操作系统

Operating System

非连续分配的设计目标

- 连续分配的缺点
  - 分配给程序的物理内存必须连续
  - 存在外碎片和内碎片
  - 内存分配的动态修改困难
  - 内存利用率较低
- 非连续分配的设计目标：提高内存利用效率和管理灵活性
  - 允许一个程序的使用非连续的物理地址空间
  - 允许共享代码与数据
  - 支持动态加载和动态链接

非连续内存分配的实现

- 非连续分配需要解决的问题
  - 如何实现虚拟地址和物理地址的转换
    - 软件实现 （灵活，开销大）
    - 硬件实现 （够用，开销小）
- 非连续分配的硬件辅助机制
  - 如何选择非连续分配中的内存分块大小
    - 段式存储管理 （segmentation）
    - 页式存储管理 （paging）

操作系统

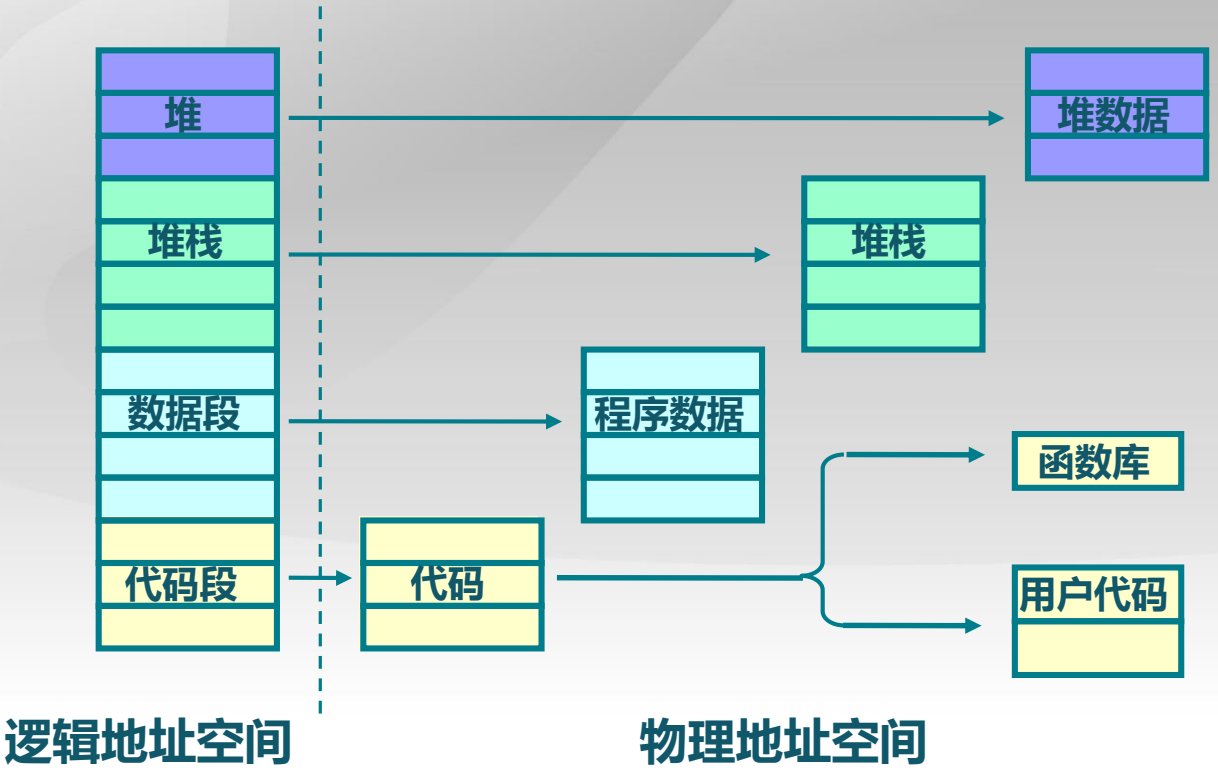
Operating System

段地址空间

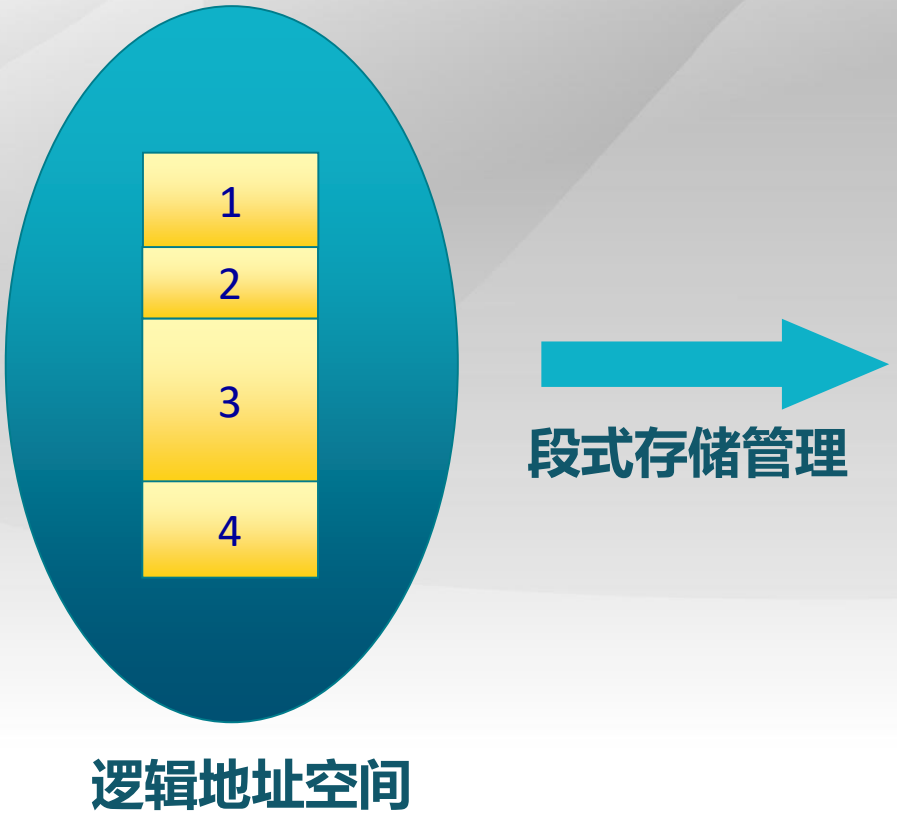
- 进程的段地址空间由多个段组成
  - 主代码段
  - 子模块代码段
  - 公用库代码段
  - 堆栈段(stack)
  - 堆数据(heap)
  - 初始化数据段
  - 符号表等
- 段式存储管理的目的
  更细粒度和灵活的分离与共享

