哈工大操作系统-L22多级页表和快表

实用的内存管理机制=多级页表+快表+分页机制

1.单纯分页的问题

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 1K是2^10
  - 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

2^20=1M
一个页表项用4字节表示，因为4字节可以描述清楚1M个页表
需要4MB内存
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

- 假设每个进程都有自己的总页表。即每个进程都建立全部的页框到页面的对应映射。进程中所有的页框我们都要存在进程中，不管这个页框有没有用到。
- 如果100个进程，就需要400M内存了。
- 把进程的总页表都存了，有点不合理。

2.只存放用到的页

第一种尝试，只存放用到的页

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

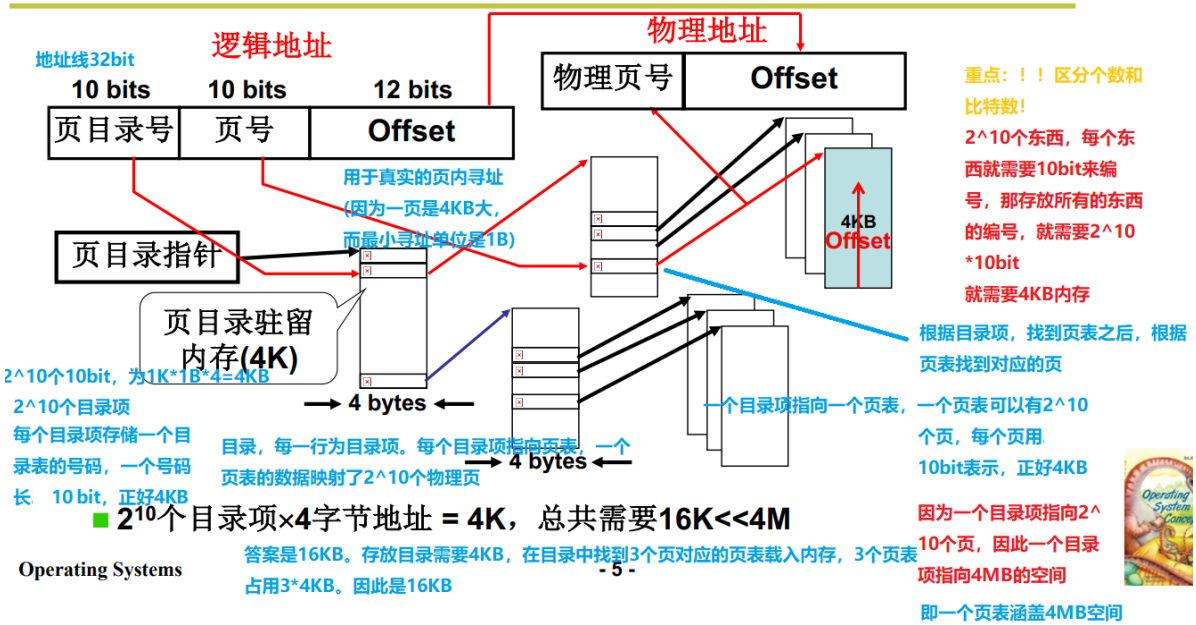
- 很自然的想法：用到的逻辑页才有页表项
  - 但页表中的页号不连续，就需要比较、查找，折半
    $\log(2^{20})=20$，20次什么？
  - 32位地址空间+ 4K页面+页号必须连续⇒2^{20}个页表项⇒大页表占用内存，造成浪费

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...

3.多级页表

第二种尝试：多级页表，即页目录表（章）+页表（节）



- 2^{10}个目录项×4字节地址 = 4K，总共需要16K<<4M

答案是16KB。存放目录需要4KB，在目录中找到3个页对应的页表载入内存，3个页表占用3*4KB。因此是16KB

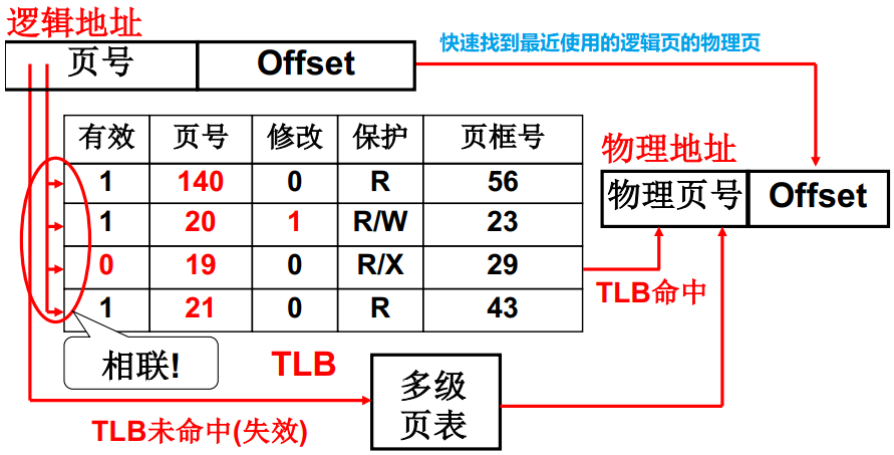
- **逻辑地址=页目录号(10bit)+页号(10bit)+偏移(12bit)**
  - 每一页有4KB，而**寻址的最小单位是B**，因此12bit可以寻址一页。因此，**偏移是为了寻址页内内容**
  - 页号有10bit，即一个页表存放2^10个页号。页表其实也是放在4KB的页中的，而4KB正好可以这样组织:一页存放2^10个页，每个页的页号用10bit表示，这样4KB就完全存放了2^10个页的所有编号(2^10乘10bit=1K 乘4B=4KB)
  - 同理，页目录号有10bit,即一个目录存放2^10个页表。
- **三级页表=目录+页表+页**
  - 一个目录，索引出多个页表，一个页表索引出页，一个页存放4KB
- 多级页表，多增加一级，内存节省，但访问内存的次数增加一次

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

4.快表(TLB)

- TLB是一组相联快速存储，是寄存器



- 快表+多级页表既保证了空间效率也保证了时间效率。
- 利用了**空间局部性**

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

这里2MA是单级页表，即从一个页表，然后到页

有效访问时间 = HitR×(TLB+MA) + (1-HitR)×(TLB+2MA)

命中率! 内存访问时间! TLB时间!

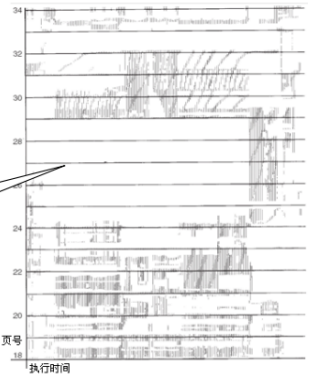
有效访问时间=98%×(20ns+100ns) + 2%×(20ns+200ns)=122ns

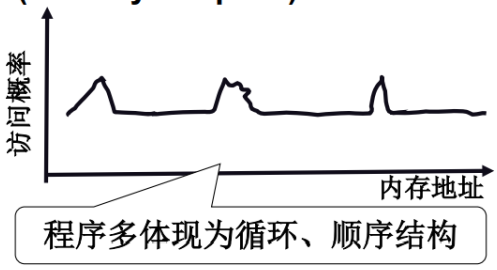
有效访问时间=10%×(20ns+100ns) + 90%×(20ns+200ns)=210ns

- 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)





- 计算机系统设计时应该充分利用这一局部性