信息是哪个主存块的。目录表存的就是主存地址的标识。



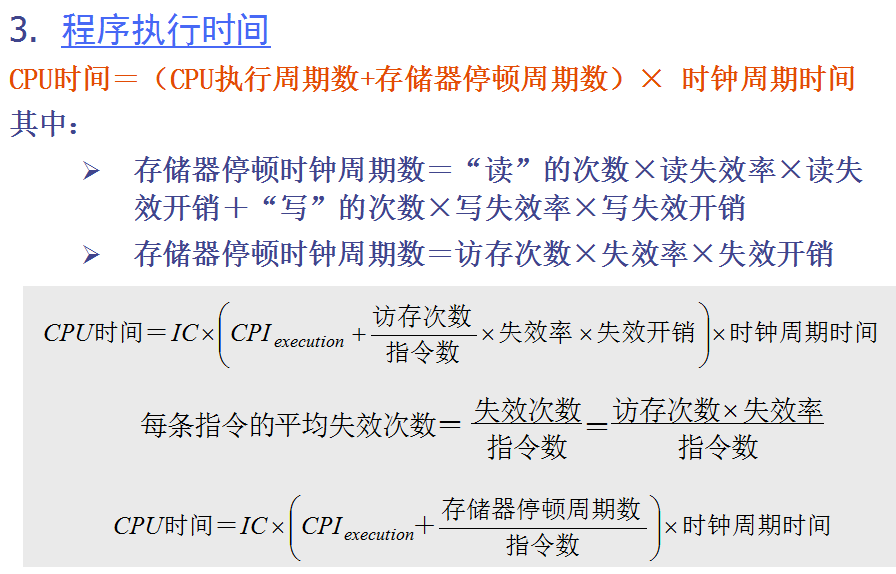
1. 候选位置：一个主存块可能映像到Cache中的一个或多个Cache块位置。
2. 并行查找实现方法：相联存储器；单体多字存储器＋比较器

**并行查找详细内容没看**

1. 替换算法：Cache块位置被占用，替换哪一个。三种算法：随机法、先进先出法、最近最少使用法。对于大容量Cache LRU和随机法的命中率差别不大
2. 两种写策略：写直达法、写回法
3. 写回法的优点：速度快，所使用的存储器带宽较低。按写分配
4. 写直达法的优点：易于实现，一致性好。不按写分配
5. Cache性能分析：平均访存时间 ＝ 命中时间＋失效率×失效开销

平均访存时间　＝　　指令所占的百分比×（指令命中时间＋指令失效率×失效开销） ＋ 数据所占的百分比×（数据命中时间＋数据失效率×失效开销）

13.



14.可以从三个方面改进Cache的性能：

降低失效率

减少失效开销

减少Cache命中时间

5.3

1.3种类型的不命中：强制性不命中，容量不命中，冲突不命中

2.相联度越高，冲突失效就越少；

强制性失效和容量失效不受相联度的影响；

强制性失效不受Cache容量的影响，但容量失效却随着容量的增加而减少；

3.减少三种失效的方法

强制性失效：增加块大小，预取

容量失效：增加容量

冲突失效：提高相联度

1. 增加Cache块大小：对于给定的Cache容量，当块大小增加时，失效率开始是下降，后来反而上升了

原因：

一方面它减少了强制性失效；

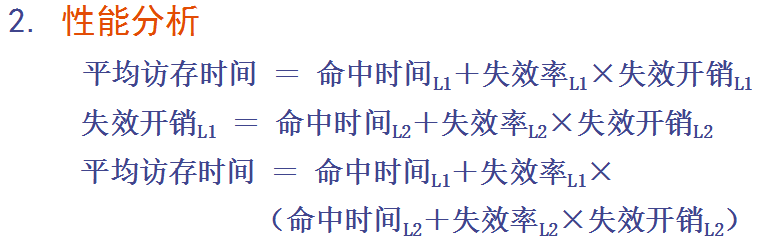
另一方面，由于增加块大小会减少Cache中块的数目，所以有可能会增加冲突失效。

1. 伪相联Cache:在逻辑上把直接映象Cache的空间上下平分为两个区。对于任何一次访问，伪相联Cache先按直接映象Cache的方式去处理。若命中，则其访问过程与直接映象Cache的情况一样。若不命中，则再到另一区相应的位置去查找。若找到，则发生了伪命中，否则就只好访问下一级存储器。
2. 硬件预取 ：指令和数据都可以预取。预取内容既可放入Cache，也可放在外缓冲器中。指令预取通常由Cache之外的硬件完成
3. 编译器控制的预取：在编译时加入预取指令，在数据被用到之前发出预取请求
4. 编译优化：对软件进行优化来降低不命中率。在编译时，对程序中的指令和数据进行重新

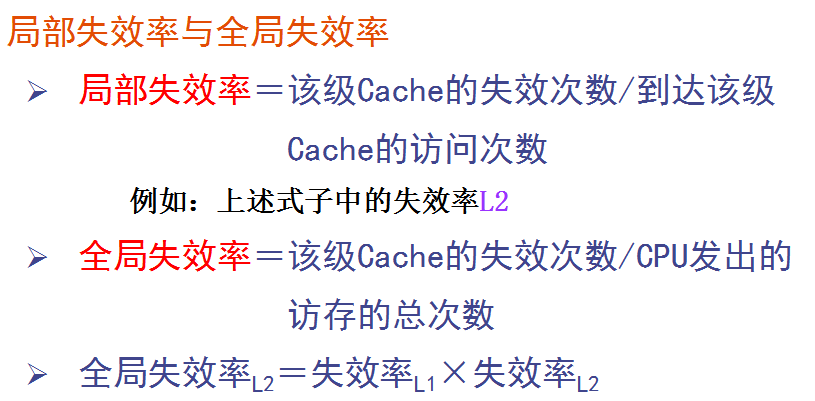
组织，以降低Cache失效率。

5.4

1. 减少Cache失效开销5种方法：采用两级Cache、让读不命中优先于写、写缓冲合并、请求字处理技术、非阻塞Cache技术
2. 采用两级Cache:第二级Cache不会影响CPU的时钟频率，在第二级Cache比第一级 Cache大得多的情况下，两级Cache的全局失效率和容量与第二级Cache相同的单级Cache的失效率非常接近。局部失效率不是衡量第二级Cache的一个好指标，因此，在评价第二级Cache时，应用全局失效率这个指标。
3. 二级Cache



12.评价第二级Cache时，应使用全局失效率这个指标



13.第二级Cache的参数:容量第二级Cache的容量一般比第一级的大许多。

相联度:第二级Cache可采用较高的相联度或伪相联方法

1. 让读失效优先于写:推迟对读不命中的处理，直至写缓冲器清空
2. 写缓冲合并：提高写缓冲器的效率
3. 如果写缓冲器为空，就把数据和相应地址写入该缓冲器。从CPU的角度来看，该写操作就算完成了。

如果写缓冲器中已经有了待写入的数据，就要把这次的写入地址与写缓冲器中已有的所有地址进行比较，看是否有匹配的项。如果有地址匹配而对应的位置又是空闲的，就把这次要写入的数据与该项合并。这就叫写缓冲合并。

如果写缓冲器满且又没有能进行写合并的项，就必须等待。

16.请求字处理：请求字，从下一级存储器调入Cache的块中，只有一个字是立即需要的。这个字称为请求字

17.两种方案：尽早重启动：调块时，从块的起始位置开始读起。一旦请求字到达，就立即发送给CPU，让CPU继续执行。

请求字优先：调块时，从请求字所在的位置读起。这样，第一个读出的字便是请求字。将之立即发送给CPU。

1. 这种技术在以下情况下效果不大：Cache块较小；下一条指令正好访问同一Cache块的另一部分
2. 非阻塞Cache技术：Cache失效时仍允许CPU进行其他的命中访问。即允许“失效下命中”

**5.5**

1. 减少命中时间：命中时间直接影响到处理器的时钟频率
2. 容量小、结构简单的Cache:硬件越简单，速度就越快。使Cache足够小以便可以与CPU放在同一块芯片上。
3. 虚拟Cache：访问Cache的索引以及Cache中的标识都是虚拟地址。
4. 虚拟Cache优点：在不命中时不需要地址转换，省去了地址转换时间。即使不命中，地址转换和访问Cache也是并行进行，速度比物理Cache快很多。
5. 缺点：每当进行进程切换时需要清空Cache；操作系统和用户程序对于同一个物理地址可能采用两种以上不同形式的虚拟地址来访问，从而导致数据在虚拟Cache中有两个副本。
6. 虚拟索引＋物理标识优点：兼顾虚拟Cache和物理Cache的好处。局限性：Cache容量受到限制。Cache容量≤页大小×相联度
7. Cache访问流水化：把对第一级Cache的访问按流水方式组织，这就使访问Cache需要多个时钟周期才可以完成。
8. 踪迹Cache:普通指令Cache存放静态指令序列。踪迹Cache存放的是CPU所执行过的动态指令序列，其中包含了由分值预测展开了的指令，该分值预测是否正确需要在取到该分支指令时确认。

总结：



5.6

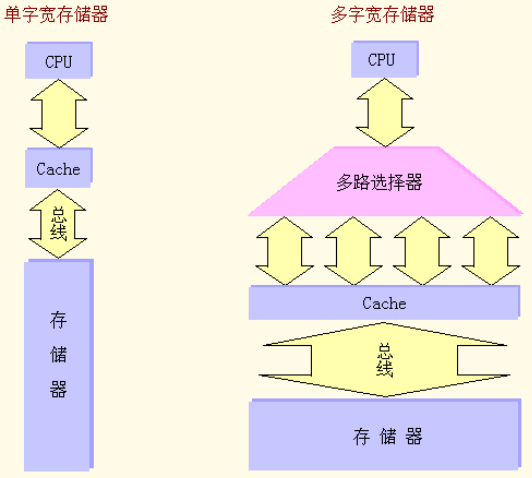
1.主存的主要性能指标：延迟和带宽

2.并行主存系统：在一个访存周期内能并行访问多个存储字的存储器，它能有效提高存储器的带宽。

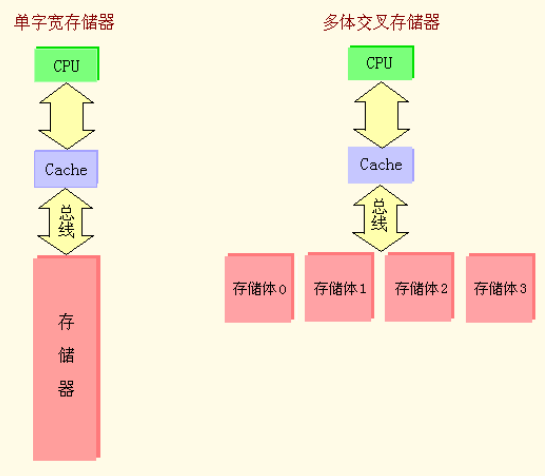
3.单体单字宽的存储器：字长与CPU相等，每次只能访问一个存储字。

4.单体单字宽的存储器带宽：Bm=W/Tm Tm是存储器访问周期 字长为W位

5.