逻辑地址空间

段式地址空间的不连续二维结构

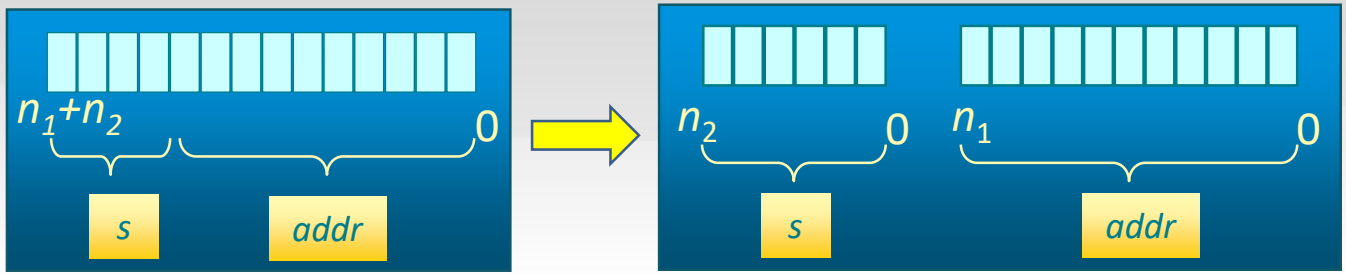


段地址空间的逻辑视图



段访问机制

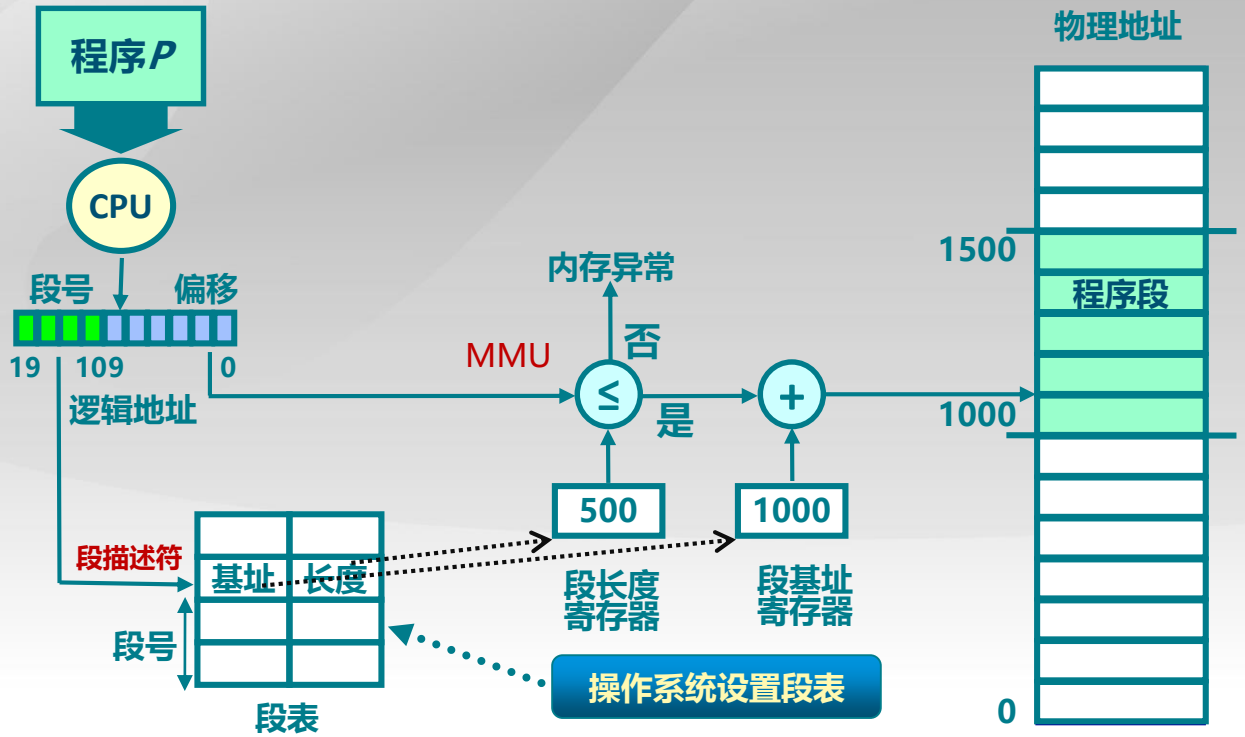
- 段的概念
  - 段表示访问方式和存储数据等属性相同的一段地址空间
  - 对应一个连续的内存“块”
  - 若干个段组成进程逻辑地址空间
- 段访问：逻辑地址由二元组(s, addr)表示
  - s — 段号
  - addr — 段内偏移



单地址实现方案

“段基址+段内偏移”实现方案

段访问的硬件实现


程序P
CPU
段号
偏移
19
109
0
逻辑地址
MMU
内存异常
否
是
≤
+
500
1000
段长度寄存器
段基址寄存器
段描述符
基址
长度
段号
段表
操作系统设置段表
物理地址
1500
程序段
1000
0

操作系统

Operating System

页式存储管理

- 页帧（帧、物理页面, Frame, Page Frame）
  - 把物理地址空间划分为大小相同的基本分配单位
  - 2的n次方，如512, 4096, 8192
- 页面（页、逻辑页面, Page）
  - 把逻辑地址空间也划分为相同大小的基本分配单位
  - 帧和页的大小必须是相同的
- 页面到页帧
  - 逻辑地址到物理地址的转换
  - 页表
  - MMU/TLB

帧 (Frame)

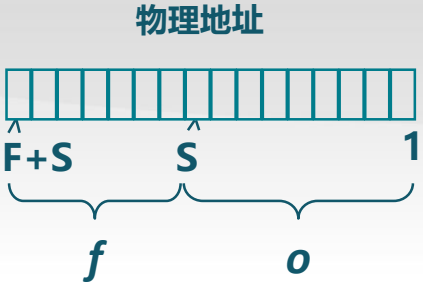
- 物理内存被划分成大小相等的帧

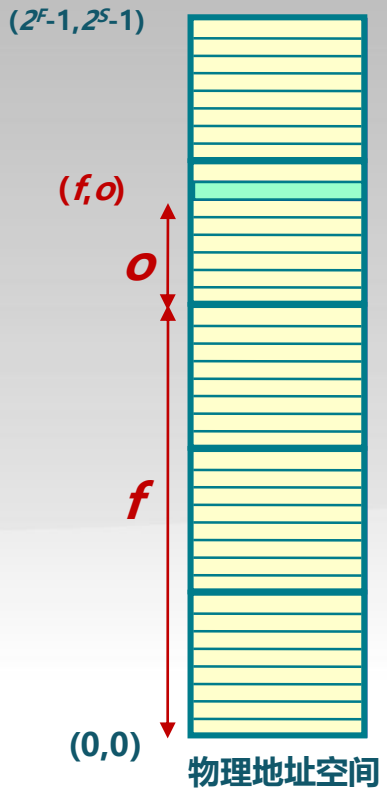
  内存物理地址的表示：二元组 (f, o)

  f — 帧号 (F 位, 共有2^F 个帧)

  o — 帧内偏移 (S 位, 每帧有2^S 字节)

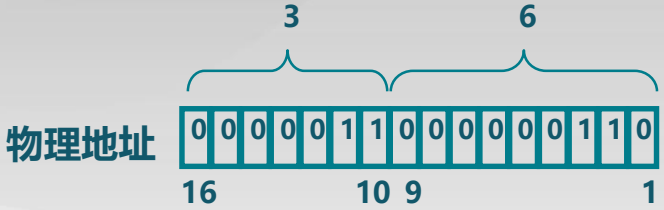
  物理地址 = $f * 2^S + o$





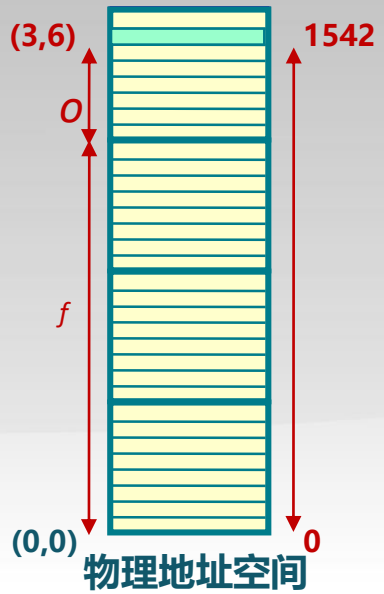
基于页帧的物理地址计算实例

- 假定
  - 16-bit的地址空间
  - 9-bit (512 byte) 大小的页帧
- 物理地址计算
  - 物理地址表示 = (3, 6)



