操作系统

Operating system

孔维强

大连理工大学

内容纲要

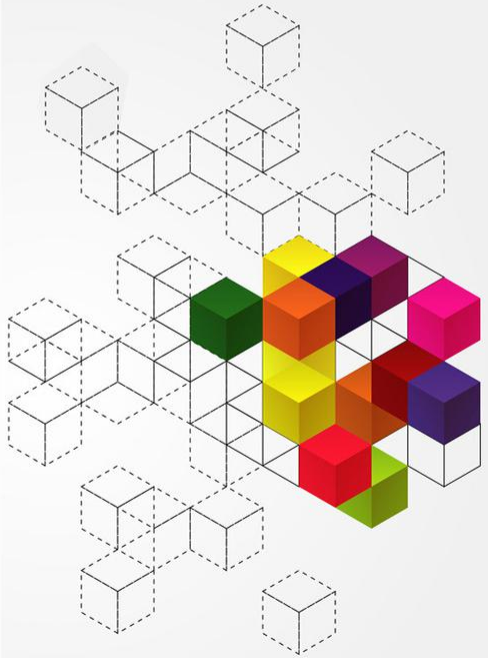
8.5 页表结构

一、 页表结构

二、 多级页表

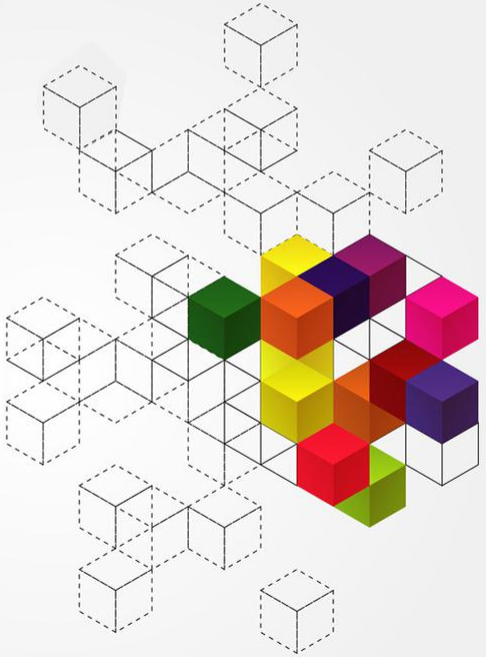
三、 哈希页表

四、 反置页表



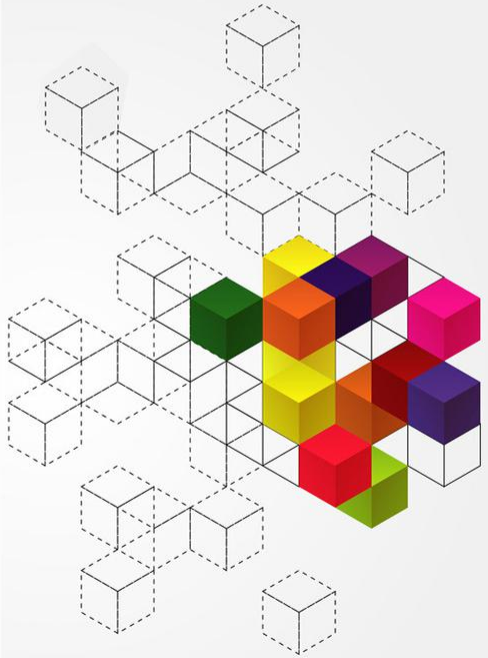
一、页表结构

- **计算机系统物理内存空间持续扩大，单级页表的结构已经无法满足实际需要**
  - 单级页表可能形成较大的页表。为了表达32位逻辑地址空间，4k大小的页，需要1 million 页表项的页表（ $2^{32}/2^{12}$ ）
  - 如果每个页表项要占用4 Bytes，则（每个进程）需要4MB连续内存去存储页表
  - 如果是64位的逻辑地址空间呢？ 2^{64} / 2^{12}
  - 问题：表项多，占据较多内存；查找效率低
    - 希望避免分配连续内存存储页表



