Lesson 12
Virtual Memory

ICS Seminar #9

张龄心

Dec 6, 2023

地址空间

- 回顾: 存储器层次结构
- 地址空间: 非负整数地址的有序集合
- 使用虚拟地址空间的好处
  - 简化链接和加载
  - 内存保护的工具(许可位)
  - 共享内存
  - 简单的物理内存分配

- *计量单位: KB MB GB TB PB EB

物理页/虚拟页

- 物理页PP和虚拟页VP
  - 为了方便管理, 把物理内存(DRAM)和虚拟内存划分为若干页
- 物理地址空间和虚拟地址空间
  - 一个物理页: 主存(DRAM)中的一块.
  - 一个虚拟页: 虚拟地址空间中的一块.
    - 可能状态: 未分配 / 已分配未缓存 / 已分配已缓存
    - 未分配/已分配: 这个虚拟页是否对应磁盘中的一块?
    - 未缓存/已缓存: 这个虚拟页是否被缓存在主存(DRAM)中了?

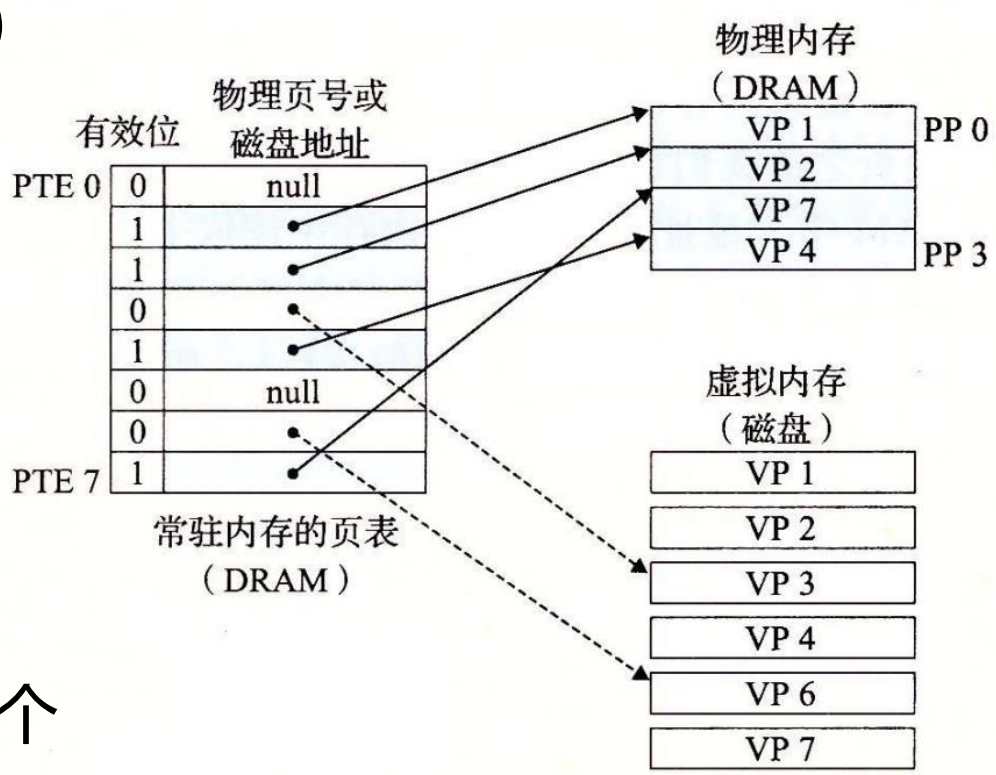
页表(PT)

- 页表PT: PTE的数组
  - PTE:页表条目(PTE / page table entry)
  - 若虚拟地址空间大小为2^m,

    每个页大小为2^n,

    则需要2^(m-n)个页表条目
- 页命中 / 页不命中: 该页是否缓存在主存中?
- 缺页: 页不命中.
  - 处理: 从主存/物理内存DRAM中选择一个

    页驱逐, 然后换入需要的页
- 结合高速缓存的知识: 页替换算法? 局部性? 抖动?