物理地址 = 2^S*f + o

F=7 S=9 f=3 o=6



- 实际物理地址 =29 *3+ 6=1536+6=1542

页(Page)

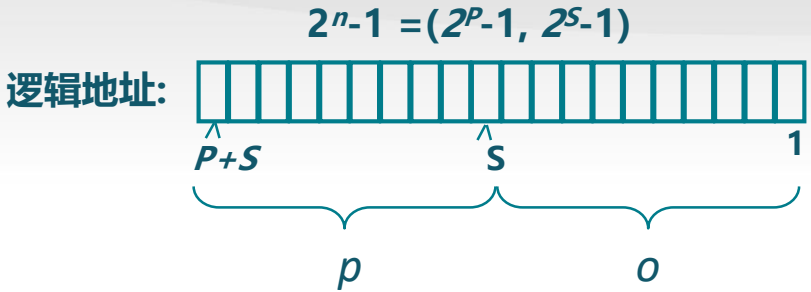
- 进程逻辑地址空间被划分为大小相等的页
  - 页内偏移 = 帧内偏移
  - 通常：页号大小 ≠ 帧号大小

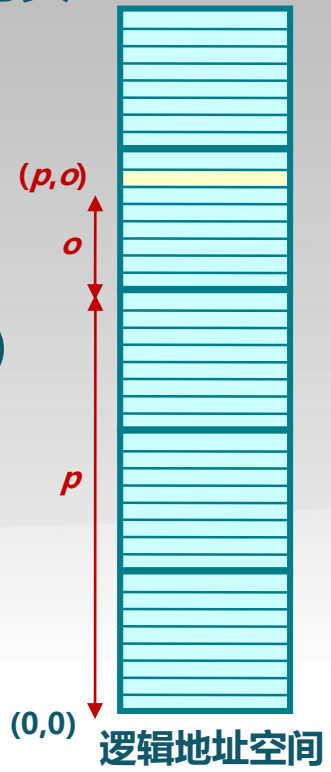
进程逻辑地址的表示：二元组 (p, o)

p — 页号 (P 位, 2^P 个页)

o — 页内偏移 (S 位, 每页有2^S 字节)