1. 单体多字存储器：带宽变为原来的m倍Bm=m\*W/Tm m为每个存储周期读出m个CPU字。
2. 多体交叉存储器：



1. 高位交叉编址:对存储单元矩阵按列优先的方式进行编址
2. 低位交叉编织：对存储单元矩阵按行优先方式进行编址
3. 避免存储体冲突：体冲突，是指两个访问请求要访问同一个存储体。

5.7

1. 虚拟存储器：虚拟存储器是“主存－辅存”层次进一步发展的结果
2. 虚拟存储器可以分为两类：页式和段式。页式：虚拟存储器把空间划分为大小相同的块，称为页面。段式：把空间划分为可变长的块，称为段。
3. 快速地址转换技术：地址变换缓冲器TLB：TLB是一个专用的高速缓冲器，用于存放近期经常使用的页表项
4. 计算机指令集结构
5. 区别不同指令集结构的主要因素： CPU中用来存储操作数的存储单元的类型（堆栈、累加器、通用寄存器组）
6. 指令集结构可以分为： 堆栈结构、累加器结构、通用寄存器结构
7. 通用寄存器结构可以根据操作数来源不同分为： 寄存器-存储器结构、寄存器-寄存器结构
8. 寄存器结构优势：与其他CPU内部存储单元一样，寄存器的访问速度比存储器快；

对编译器而言，更加容易、有效分配和使用寄存器；

寄存器可以用来存放变量。

1. 根据ALU指令操作数的两个特征对通用寄存器结构进行进一步细分：一个特征是ALU指令的操作数个数；另一个是ALU指令中存储器操作数的个数
2. 寻址方式：是指一种指令集结构如何确定所要访问的数据地址。
3. 对指令集的基本要求：完整性、规整性、高效率和兼容性。
4. 完整性： 在一个有限可用的存储空间内，对于任何可解的问题编制计算程序时，指令集所提供的指令足够使用。
5. 规整性：对称性和均匀性。对称性是指所有与指令集有关的存储单元的使用、操作码的设置都是对称的。均匀性是指对于各种不同的操作数类型、字长、操作种类和数据存储单元，指令的设置都要同等对待。
6. 高效率： 指令的执行速度快、使用频率高。
7. CISC：复杂指令集计算机 RISC：精简指令集计算机
8. 指令由两部分组成：操作码和地址码
9. 指令格式的设计：确定字的编码方式，包括操作码字段和地址码字段的编码和表示方式
10. 两种寻址方式：把寻址方式与操作码一起编码；设置专门的地址描述符
11. 指令集的3种编码格式：可变长度编码格式；固定长度编码格式；混合型编码格式。
12. 数据表示：指计算机硬件能够直接识别、指令集可以直接调用的数据类型
13. 数据结构：由软件进行处理和实现的各种数据类型。
14. 输入/输出系统
15. 输入/输出系统简称I/O系统，包括：输入/输出设备以及输入/输出设备与处理机的连接
16. 照主要完成的工作进行分类：存储I/O系统、通信I/O系统
17. 系统的响应时间：从用户输入命令开始，到得到结果所花费的时间。由两部分构成：I/O系统的响应时间；CPU的处理时间
18. 评价I/O系统性能的参数主要有：连接特性（哪些I/O设备可以和计算机系统相连接）I/O系统的容量（I/O系统可以容纳的I/O设备数）响应时间和吞吐率等
19. 另一种衡量I/O系统性能的方法：考虑I/O操作对CPU的打扰情况。即考查某个进程在执行时，由于其他进程的I/O操作，使得该进程的执行时间增加了多少。

6,2

1. 反映外设可靠性能的参数有：可靠性（Reliability）可用性（Availability）可信性（Dependability）
2. 系统的可靠性：系统从某个初始参考点开始一直连续提供服务的能力。常用平均失效前时间MTTF来衡量，倒数就是系统的失效率。如果系统中每个模块的生存期服从指数分布，则系统整体的失效率是各部件的失效率之和
3. 系统的可用性：系统正常工作的时间在连续两次正常服务间隔时间中所占的比率。



1. 系统的可信性：服务的质量。即在多大程度上可以合理地认为服务是可靠的。
2. 提高系统组成部件可靠性的方法

有效构建方法：在构建系统的过程中消除故障隐患，这样建立起来的系统就不会出现故障。

纠错方法：在系统构建中采用容错的方法。这样即使出现故障，也可以通过容错信息保证系统正常工作。

6.3

1. 磁盘阵列DA（Disk Array）：使用多个磁盘（包括驱动器）的组合来代替一个大容量的磁盘。

多个磁盘并行工作。

以条带为单位把数据均匀地分布到多个磁盘上。（交叉存放）

条带存放可以使多个数据读/写请求并行地被处理，从而提高总的I/O性能。

1. 并行线的两方面含义：多个独立的请求可以由多个盘来并行地处理，减少了I/O请求的排队等待时间 ；如果一个请求访问多个块，就可以由多个盘合作来并行处理。
2. 阵列中磁盘数量的增加会导致磁盘阵列可靠性的下降。解决方法：在磁盘阵列中设置冗余信息盘。当单个磁盘失效时，丢失的信息可以通过冗余盘中的信息重新构建。
3. 大多数磁盘阵列的组成可以由以下两个特征来区分：数据交叉存放的粒度、冗余数据的计算方法以及在磁盘阵列中的存放方式
4. 数据交叉存放的粒度（可以是细粒度的，也可以是粗粒度的）

细粒度磁盘阵列是在概念上把数据分割成相对较小的单位交叉存放。

优点：所有I/O请求都能够获得很高的数据传输率。

缺点：在任何时间，都只有一个逻辑上的I/O在处理当中，而且所有的磁盘都会因为为每个请求进行定位而浪费时间。

粗粒度磁盘阵列是把数据以相对较大的单位交叉存放。

多个较小规模的请求可以同时得到处理。

对于较大规模的请求又能获得较高的传输率。

1. 在磁盘阵列中设置冗余需要解决以下两个问题：

如何计算冗余信息?大多都是采用奇偶校验码；也有采用汉明码（Hamming code）或Reed-Solomon码的。

如何把冗余信息分布到磁盘阵列中的各个盘?有两种方法：

把冗余信息集中存放在少数的几个盘中。

把冗余信息均匀地存放到所有的盘中。

7.



1. RAID0：非冗余阵列，无冗余信息。严格地说，它不属于RAID系列。把数据切分成条带，以条带为单位交叉地分布存放到多个磁盘中。
2. RAID1：亦称镜像盘，使用双备份磁盘。每当把数据写入磁盘时，将该数据也写入其镜像盘，形成信息的两个副本。

特点：能实现快速的读取操作。

写性能由写性能最差的磁盘决定。相对以后各级RAID来说，RAID1的写速度较快。

可靠性很高，数据的恢复很简单。

最昂贵的解决方法，物理磁盘空间是逻辑磁盘空间的两倍。

10.RAID2：存储器式的磁盘阵列（按汉明纠错码的思路构建）

每个数据盘存放所有数据字的一位（位交叉存放）

各个数据盘上的相应位计算汉明校验码，编码位被存放在多个校验（ECC）磁盘的对应位上。

冗余盘是用来存放汉明码的，其个数为log2m级。m：数据盘的个数（也就是数据字的位数）

并未被广泛应用，目前还没有商业化产品。

11.RAID3：位交叉奇偶校验盘阵列

采用奇偶校验

写数据时，为每行数据形成奇偶校验位并写入校验盘

读出数据时，如果控制器发现某个磁盘出故障，就可以根据故障盘以外的所有其他盘中的正确信息恢复故障盘中的数据。（通过异或运算实现）

细粒度的磁盘阵列，即采用的条带宽度较小。（可以是1个字节或1位）

只需要一个校验盘，校验空间开销比较小。

12.RAID4：块交叉奇偶校验磁盘阵列。采用比较大的条带，以块为单位进行交叉存放和计算奇偶校验。

读取操作

每次只需访问数据所在的磁盘。

仅在该磁盘出现故障时，才会去读校验盘，并进行数据的重建。

写入操作

假定:有4个数据盘和一个冗余盘。写数据需要2次磁盘读和2次磁盘写操作。

13.RAID5：块交叉分布奇偶校验磁盘阵列

数据以块交叉的方式存于各盘，无专用冗余盘，奇偶校验信息均匀分布在所有磁盘上。

14.RAID6：P＋Q双校验磁盘阵列

特点

校验空间开销是RAID5的两倍

容忍两个磁盘出错

15.RAID10与RAID01：RAID10又称为RAID1+0，先进行镜像（RAID1），再进行条带存放（RAID0）

RAID01又称为RAID0+1先进行条带存放（RAID0），再进行镜像（RAID1）

16.实现盘阵列的方式主要有三种：

软件方式：阵列管理软件由主机来实现。

优点：成本低

缺点：过多地占用主机时间，且带宽指标上不去。

阵列卡方式：把RAID管理软件固化在I/O控制卡上，从而可不占用主机时间，一般用于工作站和PC机。

子系统方式：一种基于通用接口总线的开放式平台，可用于各种主机平台和网络系统。

6.5

1. 通道处理机（简称通道）：专门负责整个计算机系统的输入/输出工作。通道处理机只能执行有限的一组输入/输出指令。
2. 程序控制、中断和DMA方式管理外围设备会引起两个问题：