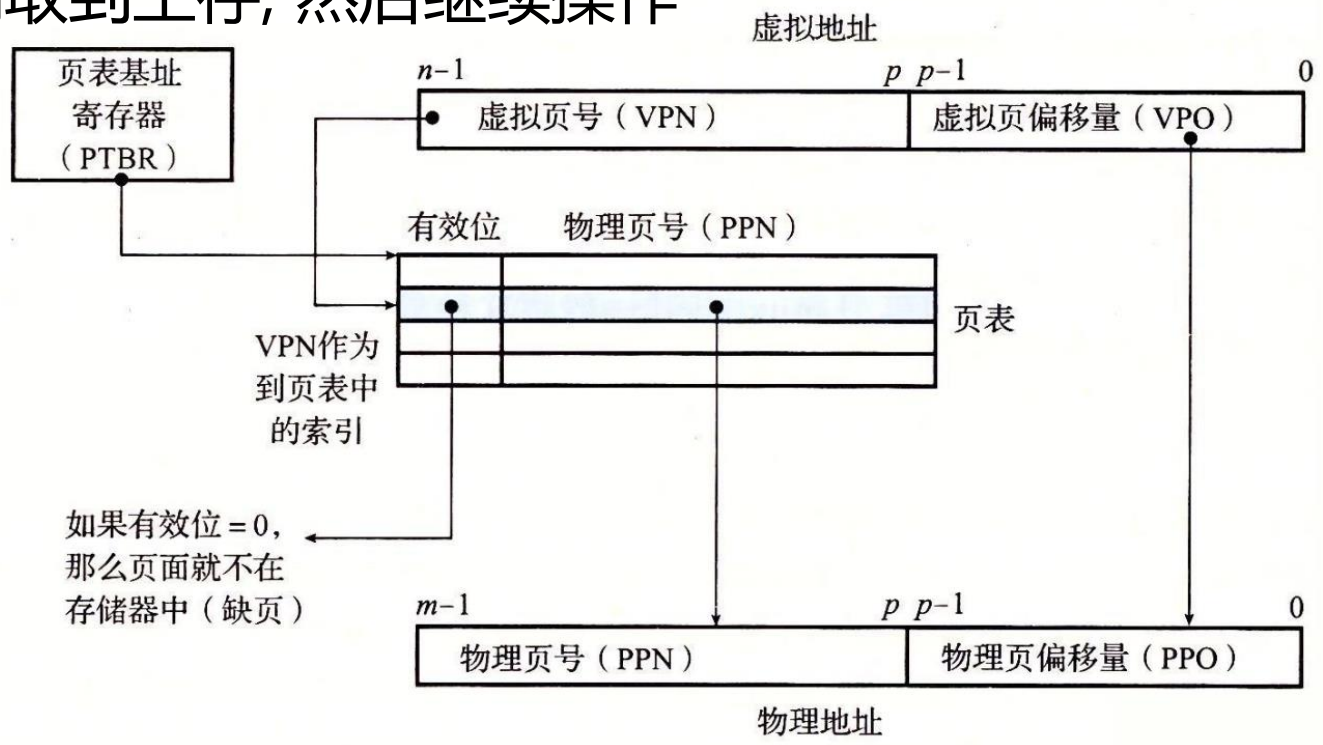


地址翻译

- 地址翻译(MMU内存管理单元): 物理地址和虚拟地址如何互相转化?
  - 将物理页号PPN转换成虚拟页号VPN, 然后拼接VPO (=PPO)

