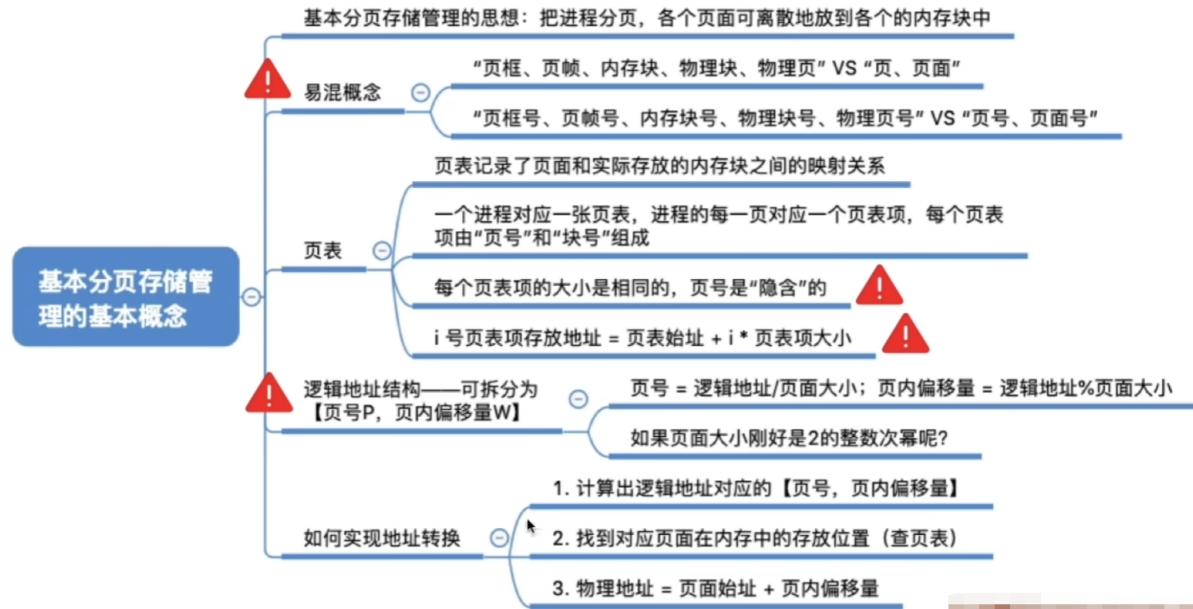
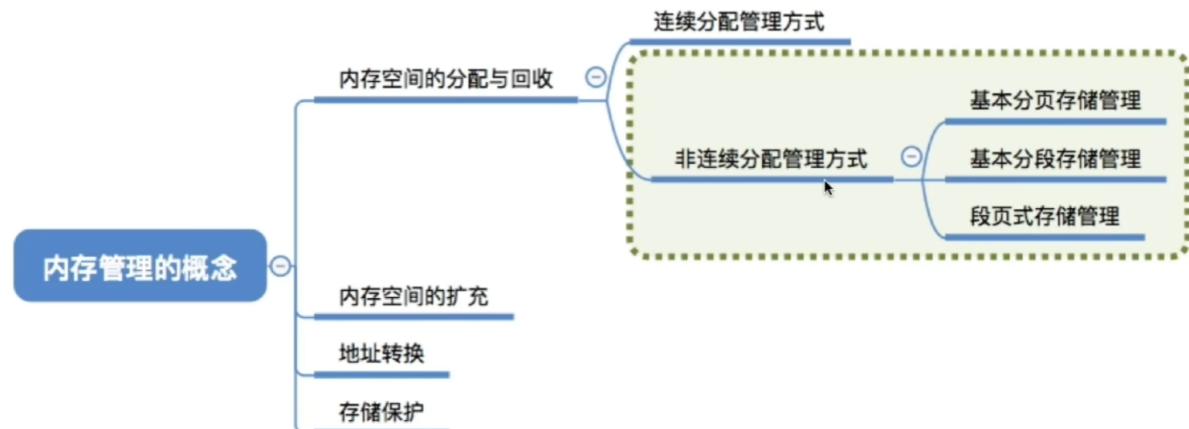