所有外设的输入/输出工作均由CPU承担，CPU的计算工作经常被打断而去处理输入/输出的事务，不能充分发挥CPU的计算能力。

大型计算机系统的外设虽然很多，但同时工作的机会不是很多。

3.4级层次结构的输入/输出系统：由CPU、通道、设备控制器、外设构成

1. 通道的功能：

接收CPU发来的I/O指令，并根据指令要求选择指定的外设与通道相连接。

执行通道程序：从主存中逐条取出通道指令，对通道指令进行译码，并根据需要向被选中的设备控制器发出各种操作命令。

给出外设中要进行读/写操作的数据所在的地址，如磁盘存储器的柱面号、磁头号、扇区号等。

给出主存缓冲区的首地址，该缓冲区存放从外设输入的数据或者将要输出到外设中去的数据。

控制外设与主存缓冲区之间的数据传送的长度。对传送的数据个数进行计数，并判断数据传送是否结束。

指定传送工作结束时要进行的操作例如：将外设的中断请求及通道的中断请求送往CPU等。

检查外设的工作状态是否正常，并将该状态信息送往主存指定单元保存。

在数据传输过程中完成必要的格式变换。例如：把字拆分为字节，或者把字节装配成字等

5.通道的主要硬件

寄存器：数据缓冲寄存器、主存地址计数器、传输字节数计数器、通道命令字寄存器、通道状态字寄存器

控制逻辑：分时控制、地址分配、数据传送、装配和拆分等

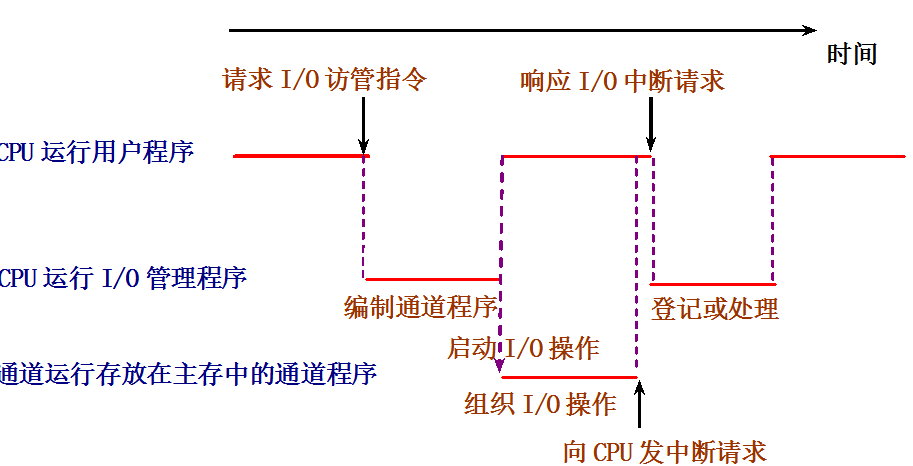
6.通道完成一次数据输入/输出的工作过程

在用户程序中使用访管指令进入管理程序，由管理程序生成一个通道程序，并启动通道。

通道处理机执行通道程序，完成指定的数据输入/输出工作

通道程序结束后向CPU发中断请求

1. CPU执行程序和通道执行通道程序的时间关系



8.通道的种类：根据信息传送方式的不同，将通道分为三种类型

字节多路通道

选择通道

数组多路通道

9.字节多路通道

为多台低速或中速的外设服务。

以字节交叉的方式分时轮流地为它们服务。

字节多路通道可以包含多个子通道，每个子通道连接一台设备控制器。

10.选择通道

为多台高速外围设备服务。

在一段时间内只为一台高速外设独占使用。

选择通道的硬件

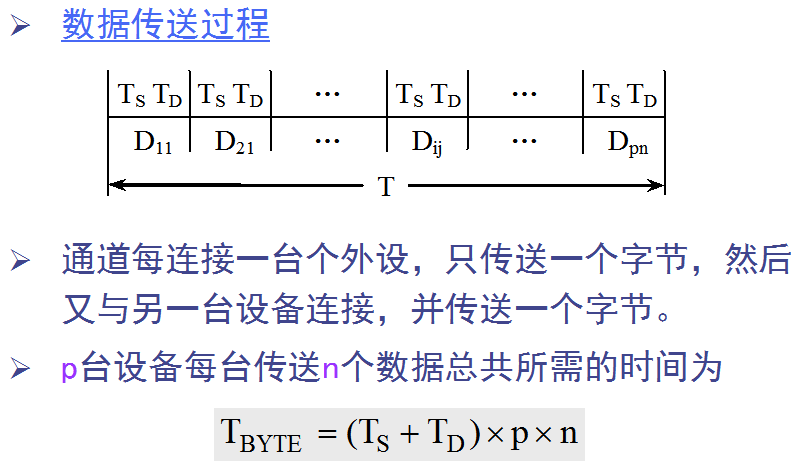
11.数组多路通道

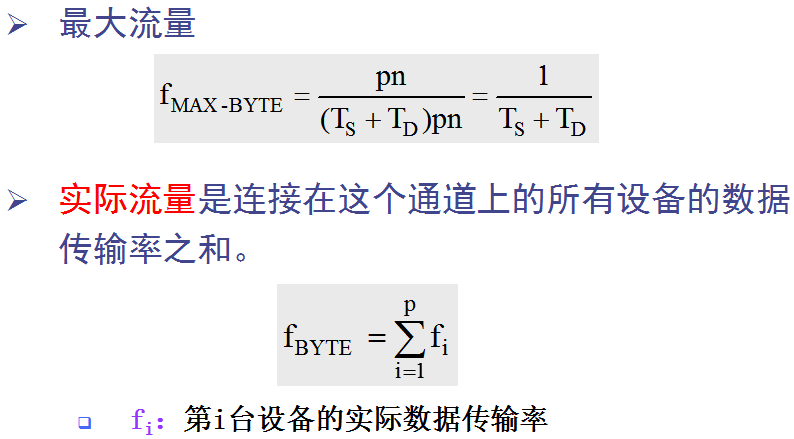
适用于高速设备。

每次选择一个高速设备后传送一个数据块，轮流为多台外围设备服务。

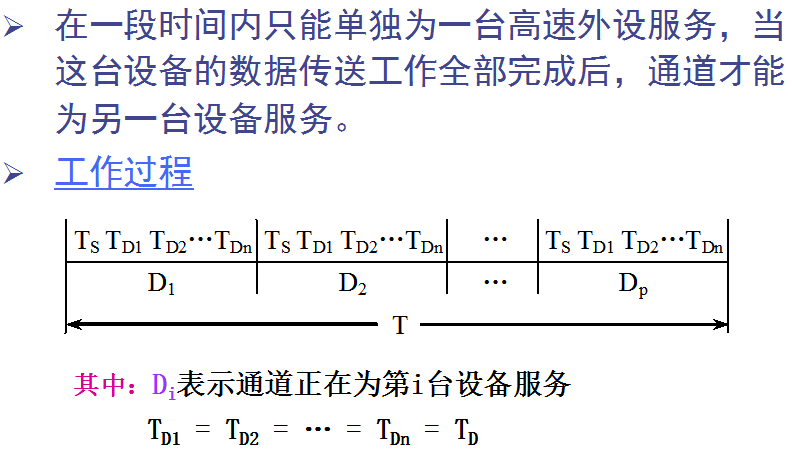
数组多路通道之所以能够并行地为多台高速设备服务，是因为虽然其所连设备的传输速率很高，但寻址等辅助操作时间很长。

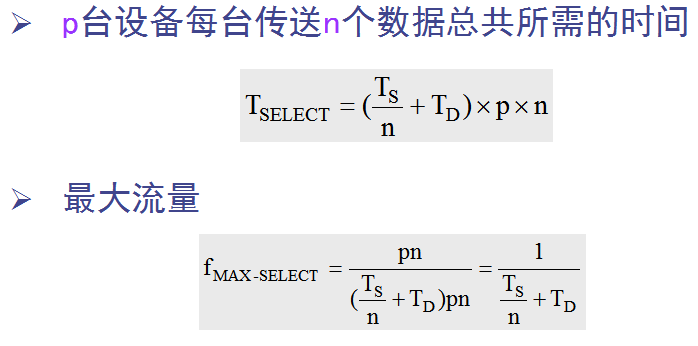
1. 通道流量：又称为通道吞吐率、通道数据传输率：一个通道在数据传送期间，单位时间内能够传送的数据量。所用单位一般为Bps。
2. 通道最大流量 ：一个通道在满负荷工作状态下的流量 。
3. 字节多路通道：