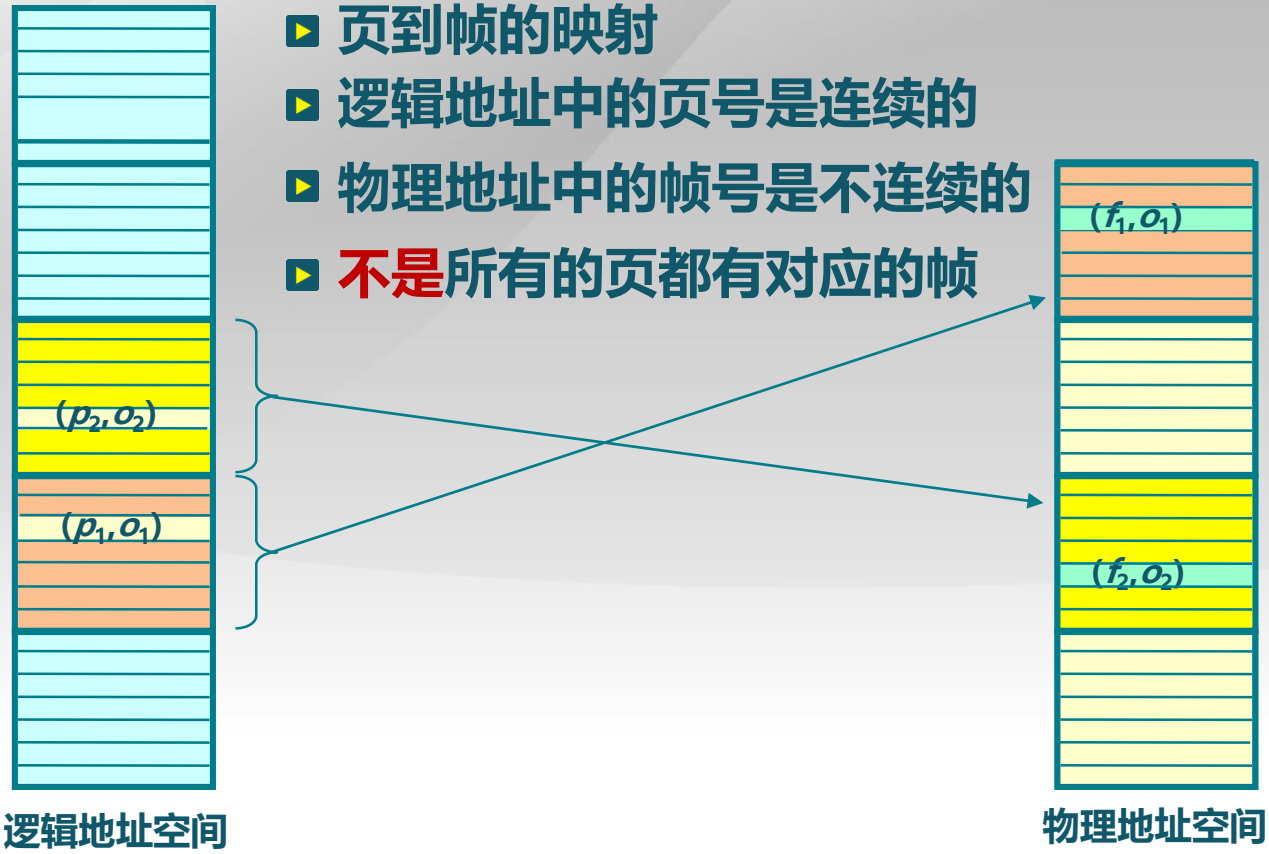
虚拟地址 = $p * 2^S + o$



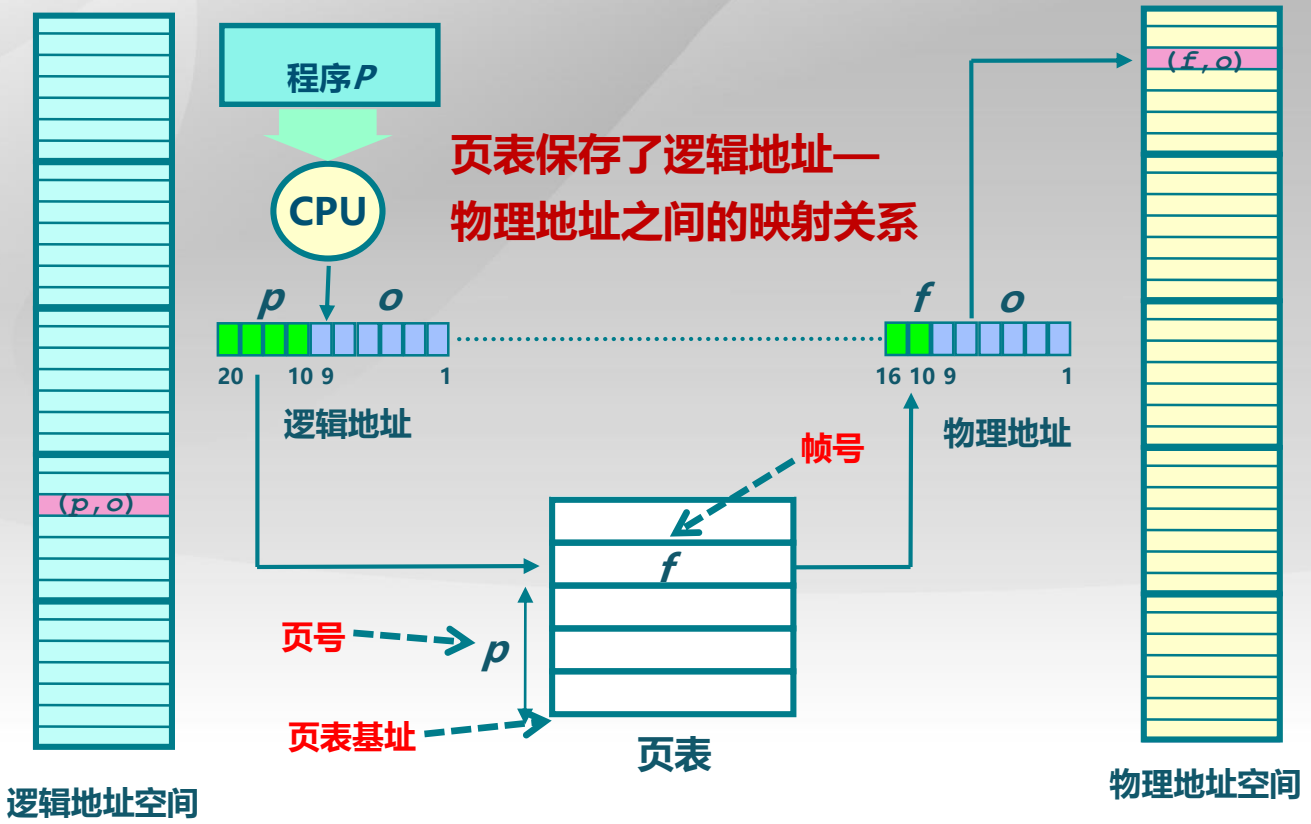


页式存储中的地址映射

- 页到帧的映射
- 逻辑地址中的页号是连续的
- 物理地址中的帧号是不连续的
- **不是**所有的页都有对应的帧



页表


程序P
CPU
页表保存了逻辑地址—
物理地址之间的映射关系
p
o
f
o
20
10 9
1
16 10 9
1
逻辑地址
物理地址
帧号
(p,o)
f
页号
p
页表基址
页表
(f,o)
逻辑地址空间
物理地址空间

操作系统

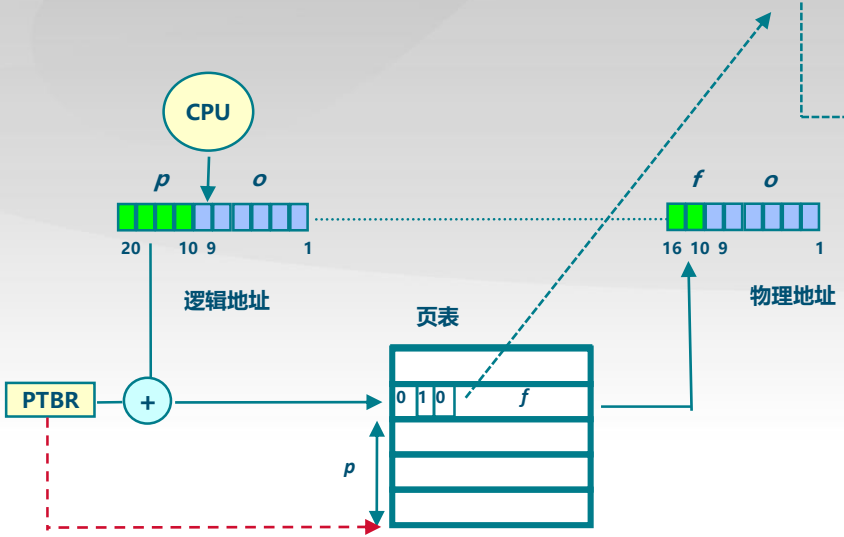
Operating System

页表结构

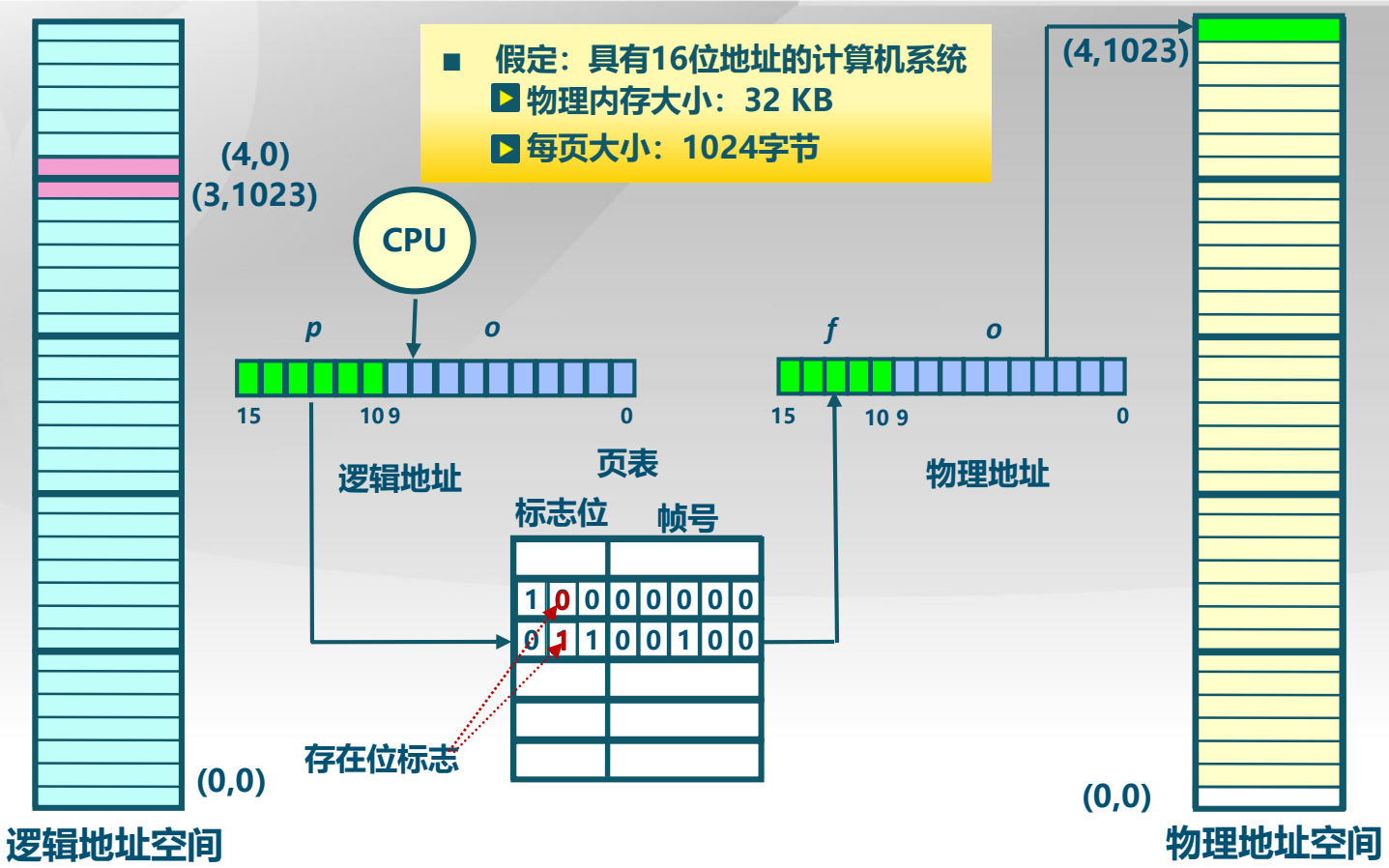
- 每个进程都有一个页表
  - 每个页面对应一个页表项
  - 随进程运行状态而动态变化
  - 页表基址寄存器(PTBR: Page Table Base Register)

