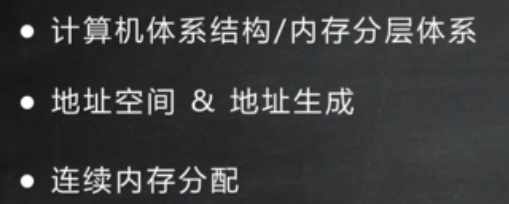
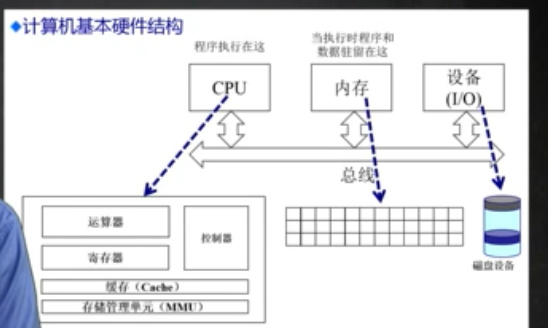
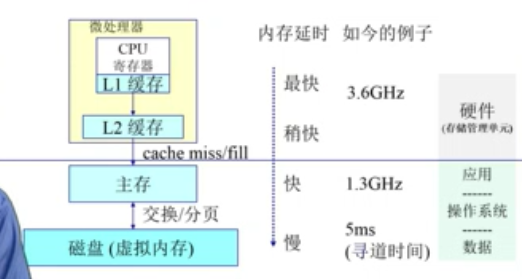
操作系统如何管理物理内存？

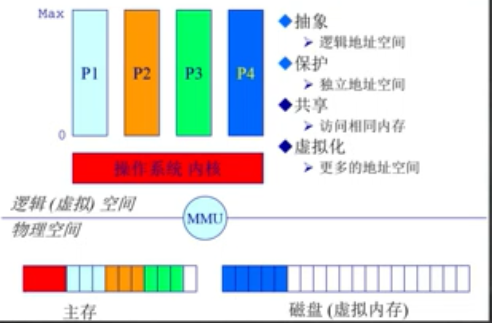






越靠近CPU的存储越快，但也越小

操作系统要完成的目标分以下4类：



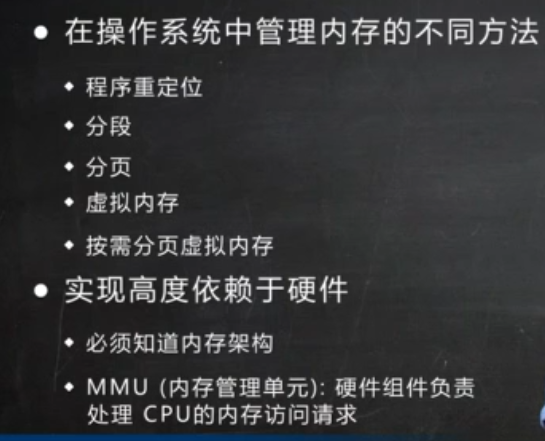
抽象：使应用程序不用考虑过多的底层的细节，只需要可以访问一段连续的地址空间，既逻辑地址空间；

保护：不同应用程序之间的地址空间的隔离措施

共享：进程之间的交互，使数据安全可靠有效的传递

虚拟化：避免内存不够的情况，把需要用到的必须的数据放到内存中，把应用程序不需要的数据暂时放到硬盘中；

MMU：内存管理单元



地址空间&地址生成？（程序运行的地方）

地址空间定义：

物理地址空间：硬件支持的地址空间，主存、磁盘等

逻辑地址空间：一个运行的程序所拥有的内存范围

逻辑地址空间和物理地址空间的映射关系需要OS的协调

地址生成：

CPU：

1、运算器需要在逻辑地址的内存内容；

2、内存管理单元寻找在逻辑地址和物理地址之间的映射；

3、控制器从总线发送在物理地址的内存内容的请求；

内存：

4、内存发送物理地址内存的内容给CPU

操作系统：

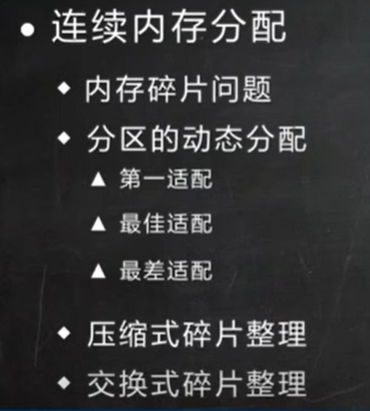
建立逻辑地址和物理地址之间的映射

.c file—编译—>.s file—汇编—>.o file—链接—>.exe file—载入—>程序在内存中重定位运行

内存管理单元MMU：逻辑地址LA和物理地址PA之间的映射

地址安全检查：

操作系统设置逻辑地址空间的基址和界限



内存碎片问题：空闲内存不能被利用；

外部碎片：在分配单元间的未使用内存；

内部碎片：在分配单元中的未使用内存；

分区的动态分配

一个简单的内存管理方法：

当一个程序准许运行在内存中时，分配一个连续的区间；

分配一个连续的内存区间给运行的程序以访问数据；

分配策略：首次适配、最优适配、最差适配

首次适配：

从0地址开始找，将第一个碰到的可以用的足够的内存块就分配给应用程序，不管大小；

基本原理&实现：

简单实现；

从0地址开始，按地址排序的空闲块列表；

分配需要寻找一个合适的分区；

重分配需要检查，看是否自由分区能合并于相邻的空闲分区（若有）；

优势：

简单；

易于产生更大空闲块，向着地址空间的结尾，不用干扰到后面的大空间块（若有）

劣势：

外部碎片；（容易产生小碎片）

不确定性；（不能总是在相同的内存块上进行）

最优适配：

从0地址开始找，找到相差最小的空间块大小作为目标内存块；

为了避免分割大空闲块、为了最小化外部碎片产生的尺寸

需求：

按尺寸排列的空闲块列表；

分配需要寻找一个合适的分区；

重分配需要搜索及合并于相邻的空闲分区（若有）

优势：

当大部分分配最小尺寸时非常有效；

比较简单；

劣势：

外部碎片；

重分配慢；

易产生很多没用的微小碎片（不太好）

最差适配：

从0地址开始找，找到相差最大的空间块大小作为目标内存块；

为了避免有太多微小的碎片；

需求：

按尺寸排列的空闲块列表；

分配很快（获得最大的分区）；

重分配需要合并于相邻的空闲分区（若有）然后调整空闲块列表

优势：假如分配是中等尺寸效果最好；

劣势：重分配慢；外部碎片；易破碎大空闲块以致大分区无法被分配；

压缩式碎片整理

重置程序以合并孔洞；

要求所有程序是可以动态可重置的；

议题：何时重置？开销

交换式碎片整理

运行程序需要更多的内存；

抢占等待的程序&回收它的内存；

利用硬盘充当虚拟内存的作用；

将空闲的程序数据暂时放在磁盘上