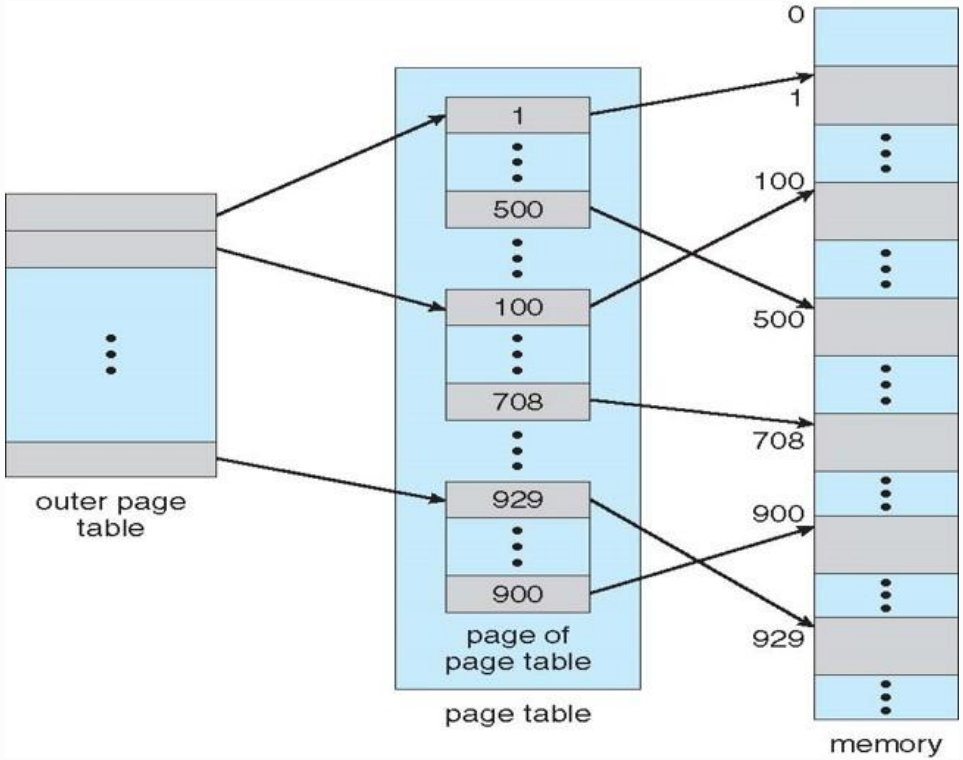
一、页表结构

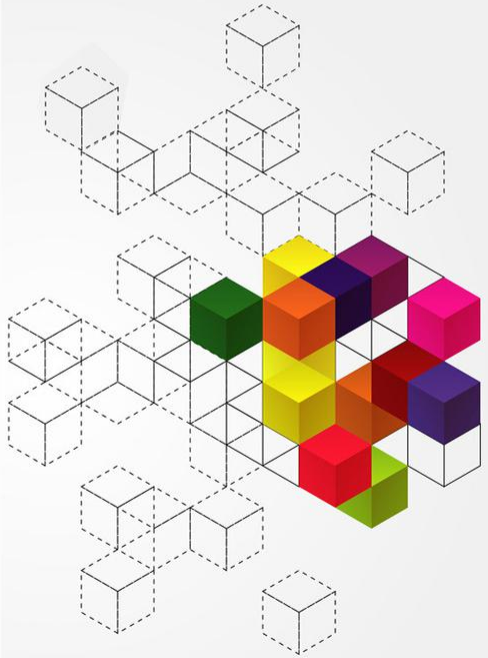
- **3种典型的页表结构**
  - 多级页表（Hierarchical Page Tables）
  - 哈希页表（Hashed Page Table）
  - 反置页表（Inverted Page Table）