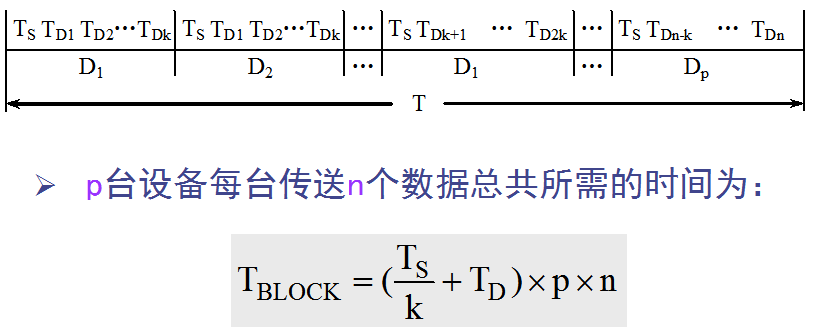


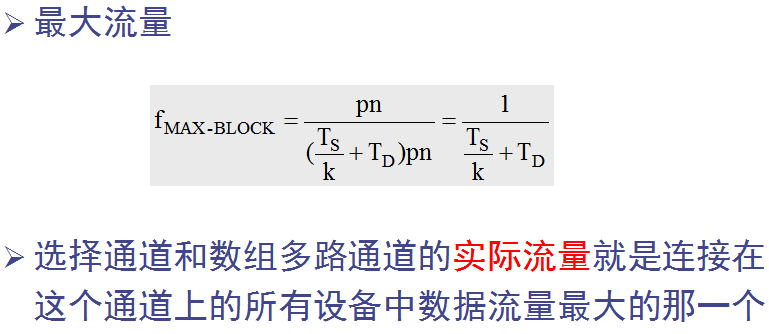
1. 选择通道：

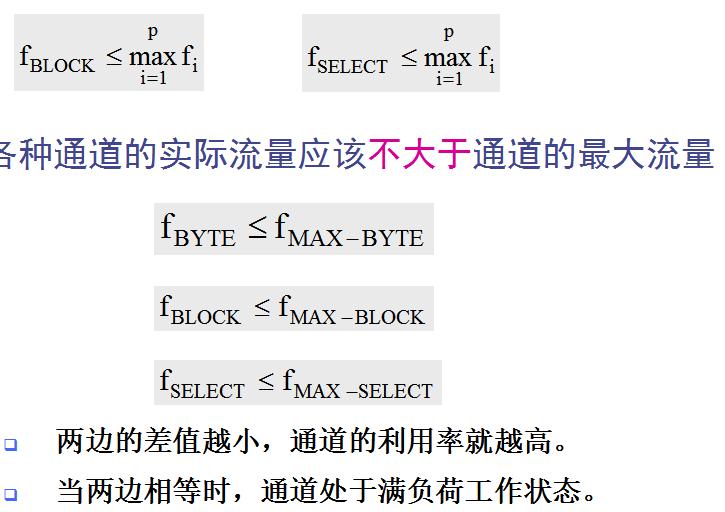




1. 数组多路通道







6.6

1.DMA和虚拟存储器：使用物理地址进行DMA传输，存在以下两个问题：

对于超过一页的数据缓冲区，由于缓冲区使用的页面在物理存储器中不一定是连续的，所以传输可能会发生问题。

如果DMA正在存储器和缓冲区之间传输数据时，操作系统从存储器中移出（或重定位）一些页面，那么，DMA将会在存储器中错误的物理页面上进行数据传输。

2.解决这些问题的方法

使操作系统在I/O的传输过程中确保DMA设备所访问的页面都位于物理存储器中，这些页面被称为是钉在了主存中。

“虚拟DMA”技术

允许DMA设备直接使用虚拟地址，并在DMA期间由硬件将虚拟地址转换为物理地址。

在采用虚拟DMA的情况下，如果进程在内存中被移动，操作系统应该能够及时地修改相应的DMA地址表。

3.I/O和Cache数据一致性

Cache会使一个数据出现两个副本：一个在Cache中，另一个在主存中

解决内容一致性问题的方法（不管Cache是采用写直达法还是写回法）

软件的方法

设法保证I/O缓冲器中的所有各块都不在Cache中。

具体做法有两种

把I/O缓冲器的页面设置为不可进入Cache的，在进行输入操作时，操作系统总是把输入的数据放到该页面上。

在进行输入操作之前，操作系统先把Cache中与I/O缓冲器相关的数据“赶出”Cache，即把相应的数据块设置为“无效”状态。

硬件的方法

在进行输入操作时，检查相应的I/O地址（I/O缓冲器中的单元）是否在Cache中（即是否有数据副本）。

如果发现I/O地址在Cache中有匹配的项，就把相应的Cache块设置为“无效”。

4.提高I/O性能的一种方法：并行：在程序中使多个磁盘同时工作，并行地访问数据。

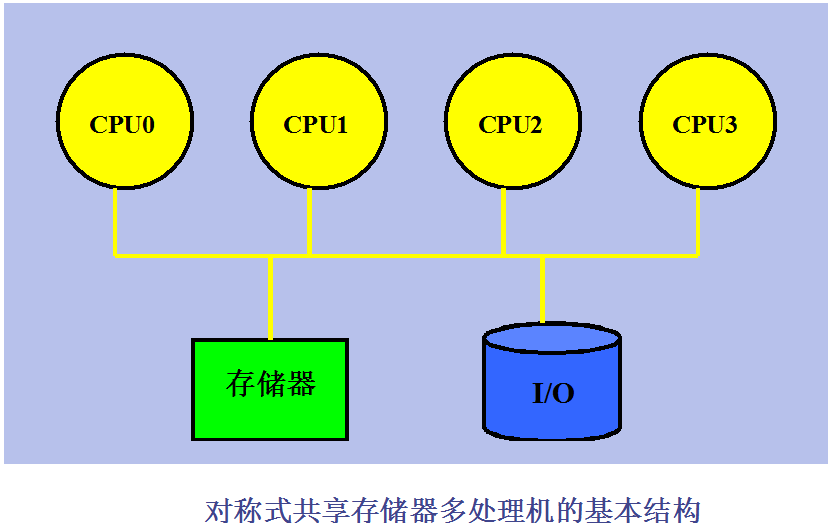
5.同步I/O进程处于等待状态直到所需的数据从磁盘读出。

6.异步I/O 允许进程在发出I/O请求后继续执行，直到该进程真正访问这些数据而它们又尚未就绪时，才被挂起。

第八章

1. 并行计算机系统结构的分类:Flynn分类法:SISD、SIMD、MISD、MIMD
2. MIMD(多指令多数据)已成为通用多处理机系统结构的选择，原因：MIMD具有灵活性；MIMD可以充分利用商品化微处理器在性能价格比方面的优势。
3. 根据存储器的组织结构 ，把现有的MIMD机器分为两类：集中式共享存储器结构、分布式存储器多处理机
4. 集中式共享存储器结构：最多由几十个处理器构成。各处理器共享一个集中式的物理存储器。

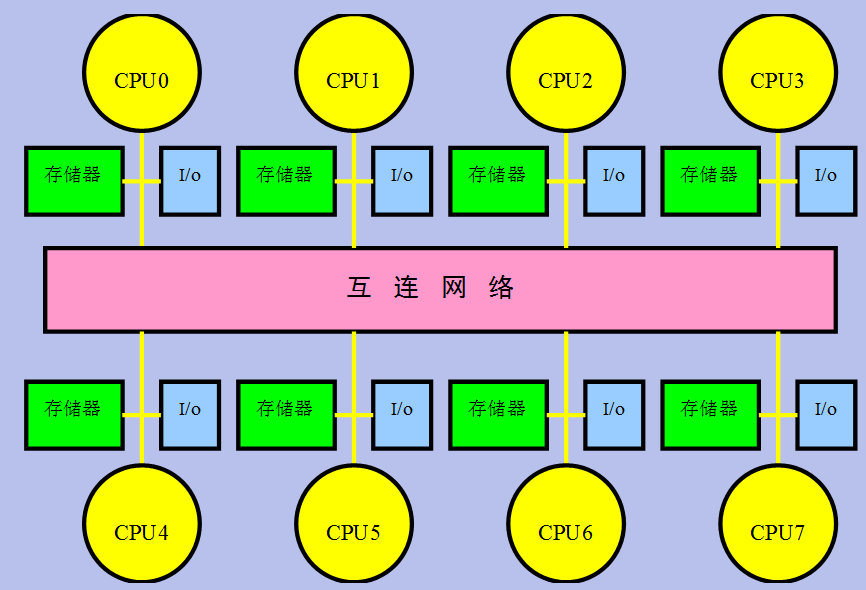
这类机器有时被称为 SMP机器UMA机器



5.分布式存储器多处理机:存储器在物理上是分布的。

每个结点包含：处理器、存储器、I／O、互连网络接口

在许多情况下，分布式存储器结构优于集中式共享存储器结构。



6.将存储器分布到各结点有两个优点：

如果大多数的访问是针对本结点的局部存储器，则可降低对存储器和互连网络的带宽要求；

对本地存储器的访问延迟时间小。

最主要的缺点：

处理器之间的通信较为复杂，且各处理器之间访问延迟较大。

解决方法：簇：超级结点

每个结点内包含个数较少（例如2～8）的处理器；

处理器之间可采用另一种互连技术（例如总线）相互连接形成簇。

1. 两种存储器系统结构和通信机制：共享地址空间、把每个结点中的存储器编址为一个独立的地址空间，不同结点中的地址空间之间是相互独立的
2. 共享地址空间：物理上分离的所有存储器作为一个统一的共享逻辑空间进行编址。任何一个处理器可以访问该共享空间中的任何一个单元（如果它具有访问权），而且不同处理器上的同一个物理地址指向的是同一个存储单元。
3. 这类计算机被称为：分布式共享存储器系统 NUMA机器
4. 把每个结点中的存储器编址为一个独立的地址空间，不同结点中的地址空间之间是相互独立的：整个系统的地址空间由多个独立的地址空间构成。每个结点中的存储器只能由本地的处理器进行访问，远程的处理器不能直接对其进行访问。

每一个处理器-存储器模块实际上是一台单独的计算机

现在的这种机器多以集群的形式存在

1. 通信机制：共享存储器通信机制、消息传递通信机制
2. 共享存储器通信机制：共享地址空间的计算机系统采用、处理器之间是通过用load和store指令对相同存储器地址进行读/写操作来实现的。
3. 消息传递通信机制：

多个独立地址空间的计算机采用

通过处理器间显式地传递消息来完成

消息传递多处理机中，处理器之间是通过发送消息来进行通信的，这些消息请求进行某些操作或者传送数据。

14.一个处理器要对远程存储器上的数据进行访问或操作：

发送消息，请求传递数据或对数据进行操作：远程进程调用(RPC， Remote Process Call)