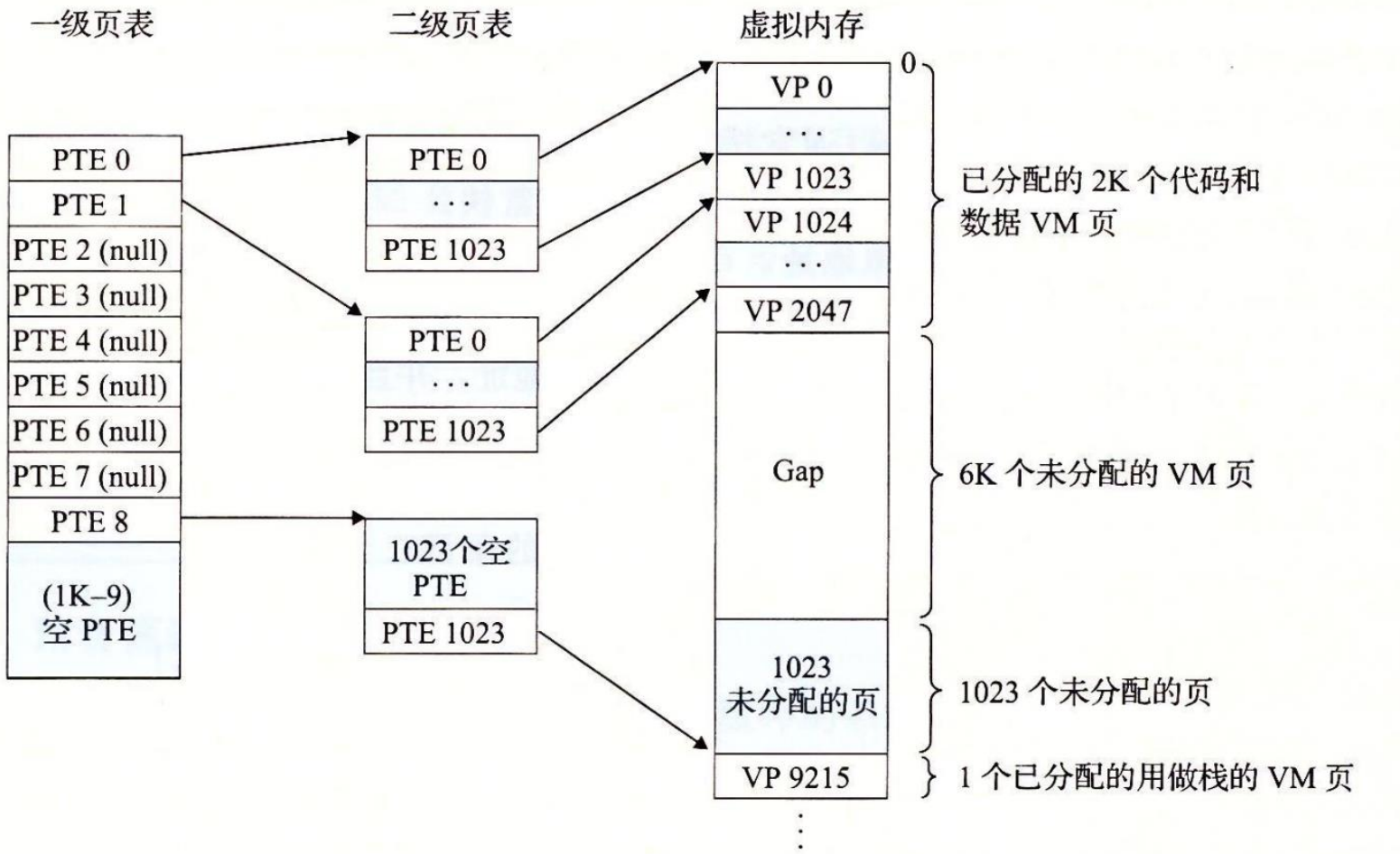
    *如何将PPN转换成VPN? 查页表
  - 如果遇到缺页: 把页从磁盘调取到主存, 然后继续操作
- 注意记忆各种缩写
- 加速翻译: 快表TLB (缓存常用PTE)
  - 保存常用的PPN/VPN转换,