二、多级页表

- **两级页表示意图**



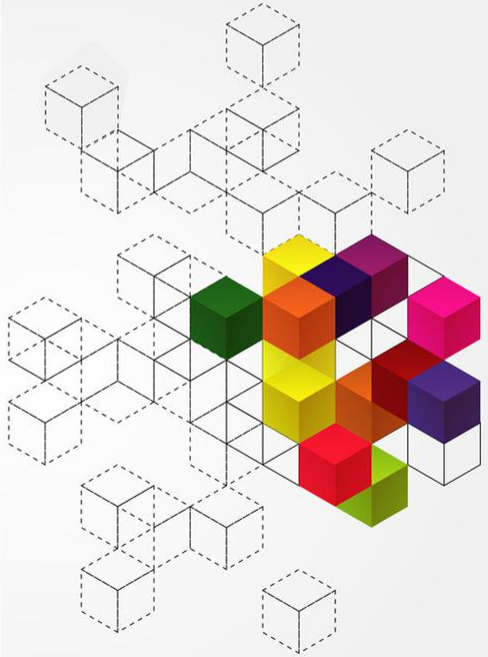


二、多级页表

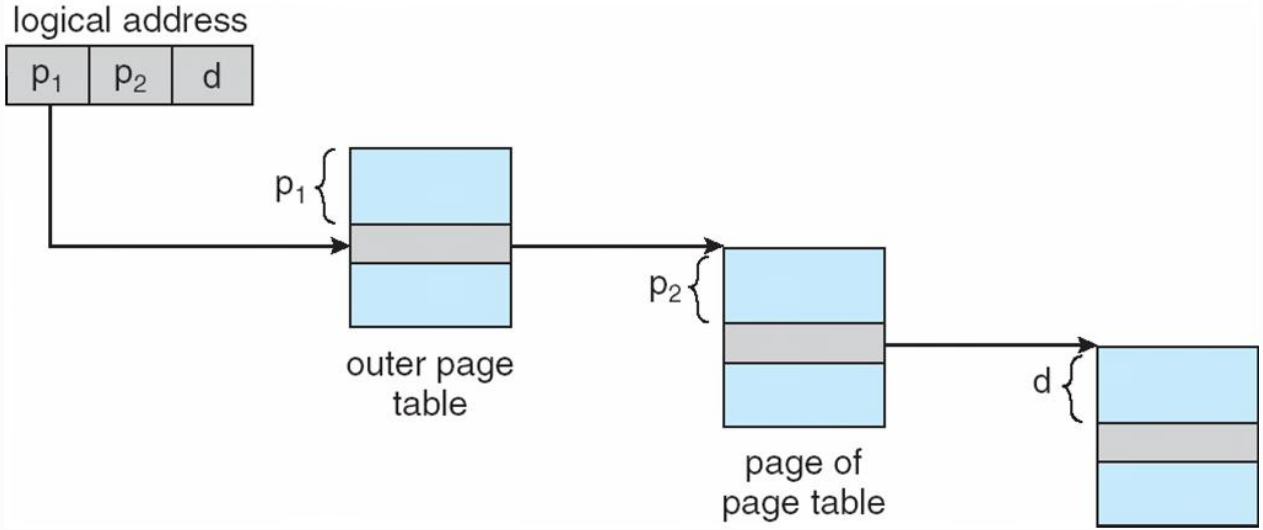
- **两级页表示例**
  - 逻辑地址（32-bit 机器，页面大小 4K (2^{12})) 被分割为:
    - 20 bits 的页号码
    - 12 bits 的页偏移
  - 由于页表也被分页，页号码被进一步分割（每个页表项 4bytes）为:
    - 10-bit 页号码
    - 10-bit 页偏移
  - 因此，逻辑地址变为:

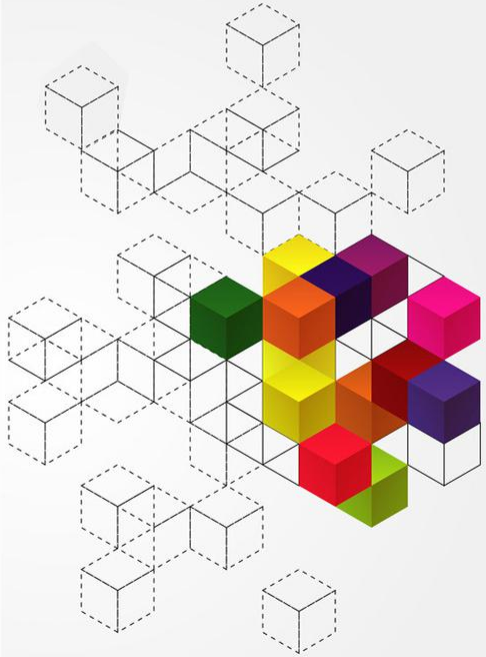
page number		page offset
p_1	p_2	d
10	10	12