目的处理器接收到消息以后，执行相应的操作或代替远程处理器进行访问，并发送一个应答消息将结果返回。

15.同步消息传递： 请求处理器发送一个消息后一直要等到应答结果才继续运行。

16.异步消息传递 ：数据发送方知道别的处理器需要数据，通信也可以从数据发送方来开始，数据可以不经请求就直接送往数据接受方。

17.共享存储器通信的主要优点 ：

与常用的对称式多处理机使用的通信机制兼容。

易于编程，同时在简化编译器设计方面也占有优势。

采用大家所熟悉的共享存储器模型开发应用程序，而把重点放到解决对性能影响较大的数据访问上。

当通信数据量较小时，通信开销较低，带宽利用较好。

可以通过采用Cache技术来减少远程通信的频度，减少了通信延迟以及对共享数据的访问冲突。

18.消息传递通信机制的主要优点：

硬件较简单。

通信是显式的，因此更容易搞清楚何时发生通信以及通信开销是多少。

显式通信可以让编程者重点注意并行计算的主要通信开销，使之有可能开发出结构更好、性能更高的并行程序。

同步很自然地与发送消息相关联，能减少不当的同步带来错误的可能性。

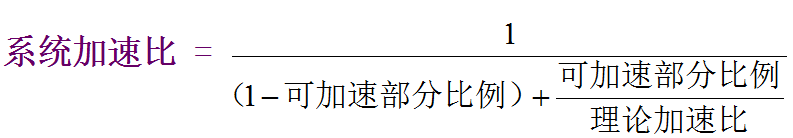
19,。可在支持上面任何一种通信机制的硬件模型上建立所需的通信模式平台。

在共享存储器上支持消息传递相对简单。

在消息传递的硬件上支持共享存储器就困难得多。所有对共享存储器的访问均要求操作系统提供地址转换和存储保护功能，即将存储器访问转换为消息的发送和接收。

20.并行处理面临着两个重要的挑战：程序中的并行性有限；相对较大的通信开销

21.Amdahl定律:



22.问题的解决

并行性不足： 采用并行性更好的算法

远程访问延迟的降低：靠系统结构支持和编程技术

23.在并行处理中，影响性能（负载平衡、同步和存储器访问延迟等）的关键因素常依赖于：

应用程序的高层特性,如数据的分配，并行算法的结构以及在空间和时间上对数据的访问模式等。

24.反映并行程序性能的一个重要的度量:计算与通信的比率

计算／通信比率随着处理数据规模的增大而增加；随着处理器数目的增加而减少。

8.2

1.对称式共享存储器系统结构:多个处理器共享一个存储器。当处理机规模较小时，这种计算机十分经济。支持对共享数据和私有数据的Cache缓存私有数据供一个单独的处理器使用，而共享数据则是供多个处理器使用。

共享数据进入Cache产生了一个新的问题：Cache的一致性问题

2.多处理机的Cache一致性问题：允许共享数据进入Cache，就可能出现多个处理器的Cache中都有同一存储块的副本，

当其中某个处理器对其Cache中的数据进行修改后，就会使得其Cache中的数据与其他Cache中的数据不一致。

3.存储器的一致性：如果对某个数据项的任何读操作均可得到其最新写入的值，则认为这个存储系统是一致的。

存储系统行为的两个不同方面：

What: 读操作得到的是什么值

When: 什么时候才能将已写入的值返回给读操作

存储器是一致的，需要满足以下条件：

处理器P对单元X进行一次写之后又对单元X进行读，读和写之间没有其他处理器对单元X进行写，则P读到的值总是前面写进去的值。

处理器P对单元X进行写之后，另一处理器Q对单元X进行读，读和写之间无其他写，则Q读到的值应为P写进去的值。

对同一单元的写是串行化的，即任意两个处理器对同一单元的两次写，从各个处理器的角度看来顺序都是相同的。(写串行化 )

4.在一致的多处理机中，Cache提供两种功能：

共享数据的迁移：减少了对远程共享数据的访问延迟，也减少了对共享存储器带宽的要求。

共享数据的复制：不仅减少了访问共享数据的延迟，也减少了访问共享数据所产生的冲突。

5.一般情况下，小规模多处理机是采用硬件的方法来实现Cache的一致性。

6.Cache一致性协议：在多个处理器中用来维护一致性的协议。

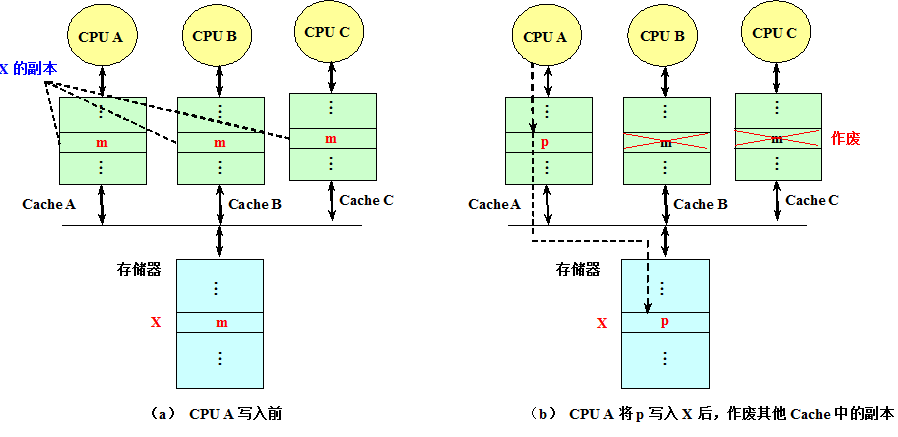
关键：跟踪记录共享数据块的状态

两类协议（采用不同的技术跟踪共享数据的状态）

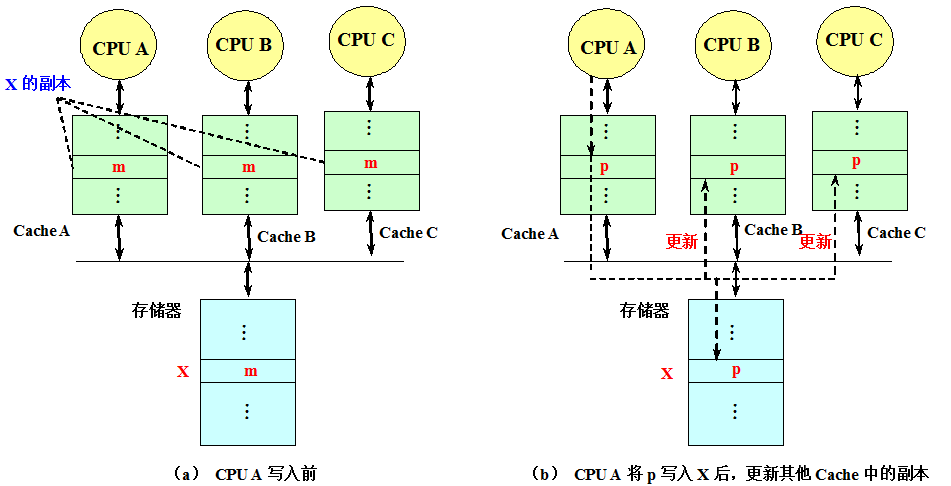
目录式协议（directory）：物理存储器中数据块的共享状态被保存在一个称为目录的地方。

监听式协议（snooping）：每个Cache除了包含物理存储器中块的数据拷贝之外，也保存着各个块的共享状态信息。Cache通常连在共享存储器的总线上，当某个Cache需要访问存储器时，它会把请求放到总线上广播出去，其他各个Cache控制器通过监听总线（它们一直在监听）来判断它们是否有总线上请求的数据块。如果有，就进行相应的操作。

