# 高级计算机体系结构

1. 冯氏结构五个组成部分：运算器、控制器、存储器、输入设备、输出设备。
2. 多核技术出现原因？

处理器性能=主频\*IPC

(IPC: Instruction Per Clock)

因此提高处理器性能有两个途径：提高主频和提高每个时钟周期内执行的指令数。处理器微架构的变化可以改变IPC，效率更高的微架构可以提高IPC从而提高处理器性能。但是，对于同一代架构，改良架构来提高IPC的幅度非常有限，所以在单核处理器时代通过提高处理器的主频来提高性能就成了唯一手段。不幸的是，给处理器提高主频不是没有止境的，如果通过提高主频来提高处理器性能，就会使处理器的功耗急剧上升，很快就会触及所谓的“频率墙”。使得业界的很多数厂商寻找另一个提高处理器性能的因子，提高IPC。

提高IPC可以通过提高指令执行的并行度来实现，而提高并行度有两种途径：一是提高处理器微架构的并行度；二是采用多核架构。在采用同样的微架构的情况下，为了达到处理器IPC的目的，我们可以采用多核的办法，同时有效地控制功耗急剧上升。

1. 2个1G的多核 VS 2G的单核处理器？

多线程任务情况下，2个1G的多核处理器性能强。

单线程任务情况下，2G的单核处理器性能强。

1. 编程模型：SIMD VS MIMD？

SIMD架构在多条数据上同时执行同一条命令，包括查询、计算和存储信息的命令。一个使用的示例是同时检索多个文件。具有局部内存的处理器以同步的方式使用一条命令处理不同数据，通过多处理器间通信来实现移位分配。

MIMD架构在大量数据集上同时执行多条命令。一个使用示例是执行各种数学运算（如加法，乘法），使用很多分离的计算组件，按照顺序同时解决复杂的数学问题。MIMD可能是也可能不是同步的，比SIMD越来越常见。

SIMD通常用于处理器执行大量计算的问题，这些计算需要处理器并行执行相同命令。MIMD常用于，将复杂算法分割为无关的、独立的部分，每个部分分配一个不同的处理器来并发处理的解决方案。

SIMD和MIMD在技术上的实现也不同。SIMD的处理器通常更简单、更小、更便宜，也比MIMD处理器更快。但是MIMD能够处理复杂的多的操作。使用SIMD模拟实现MIMD操作，会花费更多的时间。SIMD要顺序的执行复杂操作，可是SIMD处理器能够并行的处理。

1. 虚拟化技术的意义？

虚拟化技术降低了开发中的硬件成本，提高了软件开发的效率，让软件开发更便捷，不需要考虑太多硬件层面的制约。

1. 找到一个方案实现越大越快的存储。

1. 局部性原理。

时间局部性：程序结构中的循环。

空间局部性：数组或记录。

1. 存储器

SRAM：集成度较低，功耗较大。特点是工作速度快，只要电源不撤除，写入SRAM的信息就不会消失，不需要刷新电路，同时在读出时不破坏原来存放的信息，一经写入可多次读出。一般用作寄存器或者Cache。

DRAM：集成度较高，功耗也比较低。DRAM是动态随机存储器，需要动态刷新，不然数据就会丢失。一般用作计算机的内存。

命中：处理器需要的数据存放在高层存储器的某个块中。

缺失：在高层存储器中未找到需要的数据。

1. Cache

cache与主存之间的数据交换是以“块”为单位进行的。一个“块”中包含若干个“字”，字长由实际情况确定。习惯上，cache中的“块”称“行”，主存中称“块”。cache的“行”与主存的“块”存储容量相同。

高速缓存，也指基于局部性来管理的存储器。

主存和Cache之间的映射：

1. 全相联映射：

理念：主存中的一块可以拷贝到Cache中的任意行。

主存地址格式：主存块号+块内偏移地址

Cache地址格式：Cache行号+行内偏移地址

Cache标记tag：主存块号

映射过程：

CPU提供一内存地址给cache，cache中的“控制逻辑”将“主存地址格式”中的“主存块号”与cache中所有行的标记tag进行同时比较。如果存在相同的，即表示“命中”，根据“块内偏移地址”找到相应的字。如果不存在相同的，即表示“未命中”，那么将会到主存中寻找。优点：该映射方式下，块冲突的概率低，cache利用率高。缺点：硬件控制复杂，尤其是用于比较“主存块号”与tag时的比较器电路难于设计与实现。适用情况：小容量的cache