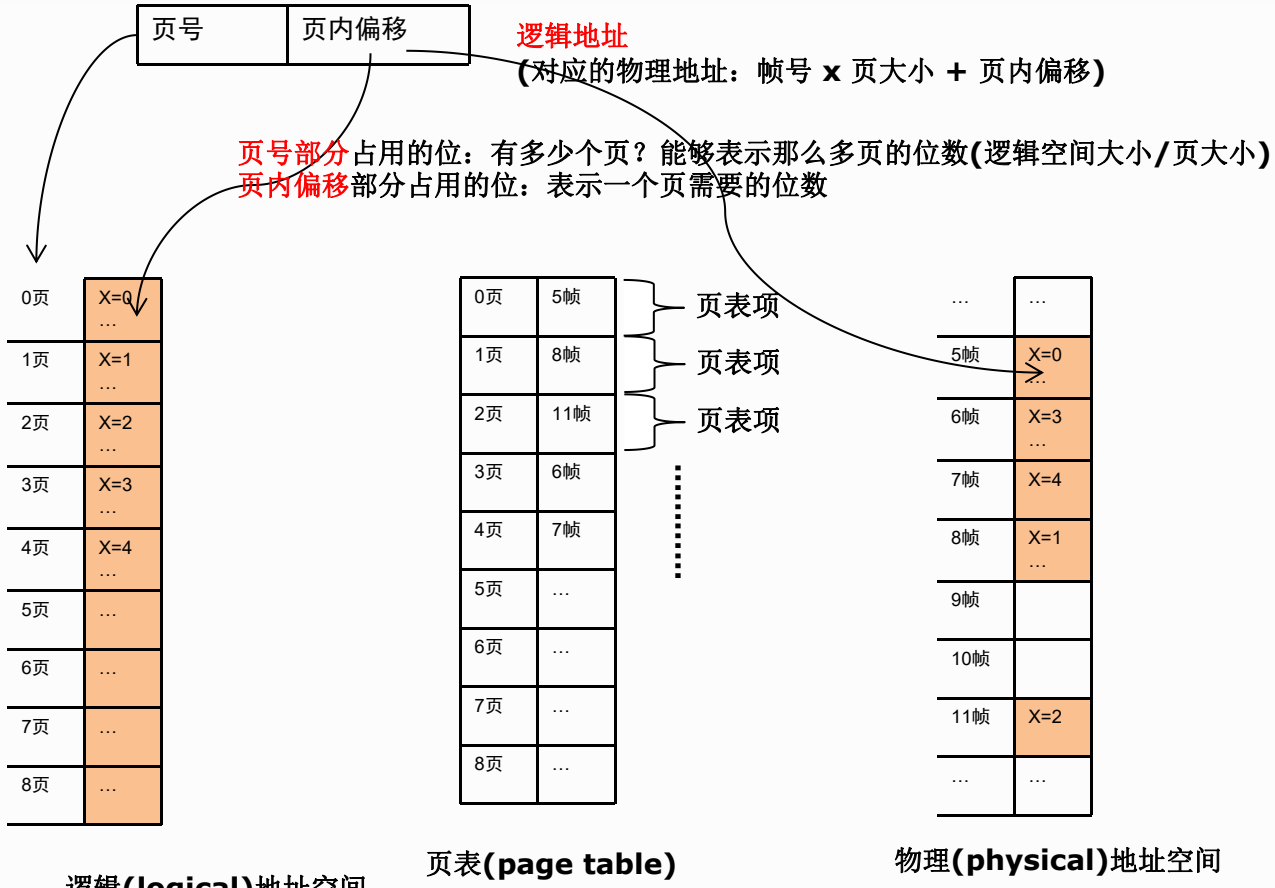
  - 其中 p_1 为外部页表的索引，p_2 为外部页表的页面中的置换

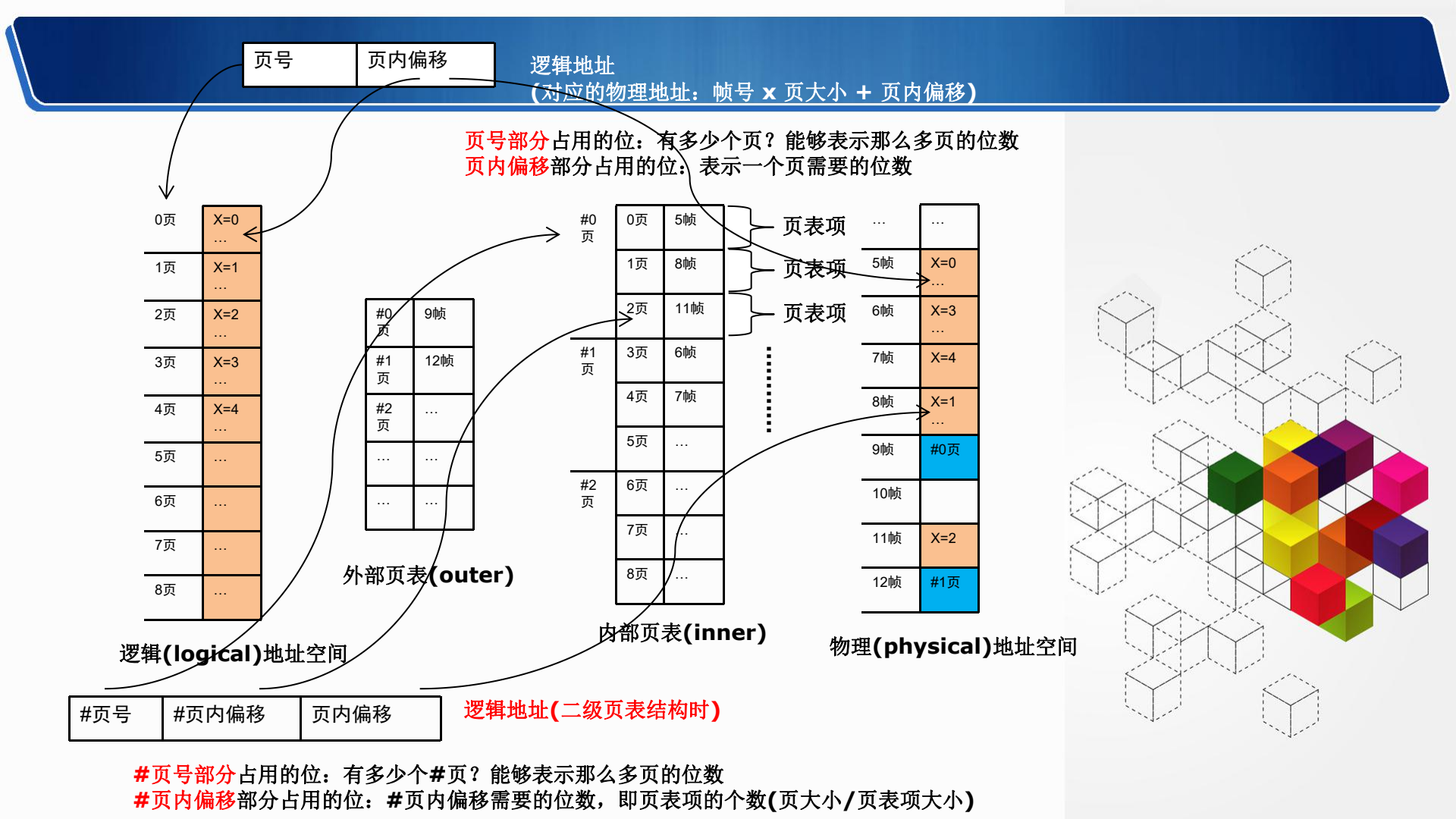


二、多级页表


logical address
p1
p2
d
p1
outer page table
p2
page of page table
d

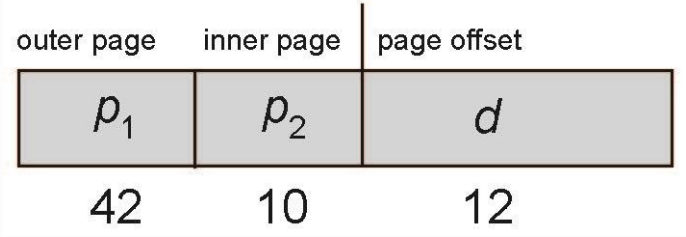


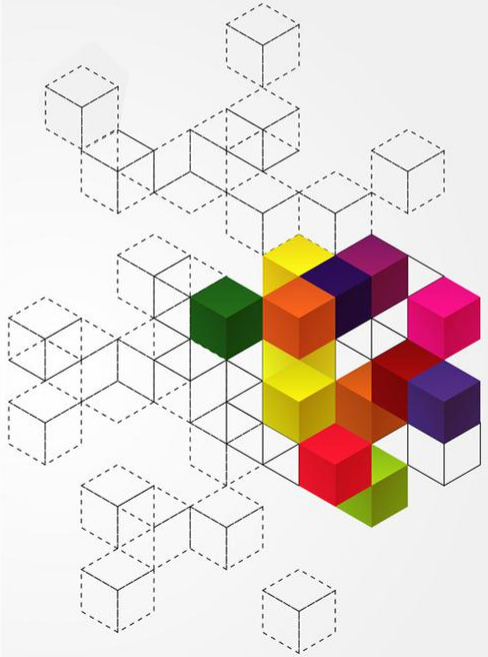

页号
页内偏移
逻辑地址
(对应的物理地址：帧号 x 页大小 + 页内偏移)
页号部分占用的位：有多少个页？能够表示那么多页的位数(逻辑空间大小/页大小)
页内偏移部分占用的位：表示一个页需要的位数
0页 X=0 …
1页 X=1 …
2页 X=2 …
3页 X=3 …
4页 X=4 …