1. 采用两种方法来解决Cache一致性问题：写作废协议、写更新协议
2. 写作废协议：在处理器对某个数据项进行写入之前，保证它拥有对该数据项的唯一的访问权



1. 写更新协议：当一个处理器对某数据项进行写入时，通过广播使其他Cache中所有对应于该数据项的副本进行更新



1. 写更新和写作废协议性能上的差别主要来自：

在对同一个数据进行多次写操作而中间无读操作的情况下，写更新协议需进行多次写广播操作，而写作废协议只需一次作废操作。

在对同一Cache块的多个字进行写操作的情况下，写更新协议对于每一个写操作都要进行一次广播，而写作废协议仅在对该块的第一次写时进行作废操作即可。

写作废是针对Cache块进行操作，而写更新则是针对字（或字节）进行。

写更新协议的延迟时间较小。

1. 监听协议：实现监听协议的关键有3个方面

处理器之间通过一个可以实现广播的互连机制相连，通常采用的是总线。

当一个处理器的Cache响应本地CPU的访问时，如果它涉及全局操作，其Cache控制器就要在获得总线的控制权后，在总线上发出相应的消息。

所有处理器都一直在监听总线，它们检测总线上的地址在它们的Cache中是否有副本。若有，则响应该消息，并进行相应的操作 。

1. Cache发送到总线上的消息主要有以下两种：

RdMiss——读不命中

WtMiss——写不命中

13.需要通过总线找到相应数据块的最新副本，然后调入本地Cache中。

写直达Cache：因为所有写入的数据都同时被写回主存，所以从主存中总可以取到其最新值。

写回Cache：得到数据的最新值会困难一些，因为最新值可能在某个Cache中，也可能在主存中。

14.Invalidate消息：用来通知其他各处理器作废其Cache中相应的副本。

与WtMiss的区别：Invalidate不引起调块

15.Cache的标识（tag）可直接用来实现监听。

16.作废一个块只需将其有效位置为无效。

17.给每个Cache块增设一个共享位：为“1”：该块是被多个处理器所共享；为“0”：仅被某个处理器所独占

块的拥有者：拥有该数据块的唯一副本的处理器

18.监听协议举例 ：

在每个结点内嵌入一个有限状态控制器。

该控制器根据来自处理器或总线的请求以及Cache块的状态，做出相应的响应

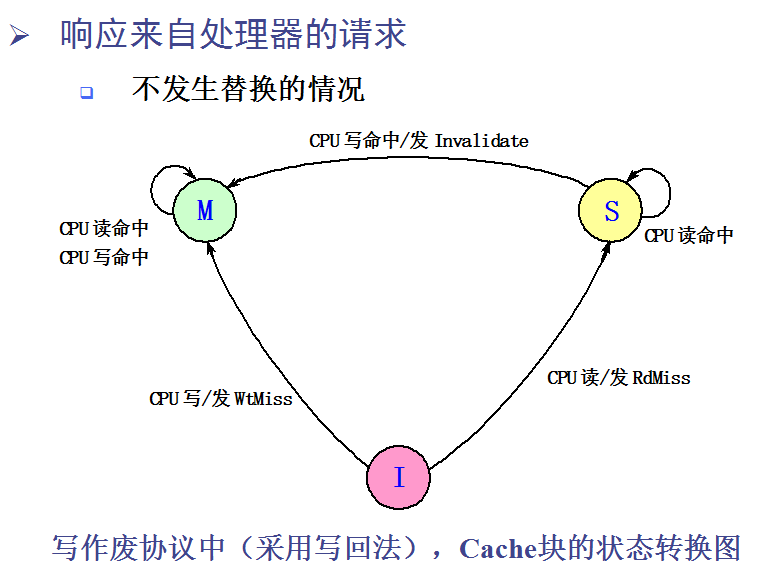
19.每个数据块的状态取以下3种状态中的一种：

无效（简称I）：Cache中该块的内容为无效。

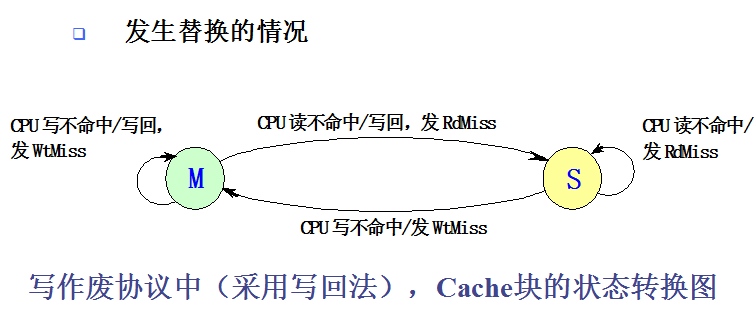
共享（简称S）：该块可能处于共享状态。在多个处理器中都有副本。这些副本都相同，且与存储器中相应的块相同。

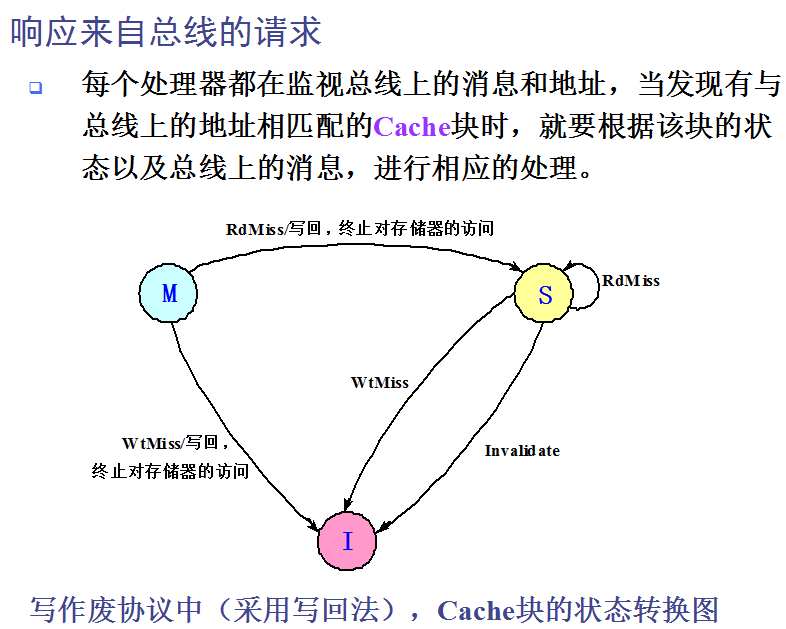
已修改（简称M）：该块已经被修改过，并且还没写入存储器。（块中的内容是最新的，系统中唯一的最新副本）

20.



21.



1. 

8.3

1.目录协议

目录：一种集中的数据结构。对于存储器中的每一个可以调入Cache的数据块，在目录中设置一条目录项，用于记录该块的状态以及哪些Cache中有副本等相关信息。

特点：对于任何一个数据块，都可以快速地在唯一的一个位置中找到相关的信息。这使一致性协议避免了广播操作。

1. 目录法采用位向量：记录哪些Cache中有副本。

每一位对应于一个处理器。

长度与处理器的个数成正比。

由位向量指定的处理机的集合称为共享集S

1. 分布式目录：目录与存储器一起分布到各结点中，从而对于不同目录内容的访问可以在不同的结点进行
2. 在目录协议中，存储块的状态有3种：

未缓冲：该块尚未被调入Cache。所有处理器的Cache中都没有这个块的副本。

共享：该块在一个或多个处理机上有这个块的副本，且这些副本与存储器中的该块相同。

独占：仅有一个处理机有这个块的副本，且该处理机已经对其进行了写操作，所以其内容是最新的，而存储器中该块的数据已过时。这个处理机称为该块的拥有者。

1. 本地结点、宿主结点以及远程结点的关系

本地结点：发出访问请求的结点

宿主结点：包含所访问的存储单元及其目录项的结点

远程结点：拥有相应存储块的副本。

远程节点可以和宿主结点是同一个结点，也可以和本地节点是同一个节点。本地节点可以和宿主节点是同一个。

6.本地结点发给宿主结点（目录）的消息：

RdMiss（P，K）：处理机P读取地址为A的数据时不命中，请求宿主结点提供数据（块），并要求把P加入共享集。

WtMiss（P，K） ：处理机P对地址A进行写入时不命中，请求宿主结点提供数据，并使P成为所访问数据块的独占者。

Invalidate（K） 请求向所有拥有相应数据块副本（包含地址K）的远程Cache发Invalidate消息，作废这些副本

7.宿主结点（目录）发送给远程结点的消息：

Invalidate（K）作废远程Cache中包含地址K的数据块。

Fetch（K）:从远程Cache中取出包含地址K的数据块，并将之送到宿主结点。把远程Cache中那个块的状态改为“共享”。

Fetch&Inv（K）:从远程Cache中取出包含地址K的数据块，并将之送到宿主结点。然后作废远程Cache中的那个块。

8.宿主结点发送给本地结点的消息

DReply（D）D表示数据内容。把从宿主存储器获得的数据返回给本地Cache。

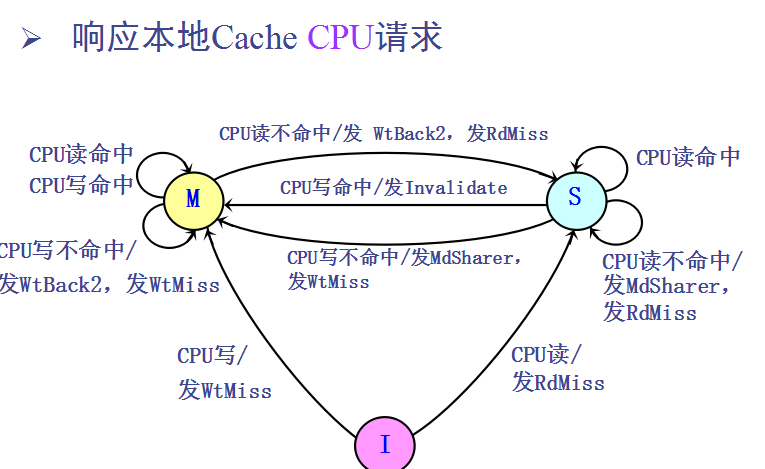
1. 远程结点发送给宿主结点的消息

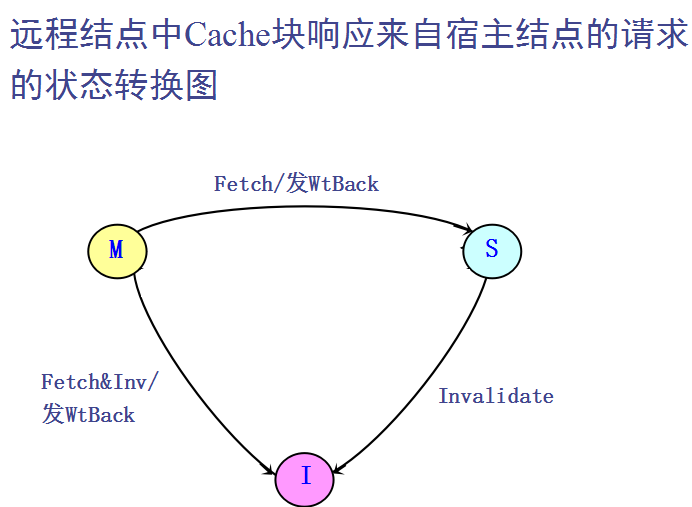
WtBack（K，D）把远程Cache中包含地址K的数据块写回到宿主结点中， 该消息是远程结点对宿主结点发来的“取数据”或“取/作废”消息的响应。

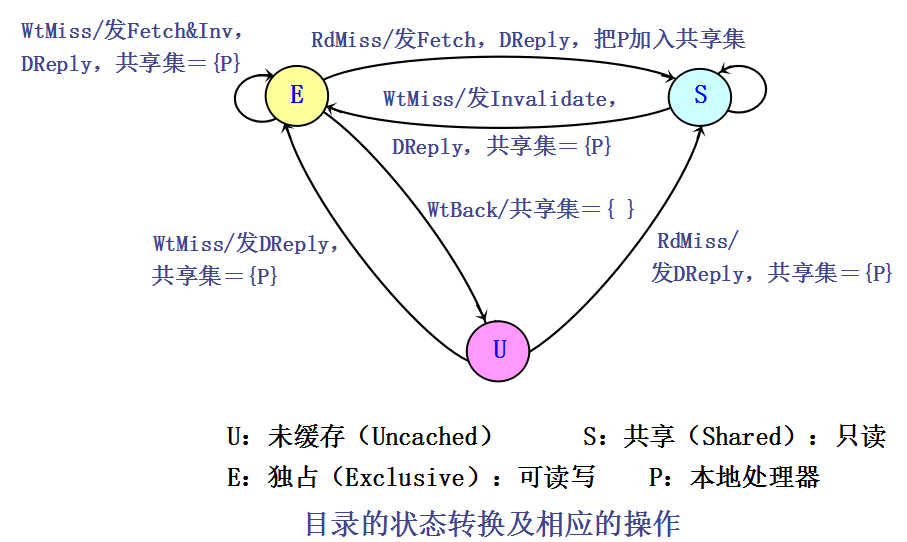
10.本地结点发送给被替换块的宿主结点的消息

MdSharer（P，K）用于当本地Cache中需要替换一个包含地址K的块、且该块未被修改过的情况。这个消息发给该块的宿主结点，请求它将P从共享集中删除。如果删除后共享集变为空集，则宿主结点还要将该块的状态改变为“未缓存”（U）。

