

# 操作系统

# Operating Systems

## L22 多级页表与快表

### Multilevel Paging

授课教师：李治军

[lizhijun\\_os@hit.edu.cn](mailto:lizhijun_os@hit.edu.cn)

综合楼411室

# 为了提高内存空间利用率，页应该小，但是页小了页表就大了...

页框7	
页框6	段0: 页3
页框5	段0: 页0
页框4	
页框3	段0: 页2
页框2	
页框1	段0: 页1
页框0	

页号	页框号	保护
0	5	R
1	1	R/W
2	3	R/W
3	6	R



# 页表会很大，页表放置就成了问题...

## ■ 纸上和实际使用总是存在很大差别!

- 页面尺寸通常为4K，而地址是32位的，有 $2^{20}$ 个页面
- $2^{20}$ 个页表项都得放在内存中，需要4M内存；系统中并发10个进程，就需要40M内存
- 实际上大部分逻辑地址根本不会用到

32位: 总空间[0, 4G]!

页号	页框号	保护	有效
0	5	R	1
1	1	R/W	1
2			0
3	6	R	1



页号	页框号	保护
0	5	R
1	1	R/W
3	3	R



# 第一种尝试，只存放用到的页

## ■ 很自然的想法：用到的逻辑页才有页表项

- 但页表中的页号不连续，就需要比较、查找，折半

$\log(2^{20})=20$ ，20次什么？

- 32位地址空间+ 4K页面+页号必须连续 $\Rightarrow 2^{20}$ 个页表项 $\Rightarrow$ 大页表占用内存，造成浪费

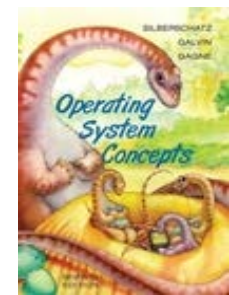
既要连续又要让页表占用内存少，怎么办？

用书的章目录和节目录来类比思考...