5页 …
6页 …
7页 …
8页 …
0页 5帧
1页 8帧
2页 11帧
3页 6帧
4页 7帧
5页 …
6页 …
7页 …
8页 …
页表项
页表项
页表项
… …
5帧 X=0 …
6帧 X=3 …
7帧 X=4
8帧 X=1 …
9帧
10帧
11帧 X=2
… …
逻辑(logical)地址空间
页表(page table)
物理(physical)地址空间


页号
页内偏移
逻辑地址
(对应的物理地址：帧号 x 页大小 + 页内偏移)
页号部分占用的位：有多少个页？能够表示那么多页的位数
页内偏移部分占用的位：表示一个页需要的位数
0页 X=0 …
1页 X=1 …
2页 X=2 …
3页 X=3 …
4页 X=4 …
5页 …
6页 …
7页 …
8页 …
逻辑(logical)地址空间
#0页 9帧
#1页 12帧
#2页 …
外部页表(outer)
#0页 0页 5帧
1页 8帧
2页 11帧
#1页 3页 6帧
4页 7帧
5页 …
#2页 6页 …
7页 …
8页 …
页表项
页表项
页表项
内部页表(inner)
… …
5帧 X=0 …
6帧 X=3 …
7帧 X=4
8帧 X=1 …
9帧 #0页
10帧
11帧 X=2
12帧 #1页
物理(physical)地址空间
#页号 #页内偏移 页内偏移
逻辑地址(二级页表结构时)
#页号部分占用的位：有多少个#页？能够表示那么多页的位数
#页内偏移部分占用的位：#页内偏移需要的位数，即页表项的个数(页大小/页表项大小)

二、多级页表

- **两级页表示例**
  - 逻辑地址（64-bit 机器，页面大小 4K (2^{12})) 被分割为:
    - 页表有 2^{52} 个页表项
    - 使用2两级页表，则内部页表有2^{10} 4-byte的页表项
    - 外部页表有2^{42}个页表项或2^{44}byte（16T？）
    - 可以进一步增加级数，但再加一级外部也会有2^{34}个页表项