页表项组成

- 帧号：f
- 页表项标志
  - 存在位(resident bit)
  - 修改位(dirty bit)
  - 引用位(clock/reference bit)



页表地址转换实例


假定：具有16位地址的计算机系统
物理内存大小：32 KB
每页大小：1024字节
(4,0)
(3,1023)
CPU
p
o
15
10 9
0
逻辑地址
页表
标志位
帧号
1 0 0 0 0 0 0 0
0 1 1 0 0 1 0 0
存在位标志
(0,0)
逻辑地址空间
f
o
15
10 9
0
物理地址
(4,1023)
(0,0)
物理地址空间

页式存储管理机制的性能问题

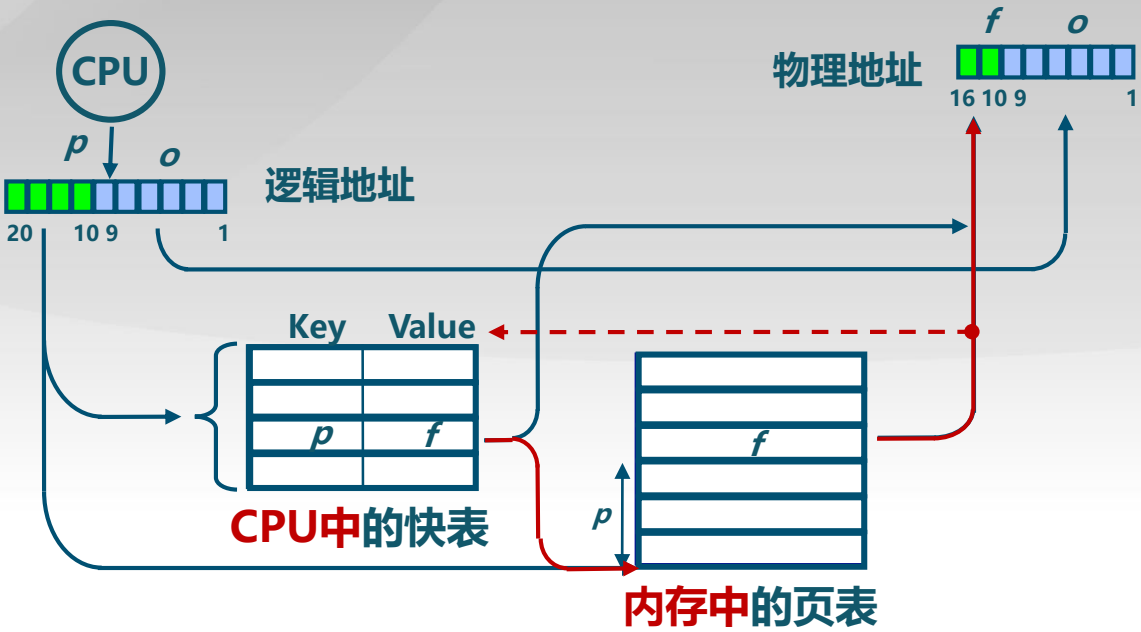
- 内存访问性能问题
  - 访问一个内存单元需要2次内存访问
  - 第一次访问：获取页表项
  - 第二次访问：访问数据
- 页表大小问题：
  - 页表可能非常大
  - 64位机器如果每页1024字节，那么一个页表的大小会是多少?
- 如何处理?
  - 缓存（Caching）
  - 间接（Indirection）访问

操作系统

Operating System

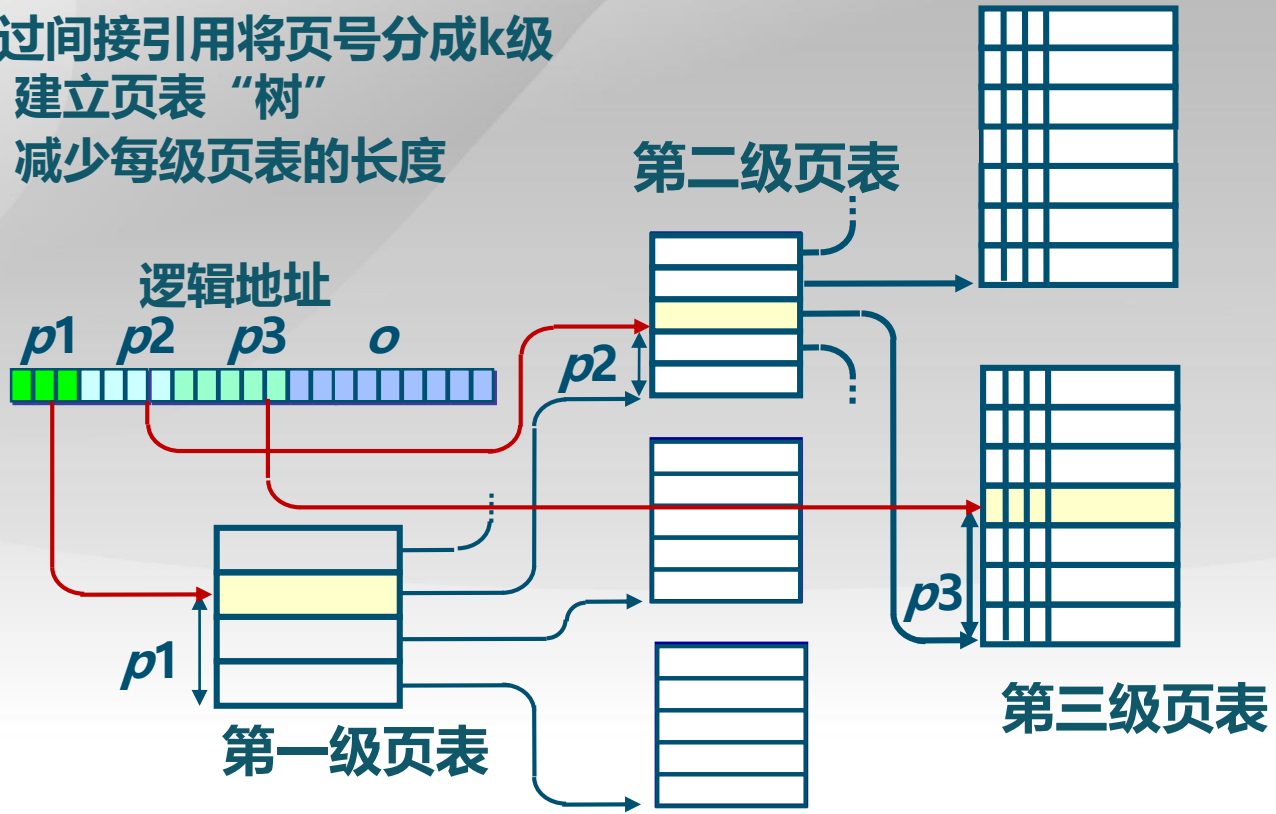
快表(Translation Look-aside Buffer, TLB)