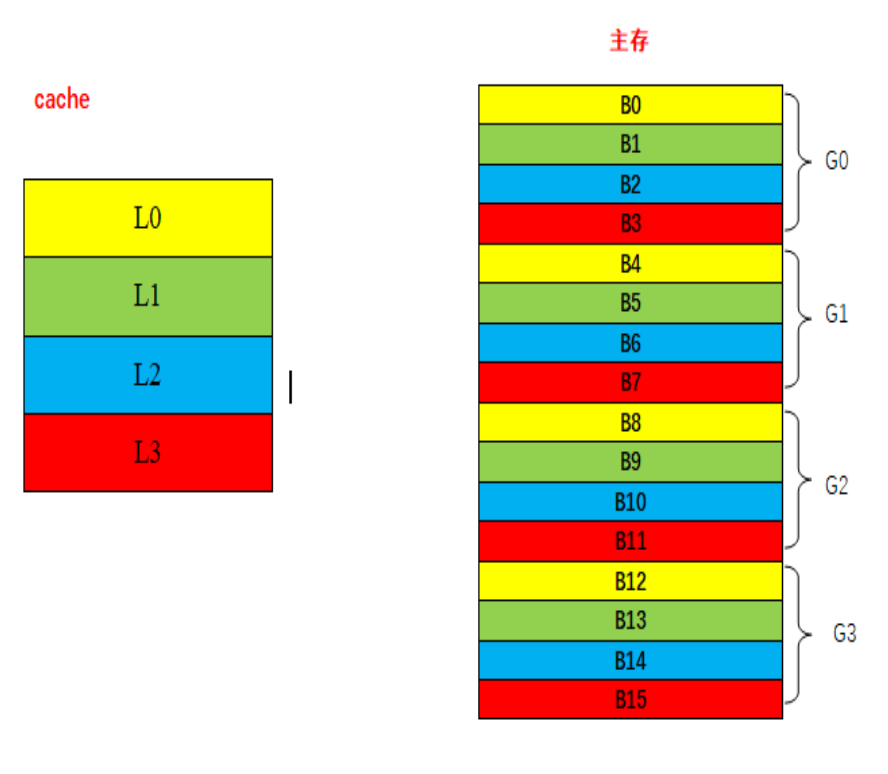
1. 直接映射：

理念：一个主存块只能拷贝到cache的一个特定行位置去。

主存地址格式：主存组号+组内块号+块内偏移地址

cache地址格式：cache行号+行内偏移地址

cache标记tag：映射到该行的主存块的主存地址的“组号”



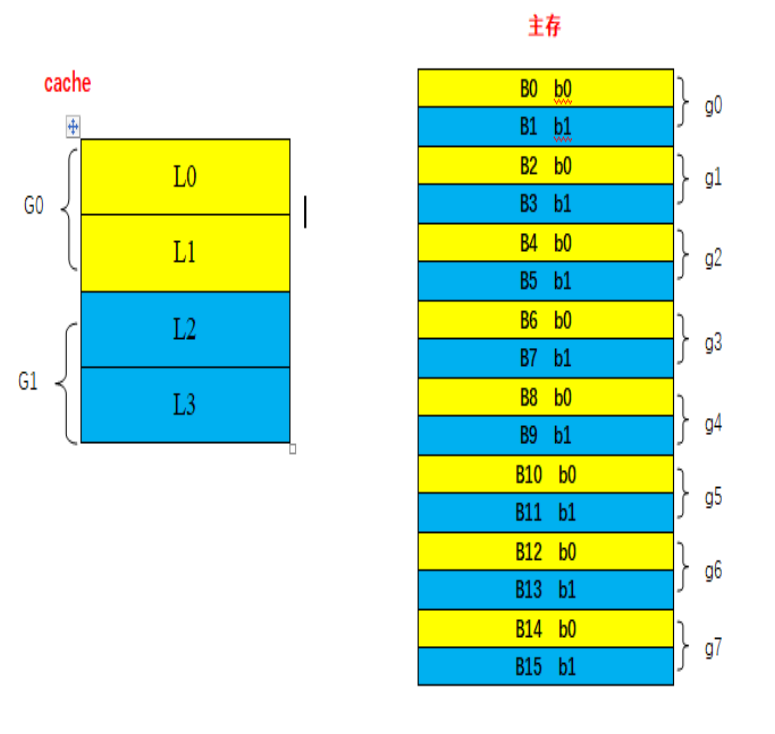
映射过程（地址变换过程）：

CPU提供一内存地址给cache，相关的逻辑根据内存地址中的“组内块号”确定该主存块如果发生拷贝会被拷贝到哪一行；然后，将内存地址中的“主存组号”与上步确定的cache行的标记tag进行比较，如果存在相同的即“命中”，如果不存在相同的即“未命中”。优点：硬件简单，容易实现，成本低。缺点：发生块冲突的概率较大，导致cache的命中率、效率下降。适用情况：大容量的cache，更多的行数可以减小冲突发生的机会。