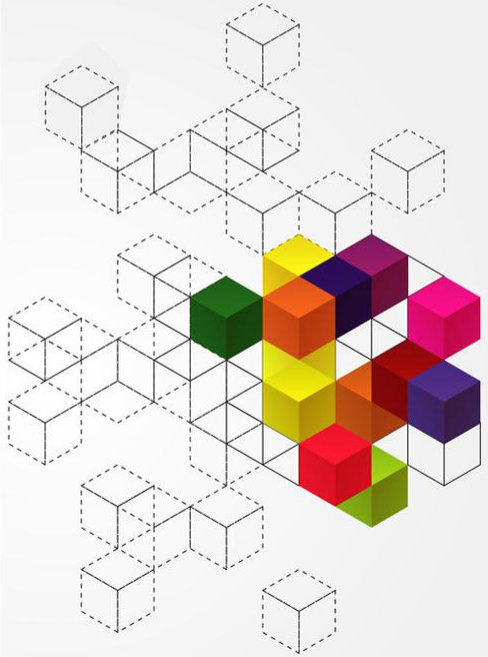


二、多级页表

outer page	inner page	offset
p_1	p_2	d
42	10	12

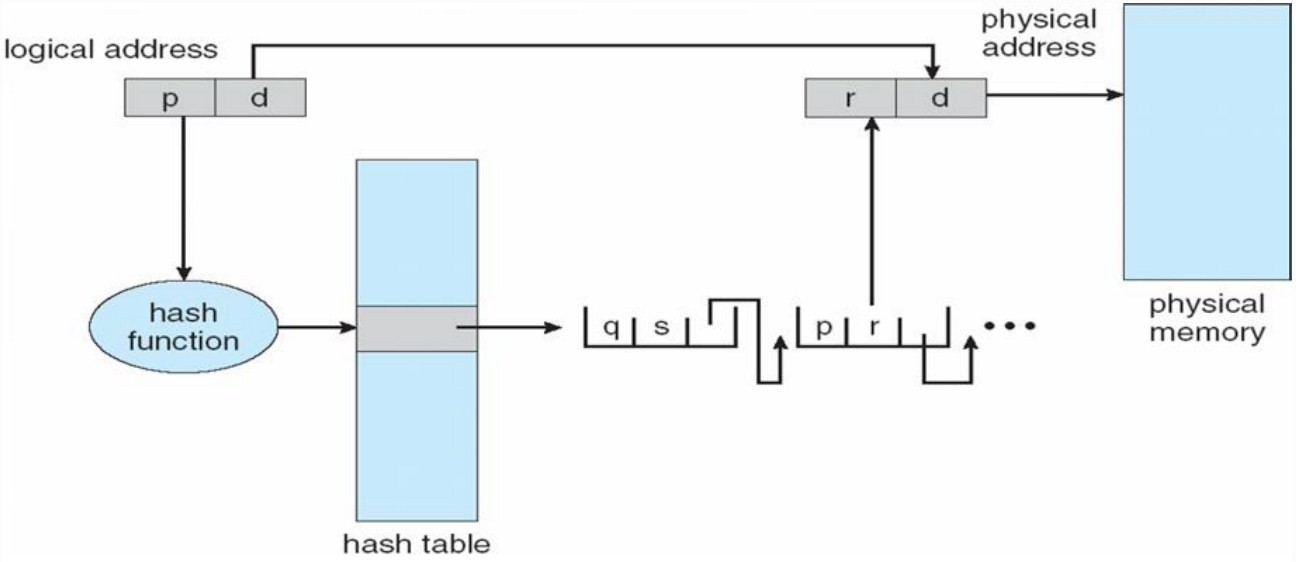
2nd outer page	outer page	inner page	offset
p_1	p_2	p_3	d
32	10	10	12