    这样就不用每次都查页表了
  - 先查TLB, 找不到再去查页表



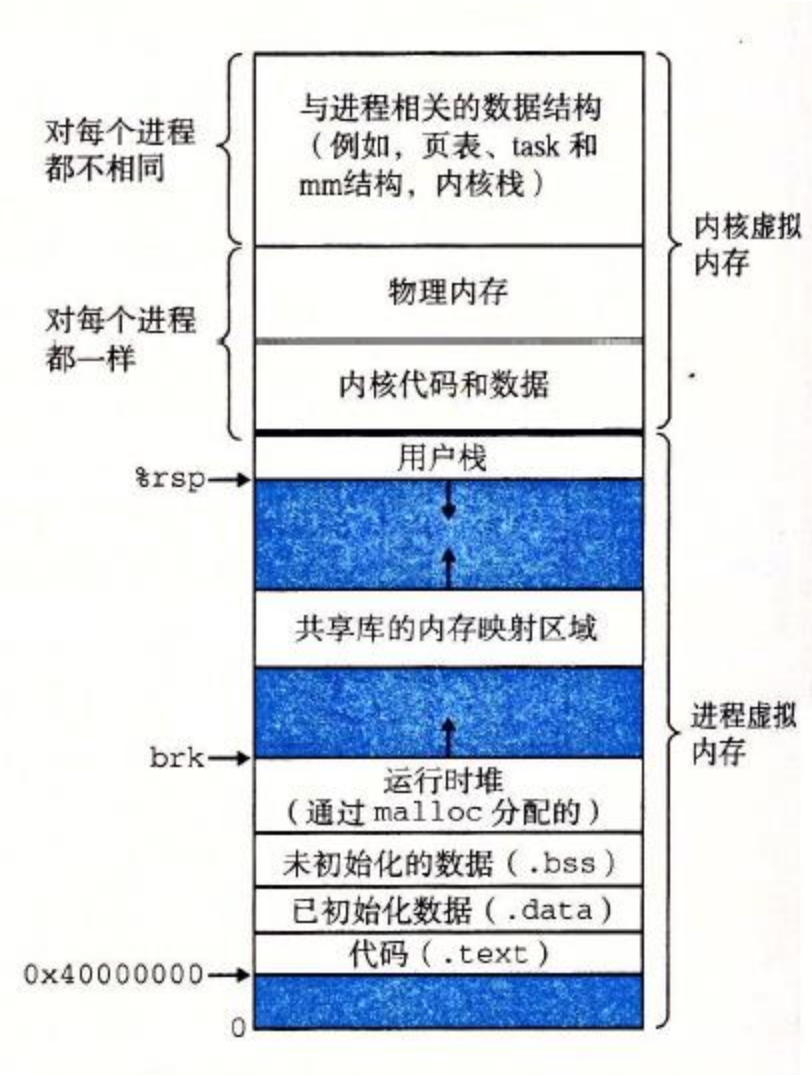
地址翻译

- 多级页表: 地址较长时使用
  - 多做题



虚拟内存

- Intel Core i7
  - 第一级页表起始位置: CR3寄存器
  - 记忆: P578
    - 不同层次页表的结构
    - 相关权限位&故障
- Linux虚拟内存系统
  - 每个进程一个单独的虚拟地址空间
  - 一些物理页面可以同时被映射到多个进程
- 内存映射/共享对象/fork等
  - 结合第7章链接部分. 如果页面相同, 多个进程可以共享物理页



动态内存分配

- 为什么要引进动态内存分配? 只有栈不好吗?
  - 最根本原因: **对象的生命周期，可能比创建者的生命周期要长**
    - *C语言中3种类型的生命周期:

      自动（栈中的对象）、动态（堆中的对象）、静态（.data .bss）
  - 方便支持变长对象
    - 当然可以被放在变长栈帧中，但不高效、不安全
  - 较大对象可能会造成栈溢出
- 堆的管理
  - 手动管理：C
  - 运行时管理（垃圾收集器）：JVM, Go, Python
  - 编译时管理：Rust

动态内存分配

- 动态内存分配的目标:
  - 最大化吞吐率 + 最大化内存利用率 (最小化碎片)
- 碎片
    - 内部碎片: 已分配块>有效载荷导致的
    - 外部碎片: 空闲块加起来足够大, 但是位置分散, 故不能使用
- 隐式空闲链表 / 显式链表 / 分离链表
- 首次适配 / 下一次适配 / 最佳适配
  - 利用率: best fit > first fit > next fit
  - 速度: next fit > first fit > best fit
- 伙伴系统 (2^n型的块)

动态内存分配

- Malloclab的建议: **不要试图一步到位**
  - 建议:

    隐式空闲链表 -> 调通了 -> 显式空闲链表 -> 调通了 -> 分离链表 -> 调通了 -> 其他优化
  - **每次加入新功能, 都要调通了(确保正确性), 再进行下一步**

- 垃圾收集 (C语言保守Mark & Sweep)
- 内存常见错误(C语言是一种相当不安全的语言)

例

12、进程 P1 通过 fork()函数产生一个子进程 P2。假设执行 fork()函数之前，进程 P1 占用了 53 个（用户态的）物理页，则 fork 函数之后，进程 P1 和进程 P2 共占用_____个（用户态的）物理页；假设执行 fork()函数之前进程 P1 中有一个可读写的物理页，则执行 fork()函数之后，进程 P1 对该物理页的页表项权限为_____。上述两个空格对应内容应该是（　　）

A．53，读写　B．53，只读　C．106，读写　D．106，只读

例

12、进程 P1 通过 fork()函数产生一个子进程 P2。假设执行 fork()函数之前，进程 P1 占用了 53 个（用户态的）物理页，则 fork 函数之后，进程 P1 和进程 P2 共占用_____个（用户态的）物理页；假设执行 fork()函数之前进程 P1 中有一个可读写的物理页，则执行 fork()函数之后，进程 P1 对该物理页的页表项权限为_____。上述两个空格对应内容应该是（　　）

A．53，读写　B．53，只读　C．106，读写　D．106，只读

B (fork之后双方共享这些物理页, 标记为只读; 如果有一个进程要修改这些页面, 则触发故障, 然后进行写时复制,)

Thank you!