- 缓存近期访问的页表项
  - TLB 使用关联存储(associative memory)实现，具备快速访问性能
  - 如果TLB命中，物理页号可以很快被获取
  - 如果TLB未命中，对应的表项被更新到TLB中

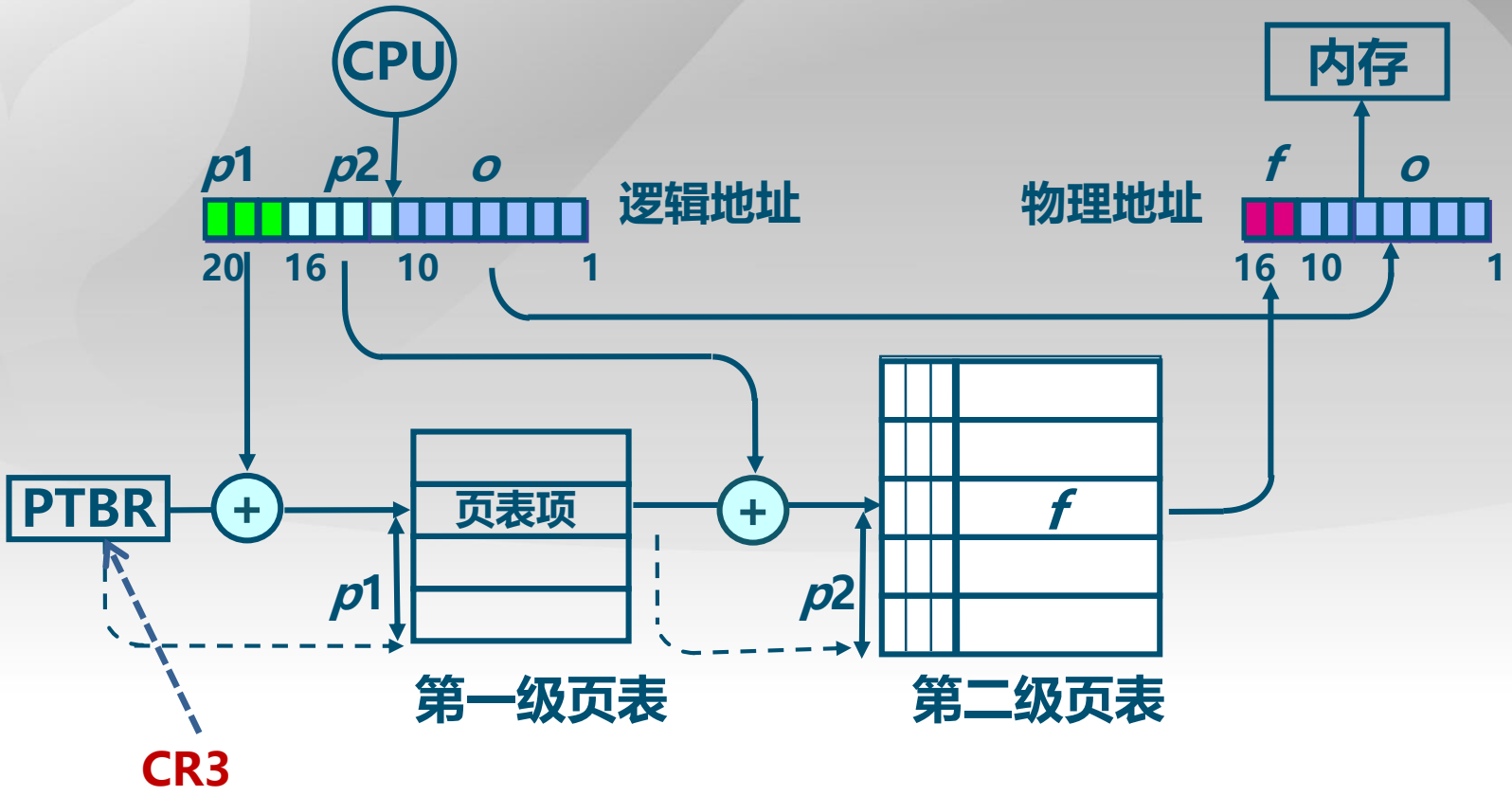


多级页表

- 通过间接引用将页号分成k级
  - 建立页表“树”
  - 减少每级页表的长度



二级页表实例


CPU
内存
p1
p2
o
逻辑地址
物理地址
f
o
20
16
10
1
16
10
1
PTBR
+
+
页表项
f
p1
p2
第一级页表
第二级页表
CR3

操作系统

Operating System

大地址空间问题

- 对于大地址空间(64-bits)系统，多级页表变得繁琐.
  - 比如：5 级页表
  - 逻辑 (虚拟) 地址空间增长速度快于物理地址空间
- 页寄存器和反置页面的思路
  - 不让页表与逻辑地址空间的大小相对应
  - 让页表与物理地址空间的大小相对应

页寄存器(Page Registers)

- 每个帧与一个页寄存器(Page Register)关联，寄存器内容包括:
  - 使用位(Residence bit): 此帧是否被进程占用
  - 占用页号(Occupier): 对应的页号p
  - 保护位(Protection bits)
- 页寄存器示例
  - 物理内存大小: 4096*4096=4K*4KB=16 MB
  - 页面大小: 4096 bytes = 4KB
  - 页帧数: 4096 = 4K
  - 页寄存器使用的空间 (假设每个页寄存器占8字节):
    - 8*4096=32 Kbytes
  - 页寄存器带来的额外开销:
    - 32K/16M = 0.2% (大约)
  - 虚拟内存的大小： 任意