- 进一步增加级数，也会增加内存访问的次数

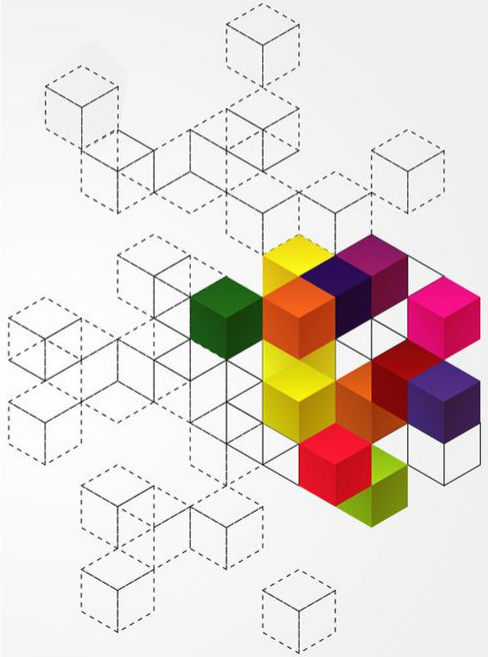


三、哈希页表

- **以哈希结构组织页表**

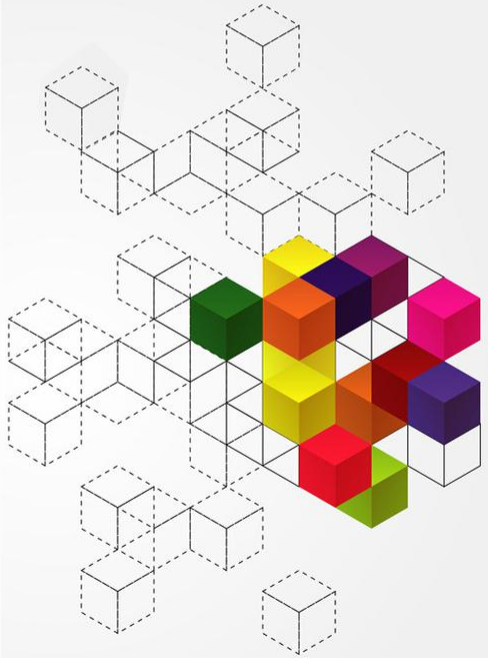


- 在地址空间 > 32位时较为常用
- 查找速度快（无查找，仅hash计算）
- 需要的存储空间更小



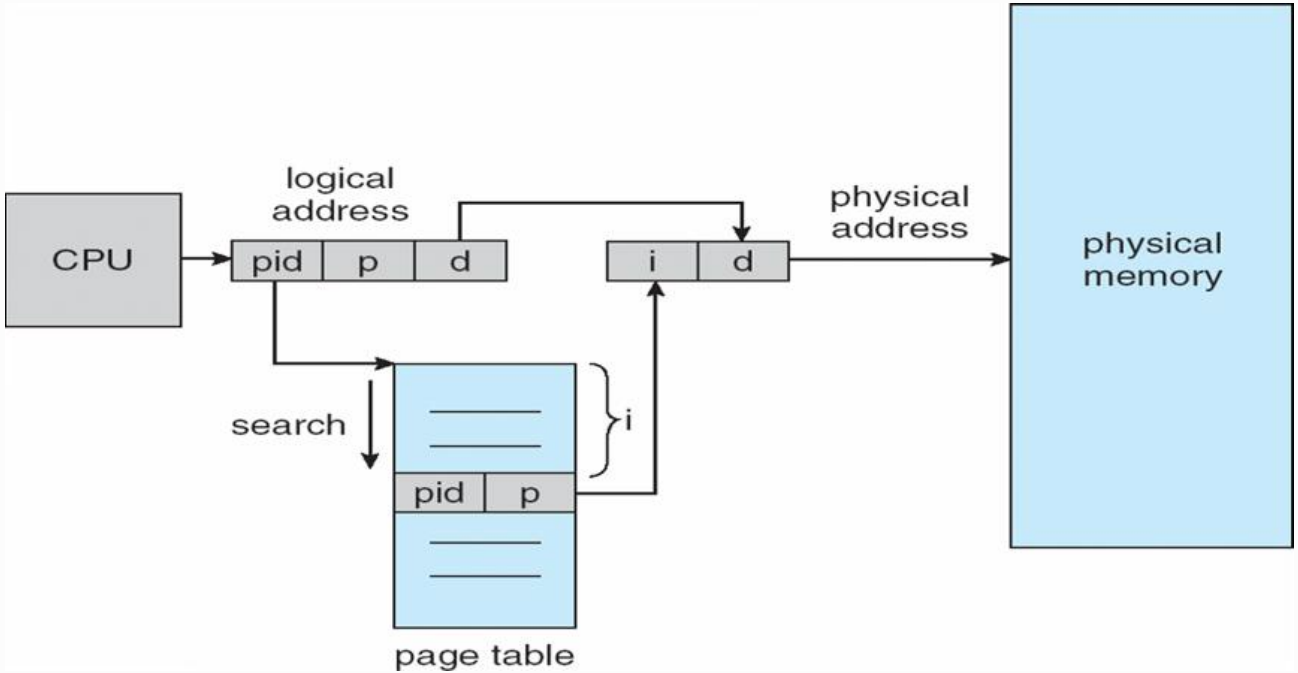
四、反置页表

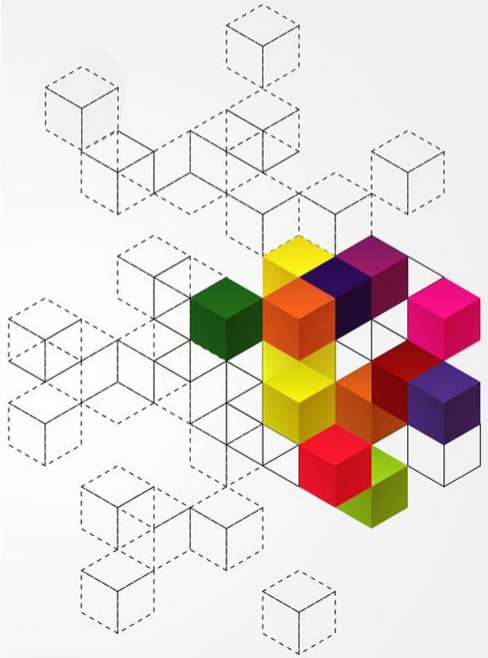
- **反置页表结构**
  - 不为每个进程建立页表，而是为物理页（帧）建页表
  - 每个页表项对应着真实的物理帧
  - 每个页表项由存在在该帧的页地址、拥有该页的进程
  - 整个系统中仅有1个页表
  - 减少了存储页表所需的内存空间，但是增加了查找页表的时间（根据物理地址排序而不是逻辑地址）



四、反置页表

- **反置页表结构**





本讲小结

- 页表结构