2018/2019(1) 学期高级计算机系统结构思考题

1. CPU性能公式及提高CPU性能的主要方法。

**性能公式：**

CPU时间=指令数×CPI×时钟周期时间

**主要方法：**

增大CPU字长

提高主频

减小纳米制程

增加核心数

超标量、超线程、流水线

改进芯片架构

1. Cache失效主要原因及解决方法。

**主要原因：**//（计算机组成与设计 硬件软件设计第5版 5.8.5）

强制缺失：对从没有在cache中出现的块的第一次进行访问引起的缺失。

容量缺失：由于cache容纳不了一个程序执行所需的所有块而引起的缺失，当某些块被替换出去，随后再被调入时，发生容量缺失。

冲突缺失：在组相联或者直接映射的cache中，多个块竞争同一个组时而引起的cache缺失。冲突缺失不存在于同样大小的全相联cache中。

**解决方法：**

//降低失效率

（1）增加块容量：减缓强制性失效、可能会增加冲突失效（因为总容量不变，块数量减少）

（2）增大Cache容量：对冲突和容量失效的减少有利

（3）提高相联度：能减少冲突缺失，降低缺失率但增加命中时间。

（?）通过预取可帮助减少强制性失效：必须小心不要把你需要的东西换出去、需要预测比较准确（对数据较困难，对指令相对容易）

（?）采用软件优化技术提高cache的有效性。

//减少失效开销

（4）多级Cache：减少失效开销、缩短平均访存时间（AMAT）

（5）使读失效优先于写失效：由于读操作为大概率事件，需要读失效优先，以提高性能

//缩短命中时间

（6）避免在索引缓存期间进行地址转换。

1. Tomasulo算法基本思想及如何解决流水线中的存在的数据冲突。

**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（写后读）冲突的可能性减少到最少。通过寄存器换名来消除WAR（读后写）和WAW（写后写）冲突。

**如何解决：**

（1）通过定向技术减少停顿

（2）设置流水线互锁机制

（3）依靠编译器解决

1. 如何实现Cache一致性？ //（计算机组成与设计 硬件软件设计第5版 5.10）

**迁移：**数据项可以移入本地cache并以透明的方式使用。迁移不但减少了访问远程共享数据项的延迟，而且减少了对共享存储器带宽的需求。

**复制：**当共享数据被同时读取时，cache在本地对数据项做了备份。复制减少了访问延迟和读取共享数据时的竞争现象。

为了实现对迁移和复制的支持，引入硬件协议来维护cache一致性，关键在于跟踪所有共享数据块的状态。最常用的是监听协议，每个含有物理存储器中数据块副本的cache还要保留该数据块共享状态的副本，但是并不集中地保存状态。Cache可以通过一些广播媒介（总线或者网络）访问，所有的cache控制器对媒介进行监视或监听，来确定它们是否含有总线或者交换机上请求的数据块副本。

1. 简述虚拟地址翻译机制及如何提高其性能。

以页式虚拟存储器的**翻译机制**为例，另有段式、段页式

1. 处理器产生一个虚拟地址；
2. 根据页表寄存器中的页表首地址和进程ID定位到该进程的页表；
3. 根据虚拟地址中的虚页号查找到页表中对应页表项记录的物理页号；
4. 将查得的物理页号和虚拟地址中的页偏移拼接获得物理地址；

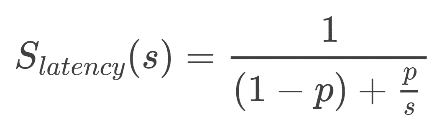
**如何提高其性能：**

使用特殊的地址转换cache（快表，TLB）：页表的高速缓存

或者直接将整个页表移入cache

1. Amdahl定理及其应用

**阿姆达尔定律**给出了任务在固定负载的情况下，随着系统资源的提升，执行速度提速比的理论上限。

Slatency : 整个任务的提速比。

s: 部分任务得益于系统资源升级带来的提速比。

p: 这部分任务执行时间占整个任务执行时间的百分比（系统资源提升前）。

**应用：**

例如，在启动一个并行化项目前，开发人员会希望预估他们能够实现的性能提升量（加速比）。 如果知道（或预估出)能够以并行方式执行的串行代码的百分数，那么开发人员可使用阿姆达尔定律计算应用的加速比上限。例如，如果 95% 的串行应用运行时间可以在 8 个内核上以并行方式执行，根据阿姆达尔定律，预估的加速比等于 6X (1 / (0.05 + 0.95/8)= 5.925)。来源：<https://software.intel.com/zh-cn/articles/predicting-and-measuring-parallel-performance/>

1. 指令静态调度及循环展开

**静态调度：**通过改变指令顺序而不改变指令间数据相关来改善指令延迟。在出现数据相关时，为了消除或者减少流水线空转，编译器确定并分离出程序中存在的相关指令，然后进行指令调度，并对代码进行优化。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的。

**循环展开：**一种从存取数组的循环中获取更多性能的技术，其中循环体会被复制多份并且不同循环体中的指令可能会调度到一起。这是一种牺牲程序的尺寸来加快程序的执行速度的优化方法。它可以由程序员完成，也可由编译器自动优化完成。循环展开通过将循环体代码复制多次实现。循环展开能够增大指令调度的空间，减少循环分支指令的开销。循环展开可以更好地实现数据预取技术。但是由于循环展开通过减少循环次数来提高程序性能，如果展开次数太多，反而会造成性能急剧下降。因为展开次数太多，那么运算过程中中间变量会很多，而计算机的寄存器个数是固定的，当变量个数超了寄存器，那么变量只能存到栈中从而导致性能下降。

1. 相变内存（PCM）

相变内存，英文全称为“Phase-Change Memory”，通常称为PCM技术或相变RAM技术。之所以被称为相变内存，是因此它利用特殊材料（包括硫、硒或者碲等，目前主要的研究方向是硫属化物玻璃（chalcogenide glass））在晶态和非晶态之间相互转化时所表现出来的导电性差异来存储数据。它是一种非易失性的内存产品。相变内存是下一代内存（闪存）技术。相变内存结合了DRAM内存的高速存取，以及闪存在关闭电源之后保留数据的特性，被业界视为未来闪存和内存的替代品。人们需要一种能够保存数据的DRAM，而相变内存将满足这一需求。相变内存的另外一个优点是，可以在不删除现有数据的情况下写入数据，这比如今的内存更为快捷。根据统计，相变内存的功耗只有现有闪存的一半，但是读写速度可以达到闪存的1000倍。

1. RISC-V开源处理器

RISC-V指令集是基于精简指令集计算(RISC)原理建立的开放指令集架构(ISA)，RISC-V是在指令集不断发展和成熟的基础上建立的全新指令。RISC-V指令集完全开源，设计简单，易于移植Unix系统，模块化设计，完整工具链，同时有大量的开源实现和流片案例，已在社区得到大力支持。它虽然不是第一个开源的的指令集(ISA)，但它很重要，因为它第一个被设计成可以根据具体场景可以选择适合的指令集的指令集架构。基于RISC-V指令集架构可以设计服务器CPU，家用电器cpu，工控cpu和用在比指头小的传感器中的cpu。