页寄存器方案的特征

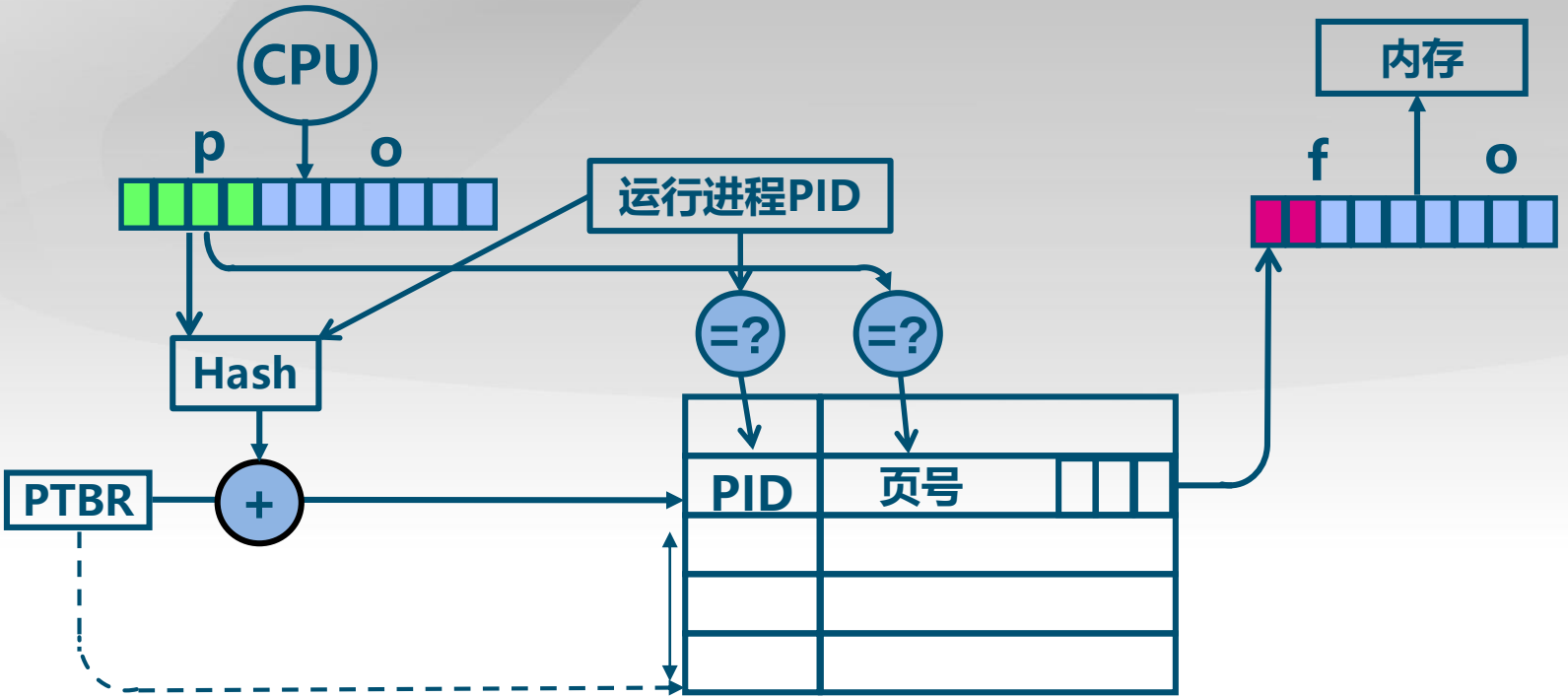
- 优点:
  - 页表大小相对于物理内存而言很小
  - 页表大小与逻辑地址空间大小无关
- 缺点:
  - 页表信息对调后，需要依据帧号可找页号
  - 在页寄存器中搜索逻辑地址中的页号

页寄存器中的地址转换

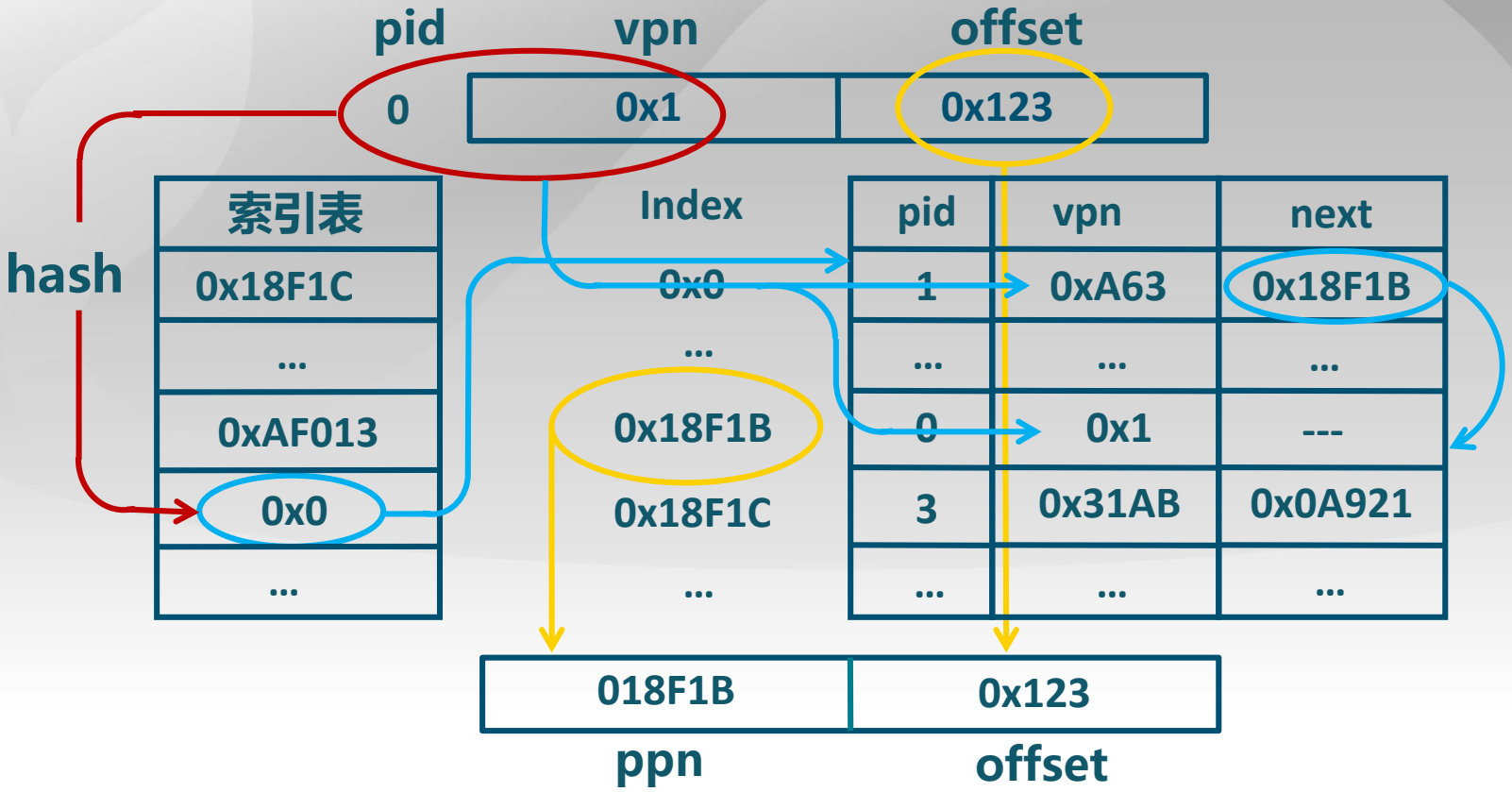
- CPU生成的逻辑地址如何找对应的物理地址?
  - 对逻辑地址进行Hash映射，以减少搜索范围
  - 需要解决可能的冲突
- 用快表缓存页表项后的页寄存器搜索步骤
  - 对逻辑地址进行Hash变换
  - 在快表中查找对应页表项
  - 有冲突时遍历冲突项链表
  - 查找失败时，产生异常
- 快表的限制
  - 快表的容量限制
  - 快表的功耗限制(StrongARM上快表功耗占27%)