WtBack2（P，K，D）用于当本地Cache中需要替换一个包含地址K的块、且该块已被修改过的情况。这个消息发给该块的宿主结点，完成两步操作：①把该块写回；②进行与MdSharer相同的操作。







11.当一个块处于未缓存状态时，对该块发出的请求及处理操作为：

RdMiss（读不命中）

将所要访问的存储器数据送往请求方处理机，且该处理机成为该块的唯一共享结点，本块的状态变成共享。

WtMiss（写不命中）

将所要访问的存储器数据送往请求方处理机，该块的状态变成独占，表示该块仅存在唯一的副本。其共享集合仅包含该处理机，指出该处理机是其拥有者。

12.当一个块处于共享状态时，其在存储器中的数据是当前最新的，对该块发出的请求及其处理操作为：

RdMiss

将存储器数据送往请求方处理机，并将其加入共享集合。

WtMiss

将数据送往请求方处理机，对共享集合中所有的处理机发送作废消息，且将共享集合改为仅含有该处理机，该块的状态变为独占。

13.当某块处于独占状态时，该块的最新值保存在共享集合所指出的唯一处理机（拥有者）中。

有三种可能的请求：

RdMiss

将“取数据”的消息发往拥有者处理机，将它所返回给宿主结点的数据写入存储器，并进而把该数据送回请求方处理机，将请求方处理机加入共享集合。

此时共享集合中仍保留原拥有者处理机（因为它仍有一个可读的副本）。

将该块的状态变为共享。

WtMiss

该块将有一个新的拥有者。

给旧的拥有者处理机发送消息，要求它将数据块送回宿主结点写入存储器，然后再从该结点送给请求方处理机。

同时还要把旧拥有者处理机中的该块作废。把请求处理机加入共享者集合，使之成为新的拥有者。该块的状态仍旧是独占。

WtBack（写回）

当一个块的拥有者处理机要从其Cache中把该块替换出去时，必须将该块写回其宿主结点的

存储器中，从而使存储器中相应的块中存放的数据是最新的（宿主结点实际上成为拥有者）；

该块的状态变成未缓冲，其共享集合为空。

14.目录协议的三种结构：全映像目录、有限映像目录、链式目录

15.不同目录协议的主要区别主要有两个：所设置的存储器块的状态及其个数不同；目录的结构

16.全映像目录： 每一个目录项都包含一个N位（N为处理机的个数）的位向量，其每一位对应于一个处理机。

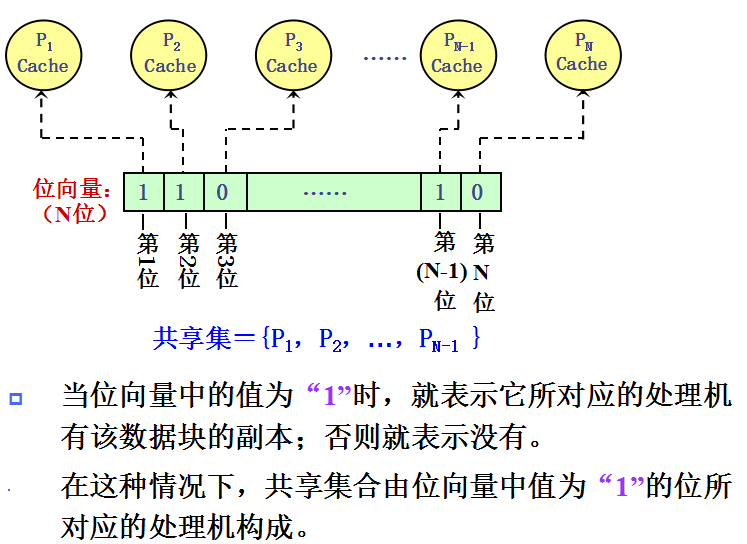
优点：处理比较简单，速度也比较快。

缺点：

存储空间的开销很大。

目录项的数目与处理机的个数N成正比，而目录项的大小（位数）也与N成正比，因此目录所占用的空间与N2成正比。

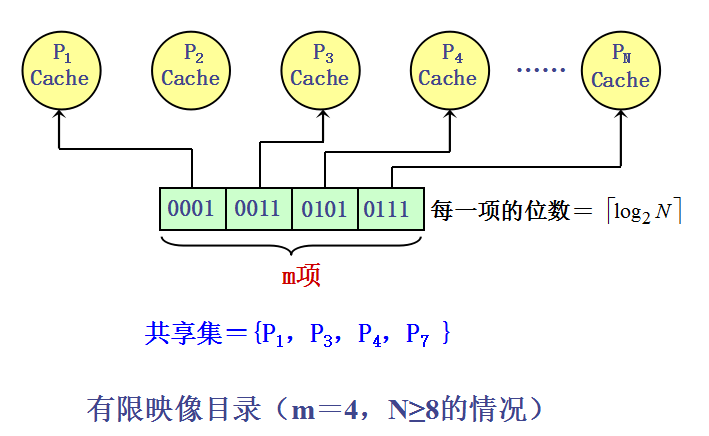
可扩放性很差。



17.有限映像目录 ：提高其可扩放性和减少目录所占用的空间。

核心思想：采用位数固定的目录项目

限制同一数据块在所有Cache中的副本总数。



缺点

当同一数据的副本个数大于m时，必须做特殊处理。当目录项中的m个指针都已经全被占满，而某处理机又需要新调入该块时，就需要在其m个指针中选择一个，将之驱逐，以便腾出位置，存放指向新调入块的处理机的指针

1. 链式目录

用一个目录指针链表来表示共享集合。当一个数据块的副本数增加（或减少）时，其指针链表就跟着变长（或变短）。

由于链表的长度不受限制，因而带来了以下优点：既不限制副本的个数，又保持了可扩展性

链式目录有两种实现方法

单链法

当Cache中的块被替换出去时，需要对相应的链表进行操作——把相应的链表元素（假设是链表中的第i个）删除。实现方法有以下两种：

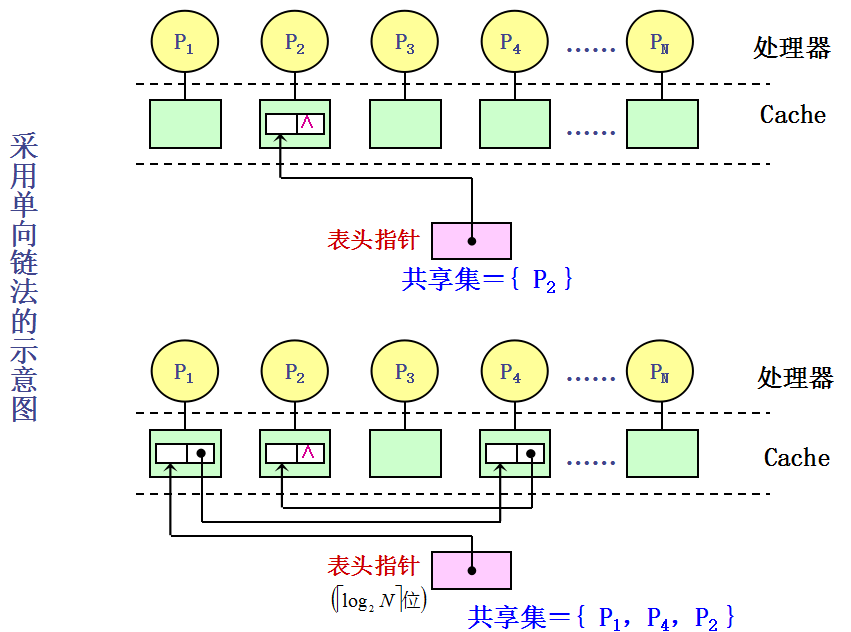
沿着链表往下寻找第i个元素，找到后，修改其前后的链接指针，跳过该元素。

找到第i个元素后，作废它及其后的所有元素所对应的Cache副本。

双链法

在替换时不需要遍历整个链表。

节省了处理时间，但其指针增加了一倍，而且一致性协议也更复杂了。



8.5线程级并行性

线程是进程内的一个相对独立且可独立调度和指派的执行单元，它比进程要“轻巧”得多。

只拥有在运行过程中必不可少的一点资源，如：程序计数器、一组寄存器、堆栈等。

线程切换时，只需保存和设置少量寄存器的内容，开销很小。

线程切换只需要几个时钟周期。

进程的切换一般需要成百上千个处理器时钟周期。

1. 实现多线程有两种主要的方法：细粒度多线程、粗粒度多线程
2. 细粒度多线程

在每条指令之间都能进行线程的切换，从而使得多个线程可以交替执行。

通常以时间片轮转的方法实现这样的交替执行，在轮转的过程中跳过当时处于停顿的线程。

CPU必须在每个时钟周期都能进行线程的切换。

主要优点：既能够隐藏由长时间停顿引起的吞吐率的损失，又能够隐藏由短时间停顿带来的损失。

主要缺点：减慢了单个线程的执行

1. 粗粒度（coarse-grained）多线程

线程之间的切换只发生在时间较长的停顿出现时。

减少了切换次数，也不太会降低单个线程的执行速度 。

缺点：减少吞吐率损失的能力有限，特别是对于较短的停顿来说更是如此。

原因：由粗粒度多线程的流水线建立时间的开销造成的。由于实现粗粒度多线程的CPU只执行单个线程的指令，因此当发生停顿时，流水线必须排空或暂停。停顿后切换的新线程也有个填满流水线的过程，填满后才能不断地流出指令执行结果。