1. 组相联映射：

理念：是直接映射方式与全相联映射方式的折衷方案，适度地兼顾了二者的优点又避免二者的缺点。将cache的空间分为若干组，主存块与cache组之间直接映射，而组内各块之间全相联映射。

关键：cache的分组数=主存每一组的块数



主存地址格式：主存组号+组内块号+块内偏移地址

cache地址格式：cache组号+组内行号+行内偏移地址

cache标记tag：组号

映射过程（地址变换过程）：CPU提供一内存地址给cache，相关逻辑根据地址中的“组内块号”部分确定主存块如果发生拷贝将会被放置到cache的哪一组中；然后，将地址中的“主存组号”与上步所确定的那一组中所有行的tag同时进行比较，如果存在相同的即“命中”，如果不存在相同的即为“未命中”。该映射方式实现较为容易，块冲突概率比直接映射方式低，命中率介于直接映射方式与全相联映射方式之间。被普遍采用。

1. LRU

最近最少使用法：被替代的块是最久没有使用的那一块。

在内存满的时候，淘汰那些不常用的数据，空出空间存储新的数据，这时可以用LRU。既然是淘汰最近最少使用的数据，姑且就可以理解为，当内存满了的那个时刻，内存中，哪些数据最后一次被访问的时间最小， 不表示哪些数据是冷数据，应该被移除。假如每条数据有一个属性lasttime，用来记录被访问时刻的时间，这样，每一条数据都有一个最后访问时间， 当内存满的时候，遍历所有元素，删除最后访问时间最小的那个元素。

1. 6种基本的Cache优化方法：
2. 更大的块
3. 更大的Cache容量
4. 更高的相联度
5. 更多级Cache
6. 读缺失优先级更高
7. 缓存索引避免地址转换
8. 流水线

计算机中的流水线是把一个重复的过程分解为若干个子过程，每个子过程与其他子过程并行进行。由于这种工作方式与工厂中的生产流水线十分相似， 因此称为流水线技术

从本质上讲，流水线技术是一种时间并行技术。

过程：Fetch取指令——Dec译码——Exec执行——MEM访存——WB写回

流水线冒险：

在流水线当中我们希望每一个时钟周期都有一条指令进入流水线开始执行，但是在某些情况下，下一条指令无法按照预期开始执行，这种情况就被称为冒险。

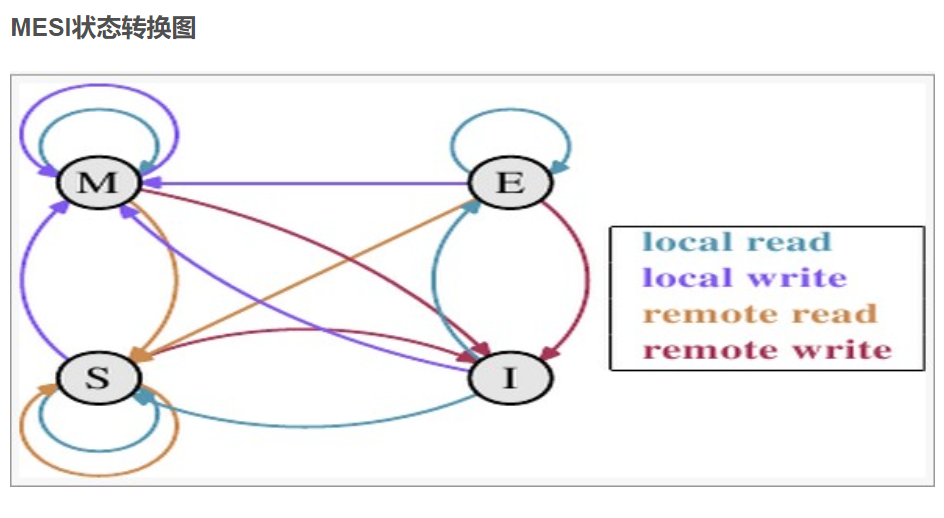
流水线三种冒险：

1. 结构冒险：所需的硬件部件正在为之前的指令工作
2. 数据冒险：需要等待之前的指令完成数据的读写
3. 控制冒险：需要根据之前指令的结果决定下一步行为

数据冒险解决：

1. 硬件阻塞
2. 软件插入“NOP”指令
3. 编译优化：调整指令顺序
4. 合理实现寄存器堆的读/写操作
5. 转发技术
6. MESI协议

M:modified被修改E:exclusive独享的 S:shared共享的 I:invalid无效的



在多核cpu中，每个cpu有自己的缓存，这些缓存又可以相互共享，该协议就是说明某缓存在本cpu读写（local）和其他cpu读写（remote）时，状态的转换。

1. Amdahl定律

在计算机体系架构设计过程中，某个部件的优化对整个框架的优化和改善是有上限的。