反置页表

- 基于Hash映射值查找对应页表项中的帧号
  - 进程标识与页号的Hash值可能有冲突
  - 页表项中包括保护位、修改位、访问位和存在位等标识



反置页表的Hash冲突



操作系统

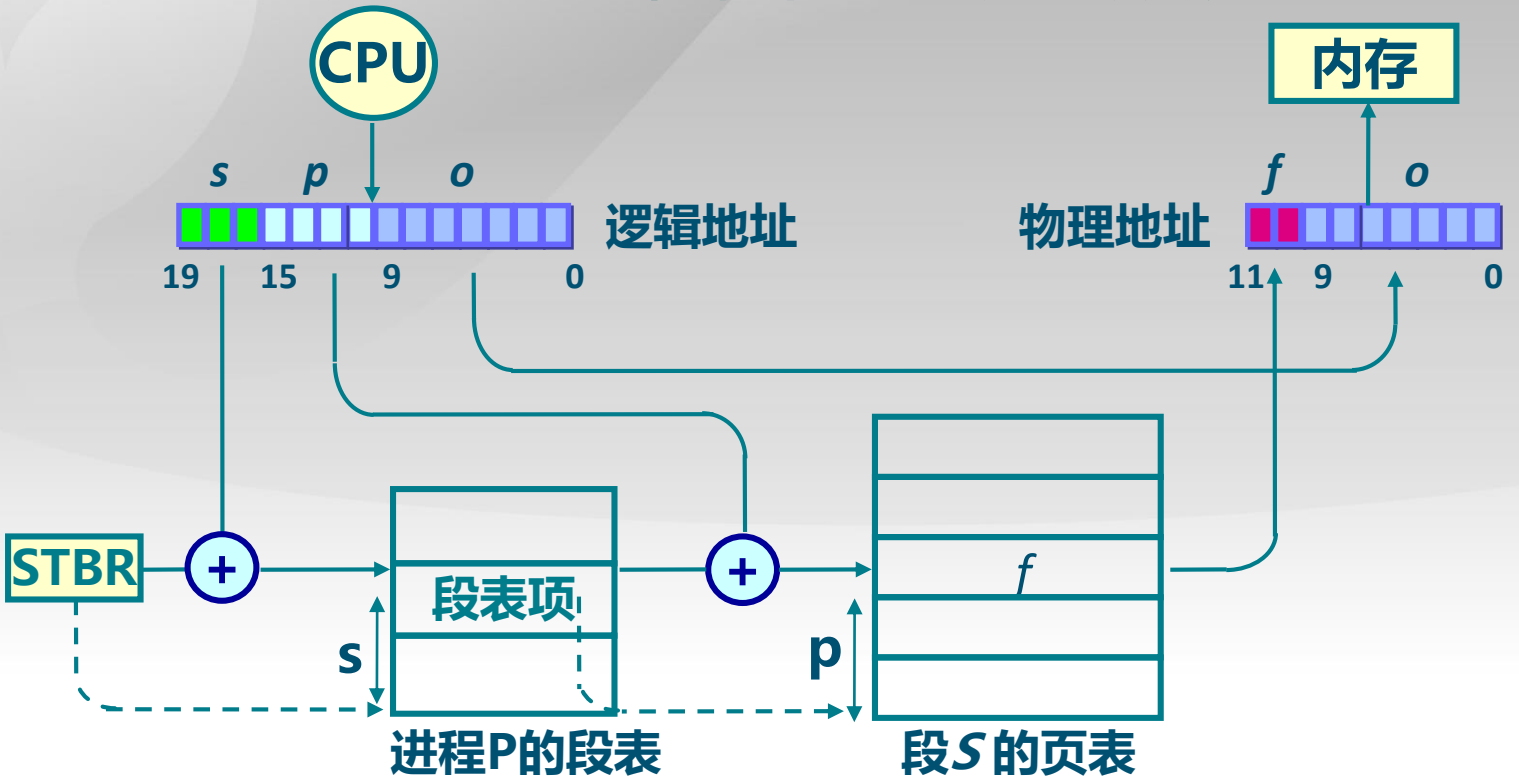
Operating System

段页式存储管理的需求

- 段式存储在内存保护方面有优势，页式存储在内存利用和优化转移到后备存储方面有优势。
- 段式存储、页式存储能否结合?

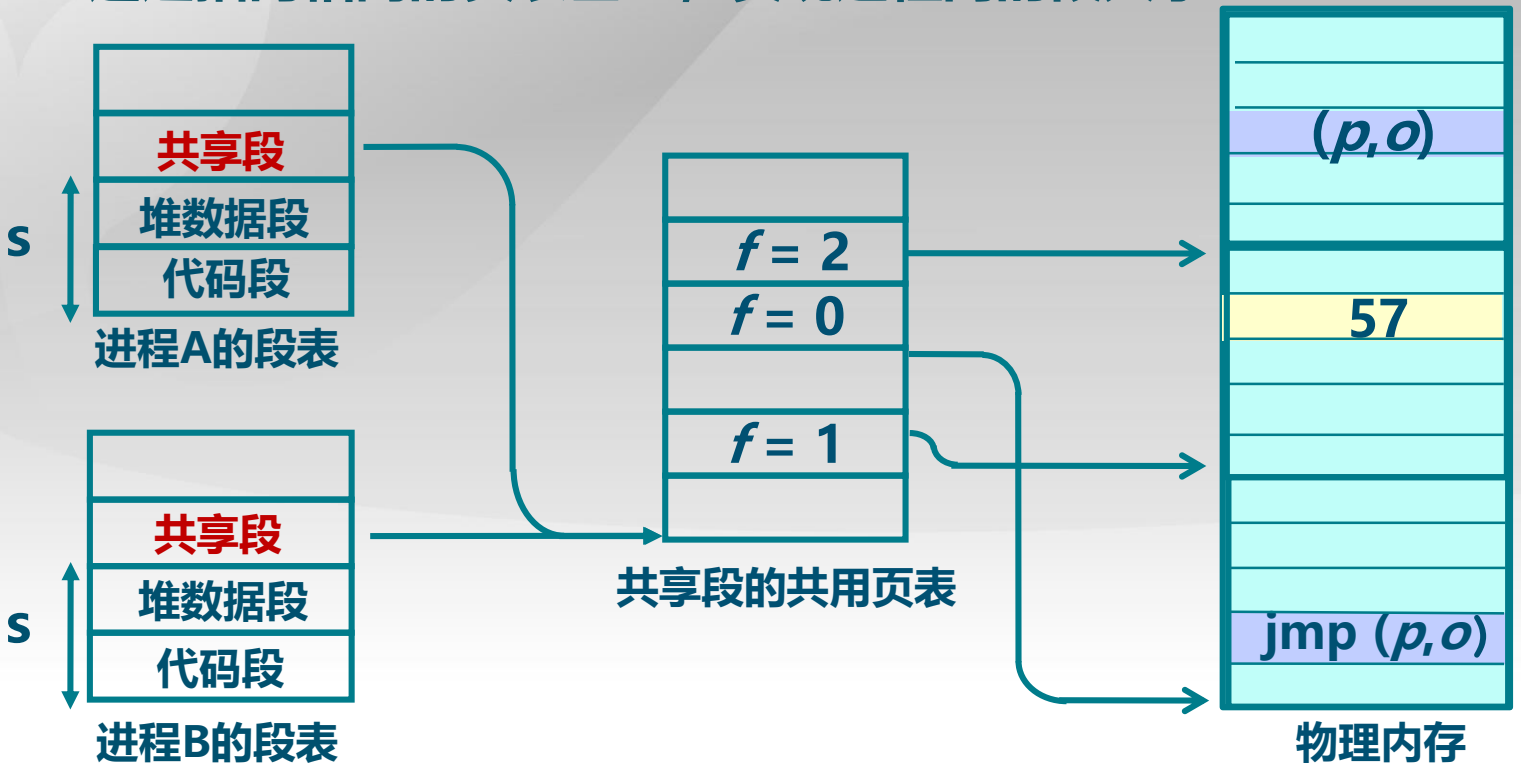
段页式存储管理

- 在段式存储管理基础上，给每个段加一级页表



段页式存储管理中的内存共享

- 通过指向相同的页表基址，实现进程间的段共享



操作系统

Operating System