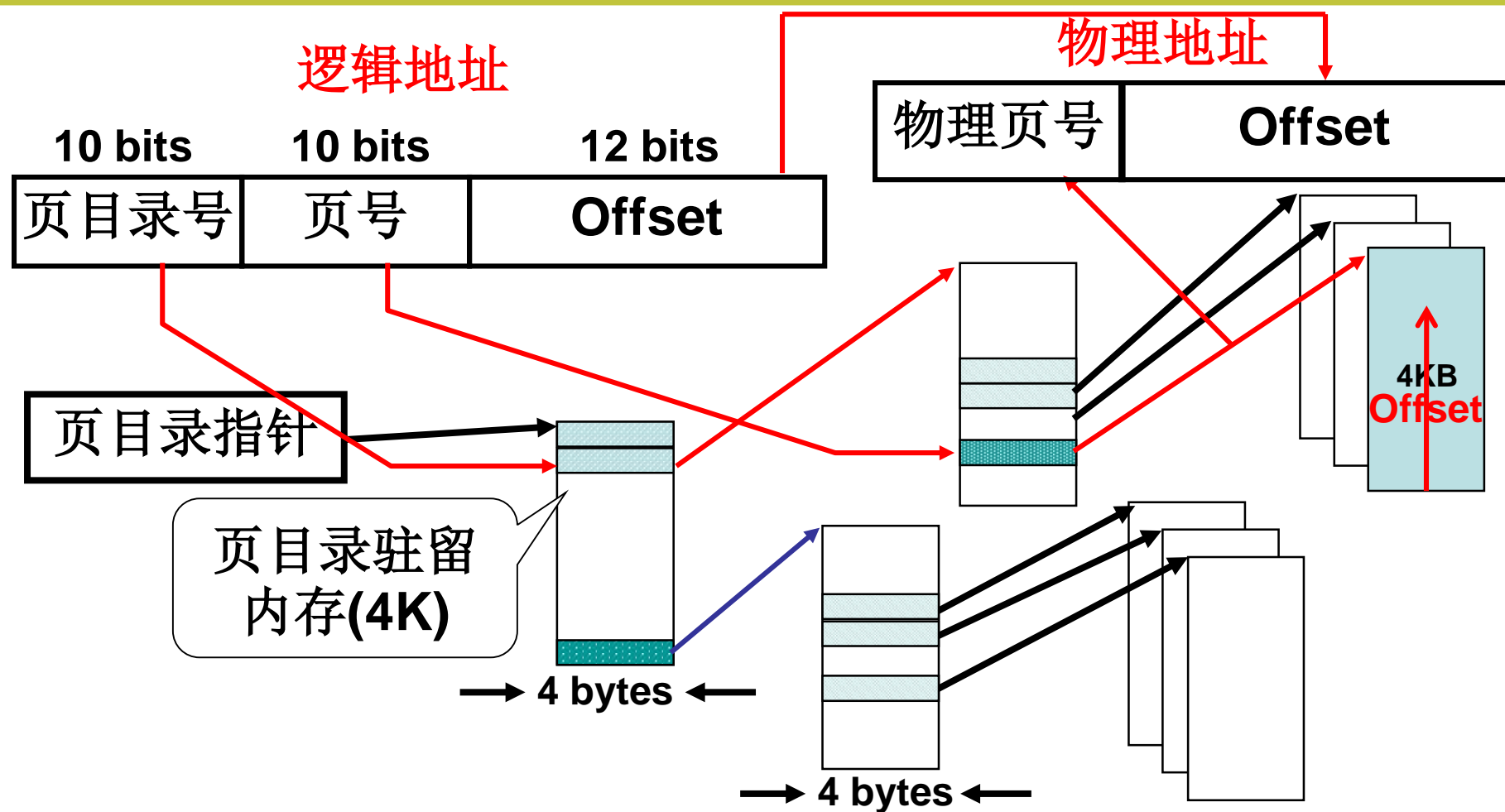
页号	页框号	保护
0	5	R
1	1	R/W
3	3	R



页号	页框号	保护	有效
0	5	R	1
1	1	R/W	1
2			0
3	6	R	1



## 第二种尝试：多级页表，即页目录表（章）+页表（节）



- $2^{10}$ 个目录项  $\times$  4字节地址 = 4K，总共需要  $16K \ll 4M$

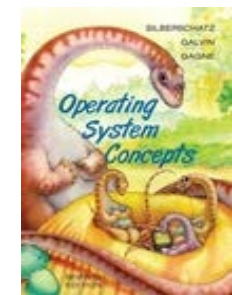
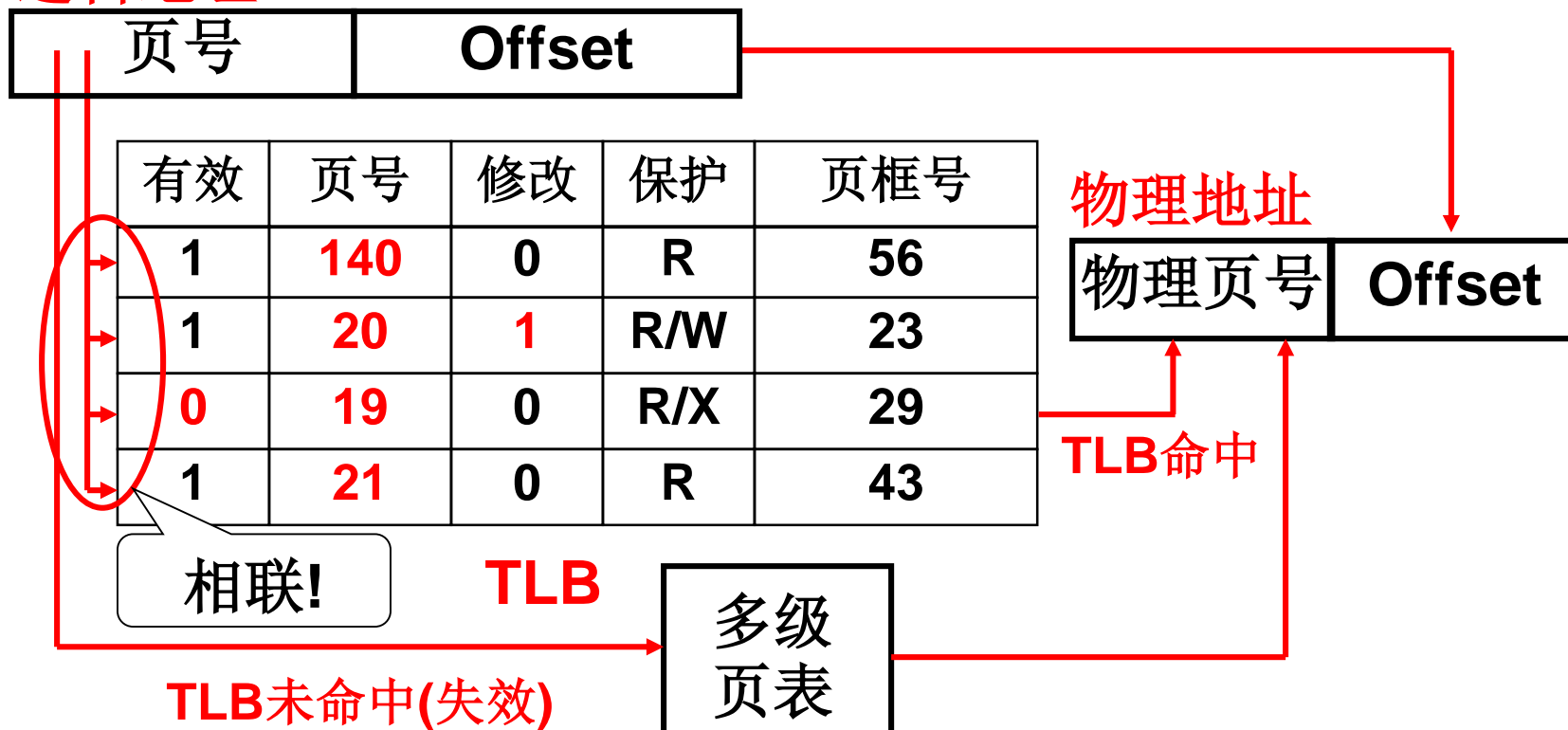