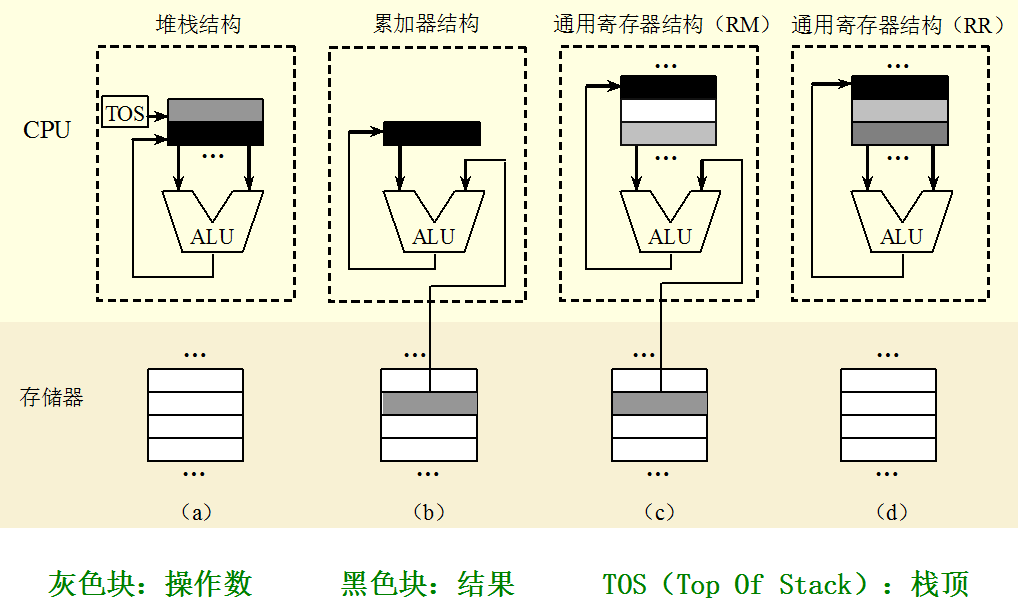
1. 同时多线程技术：一种在多流出、动态调度的处理器上同时开发线程级并行和指令级并行的技术。

第二章计算机指令集结构

2.1

1. 区别不同指令集结构的主要因素：CPU中用来存储操作数的存储单元的类型
2. CPU中用来存储操作数的存储单元的主要类型：堆栈、累加器、通用寄存器组
3. 将指令集结构分为三种类型：堆栈结构、累加器结构、通用寄存器结构
4. 据操作数的来源不同，又可进一步分为:寄存器-存储器结构（RM结构）(操作数可以来自存储器 )寄存器-寄存器结构（RR结构）(所有操作数都是来自通用寄存器组)
5. 对指令集的基本要求是：完整性、规整性、高效率和兼容性
6. 完整性是指在一个有限可用的存储空间内，对于任何可解的问题，编制计算程序时，指令集所提供的指令足够使用。
7. 规整性主要包括对称性和均匀性。对称性是指所有与指令集有关的存储单元的使用、操作码的设置等都是对称的。均匀性是指对于各种不同的操作数类型、字长、操作种类和数据存储单元，指令的设置都要同等对待。
8. 高效率是指指令的执行速度快、使用频度高
9. 指令集结构设计所涉及的内容有哪些？

答： (1) 指令集功能设计：主要有RISC和CISC两种技术发展方向； (2) 寻址方式的设计：设置寻址方式可以通过对基准程序进行测试统计，察看各种寻址方式的使用频率，根据适用频率设置必要的寻址方式。 (3) 操作数表示和操作数类型：主要的操作数类型和操作数表示的选择有：浮点数据类型、整型数据类型、字符型、十进制数据类型等等。 (4) 寻址方式的表示：可以将寻址方式编码于操作码中，也可以将寻址方式作为一个单独的域来表示。 (5) 指令集格式的设计：有变长编码格式、固定长度编码格式和混合型编码格式3种

1. 
2. CISC: 复杂指令集计算机 RISC：精简指令集计算机
3. 寻址方式：指令集结构如何确定所要访问的数据的地址
4. 数据表示：指计算机硬件能够直接识别、指令集可以直接调用的数据类型。
5. 简述RISC指令集结构的设计原则。

答（1） 选取使用频率最高的指令，并补充一些最有用的指令；（2）每条指令的功能应尽可能简单，并在一个机器周期内完成；（3）所有指令长度均相同；（4）只有Load和Store操作指令才访问存储器，其它指令操作均在寄存器之间进行； (5) 以简单有效的方式支持高级语言

15.简述CISC指令集结构功能设计的主要目标。从当前的计算机技术观点来看，CISC指令集结构的计算机有什么缺点？

答：主要目标是增强指令功能，把越来越多的功能交由硬件来实现，并且指令的数量也是越来越多。

缺点： (1) CISC结构的指令集中，各种指令的使用频率相差悬殊。（2）CISC结构指令的复杂性带来了计算机体系结构的复杂性，这不仅增加了研制时间和成本，而且还容易造成设计错误。（3）CISC结构指令集的复杂性给VLSI设计增加了很大负担，不利于单片集成。（4）CISC结构的指令集中，许多复杂指令需要很复杂的操作，因而运行速度慢。 (5) 在CISC结构的指令集中，由于各条指令的功能不均衡性，不利于采用先进的计算机体系结构技术（如流水技术）来提高系统的性能。

1. 操作数类型有两种表示方法：（1）操作数的类型由操作码的编码指定，这是最常见的一种方法；（2）数据可以附上由硬件解释的标记，由这些标记指定操作数的类型，从而选择适当的运算。
2. 表示寻址方式有两种常用的方法：（1）将寻址方式编于操作码中，由操作码在描述指令的同时也描述了相应的寻址方式。这种方式译码快，但操作码和寻址方式的结合不仅增加了指令的条数，导致了指令的多样性，而且增加了CPU对指令译码的难度。（2）为每个操作数设置一个地址描述符，由该地址描述符表示相应操作数的寻址方式。这种方式译码较慢，但操作码和寻址独立，易于指令扩展
3. 通常有哪几种指令格式，请简述其适用范围。

答： (1) 变长编码格式。如果系统结构设计者感兴趣的是程序的目标代码大小，而不是性能，就可以采用变长编码格式。（2）固定长度编码格式。如果感兴趣的是性能，而不是程序的目标代码大小，则可以选择固定长度编码格式。 (3) 混合型编码格式。需要兼顾降低目标代码长度和降低译码复杂度时，可以采用混合型编码格式。

19.根据CPU性能公式简述RISC指令集结构计算机和CISC指令集结构计算机的性能特点。

答：CPU性能公式：CPU时间＝IC×CPI×T

其中，IC为目标程序被执行的指令条数，CPI为指令平均执行周期数，T是时钟周期的时间。

相同功能的CISC目标程序的指令条数ICCISC 少于RISC的ICRISC，但是CISC的CPICISC和TCISC都大于RISC的CPIRISC和TRISC，因此，CISC目标程序的执行时间比RISC的更长

1. 输入输出系统硬件的功能对(B )是透明的 应用程序员
2. 属于MIMD系统结构 松耦合多处理机和多计算机系统
3. 计算机使用的语言是 分属于计算机系统各个层次
4. 计算机仿真用      解释，计算机模拟用   解释 微程序、机器语言
5. 阵列处理机按存储器的组成方式可分为两种构形,分别为 分布式存储器和集中共享式
6. 对系统程序员不透明的应当是 虚拟存储器
7. 数据宽度是 I／Ｏ设备取得Ｉ／Ｏ总线后所传送数据的总量
8. 实现软件移植的基本技术有  同一高级语言， 采用系列机， 模拟和仿真
9. 并行主存系统包括   单体多字，多体单字 和 多体多字交叉存储器
10. 举例说明几种指令级高度并行的超级处理机 超标量、超流水线处理机、超长指令字
11. 计算机系统中提高并行性的技术途径有 时间重叠、资源重复、资源共享
12. 对向量的处理方式 横向处理方式、纵向处理方式、横纵处理方式
13. 指令集结构设计涉及哪些问题 指令集功能设计、寻址方式设计、寻址方式表示、操作数表示和操作数类型
14. 解决数据冲突的方法： 定向技术、停顿时钟周期、依靠编译器解决
15. 流水线性能指标： 吞吐率、加速比、效率
16. 2:1 Cache经验规则: 容量为N 的直接映象 Cache失效率 ≈容量为N/2的两路 组相联Cache失效率
17. 输入/输出系统包括 输入/输出设备和输入/输出设备与处理机之间的连接
18. 通道分为哪三种通道 字节多路通道、选择通道、数组多路通道
19. 虚拟存储器中一般采用\_\_地址映像方法和\_\_更新策略 全相联、LRU
20. 前瞻执行的基本思想
21. Tomasulo算法基本思想
22. 调度分支延迟指令常用的三种方法 从前调度、从目标出调度、从失败处调度
23. 三级Cache的平均访问时间 TA ＝HL1＋Ml1{Hl2＋Ml2(HL3＋ML3×PL3)}
24. 目录协议中，Cache块有三种状态 共享、未缓存、独占
25. 什么是多处理机Cache的一致性，解决Cache一致性有哪些方法
26. 简述三种向量处理方式，它们对向量处理机的结构要求有何不同？
27. 答 (1)横向处理方式：若向量长度为N，则水平处理方式相当于执行N次循环。若使用流水线，在每次循环中可能出现数据相关和功能转换，不适合对向量进行流水处理。 (2)纵向处理方式：将整个向量按相同的运算处理完毕之后，再去执行其他运算。适合对向量进行流水处理，向量运算指令的源/目向量都放在存储器内，使得流水线运算部件的输入、输出端直接与存储器相联，构成M-M型的运算流水线。 (3)纵横处理方式：把长度为N的向量分为若干组，每组长度为n，组内按纵向方式处理，依次处理各组，组数为「N/n」，适合流水处理。可设长度为n的向量寄存器，使每组向量运算的源/目向量都在向量寄存器中，流水线的运算部件输入、输出端与向量寄存器相联，构成R-R型运算流水线。
28. 可采用哪些方法来提高向量处理机的性能？ 答：可采用多种方法：
29. （1） 设置多个功能部件，使它们并行工作； （2） 采用链接技术，加快一串向量指令的执行； （3） 采用循环开采技术，加快循环的处理； （4） 采用多处理机系统，进一步提高性能。