# 多级页表提高了空间效率，但在时间上？

■ 多级页表增加了访存的次数，尤其是64位系统

■ TLB是一组相联快速存储，是寄存器

逻辑地址



# TLB得以发挥作用的原因

- TLB命中时效率会很高，未命中时效率降低

$$\text{有效访问时间} = \text{HitR} \times (\text{TLB} + \text{MA}) + (1 - \text{HitR}) \times (\text{TLB} + 2\text{MA})$$

命中率!

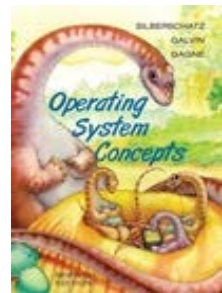
内存访问时间!

TLB时间!

$$\text{有效访问时间} = 98\% \times (20\text{ns} + 100\text{ns}) + 2\% \times (20\text{ns} + 200\text{ns}) = 122\text{ns}$$

$$\text{有效访问时间} = 10\% \times (20\text{ns} + 100\text{ns}) + 90\% \times (20\text{ns} + 200\text{ns}) = 210\text{ns}$$

- 要想真正实现“近似访存1次”，  
TLB的命中率应该很高
- TLB越大越好，但TLB很贵，通常  
只有[64, 1024]

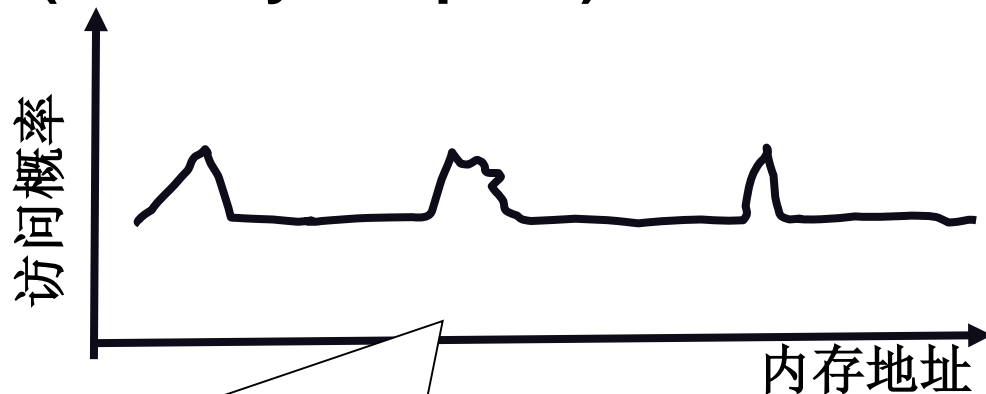


# 为什么TLB条目数可以在64-1024之间？

■ 相比 $2^{20}$ 个页，64很小，为什么TLB就能起作用？

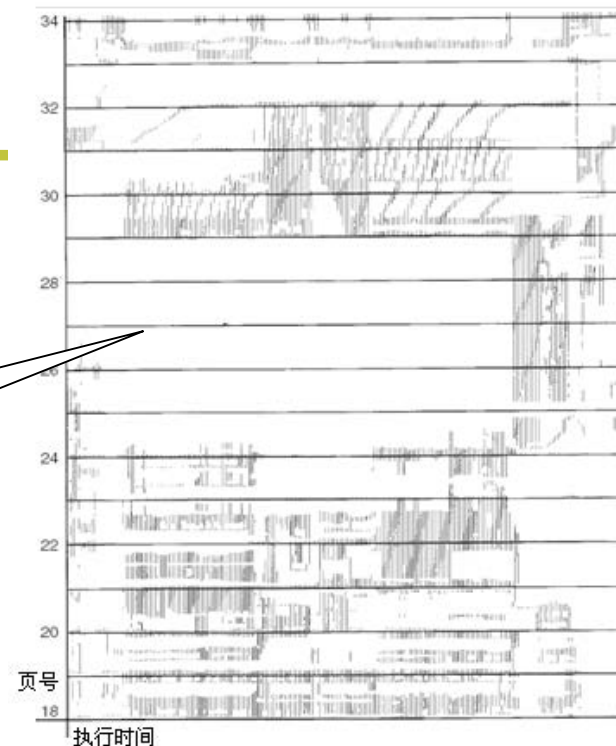
■ 程序的地址访问存在局部性

■ 空间局部性(Locality in Space)



程序多体现为循环、顺序结构

某内存引用模式



■ 计算机系统设计时应该充分利用这一局部性