基本分页存储管理



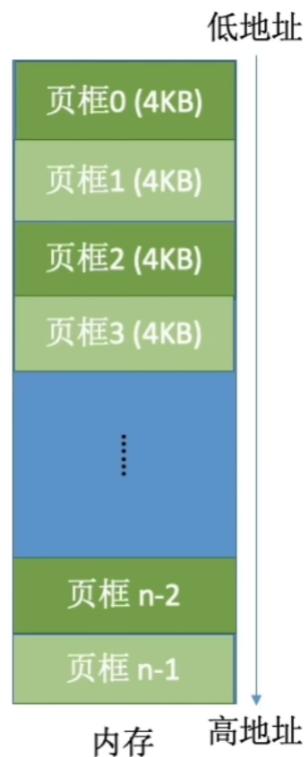
连续分配：为用户进程分配的必须是一个连续的内存空间。

非连续分配：为用户进程分配的可以是一些分散的内存空间。

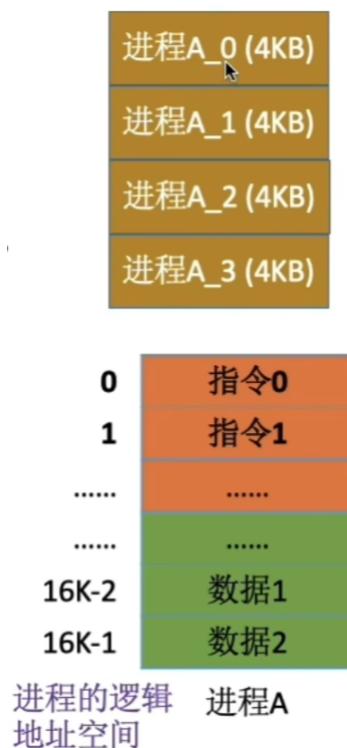


什么是分页存储

将内存空间分为一个个大小相等的分区（比如：每个分区4KB），每个分区就是一个“页框”（页框=页帧=内存块=物理块=物理页面）。每个页框有一个编号，即“页框号”（页框号=页帧号=内存块号=物理块号=物理页号），页框号从0开始。



将进程的逻辑地址空间也分为与页框大小相等的一个个部分，每个部分称一个“页”或“页面”。每个页面也有一个编号，即“页号”，页号也是从0开始。



Tips: 易混淆

页、页面vs页框、页帧、物理页

页号、页面号vs页框号、页帧号、物理页号

操作系统以页框为单位为各个进程分配内存空间。进程的每个页面分别放入一个页框中。也就是说，进程的页面与内存的页框有一一对应的关系。

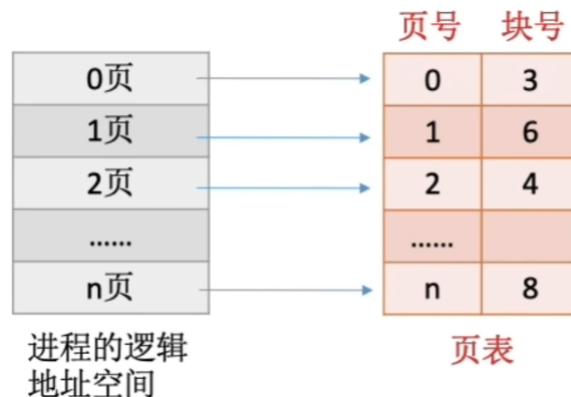


各个页面不必连续存放，可以放到不相邻的各个页框中

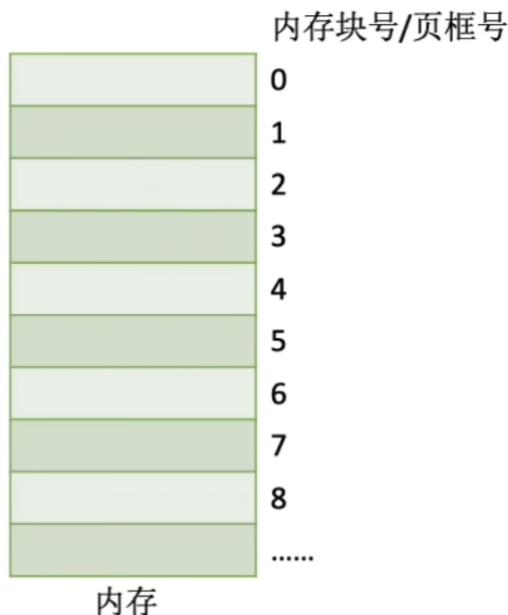
重要的数据结构 页表

为了能知道进程的每个页面在内存中存放的位置，操作系统要为每个进程建立一张页表。

注：页表通常存在PCB（进程控制块）中



1. 一个进程对应一张页表
2. 进程的每个页面对应一个页表项
3. 每个页表项由“页号”和“块号”组成
4. 页表记录进程页面和实际存放的内存块之间的映射关系



思考：

1. 每个页表项多大？占几个字节？

假设某系统物理内存大小为4GB，页面大小为4KB，则每个页表项至少应该为多少字节？

$$\text{内存块大小} = \text{页面大小} = 4KB = 2^{12}B$$

→4GB的内存总共会被分为 $2^{32}/2^{12} = 2^{20}$ 个内存块

→内存块号的范围应该是 $[0, 2^{20} - 1]$

→内存块号至少要用20bit来表示

→至少要用3B来表示块号($3 * 8 = 24bit$)

→由于页号是隐含的，因此每个页表项占3B，存储整个页表至少需要 $3 * (n + 1)B$

注意：页表记录的只是内存块号，而不是内存块的起始地址！

$$J\text{号内存块的起始地址} = J * \text{内存块大小}$$

计算机中内存块的数量 → 页表项中块号至少占多少字节

页表项连续存放，因此页号可以是隐含的，不占存储空间（类比数组）



假设页表中的各页表项从内存地址为X的地方开始连续存放

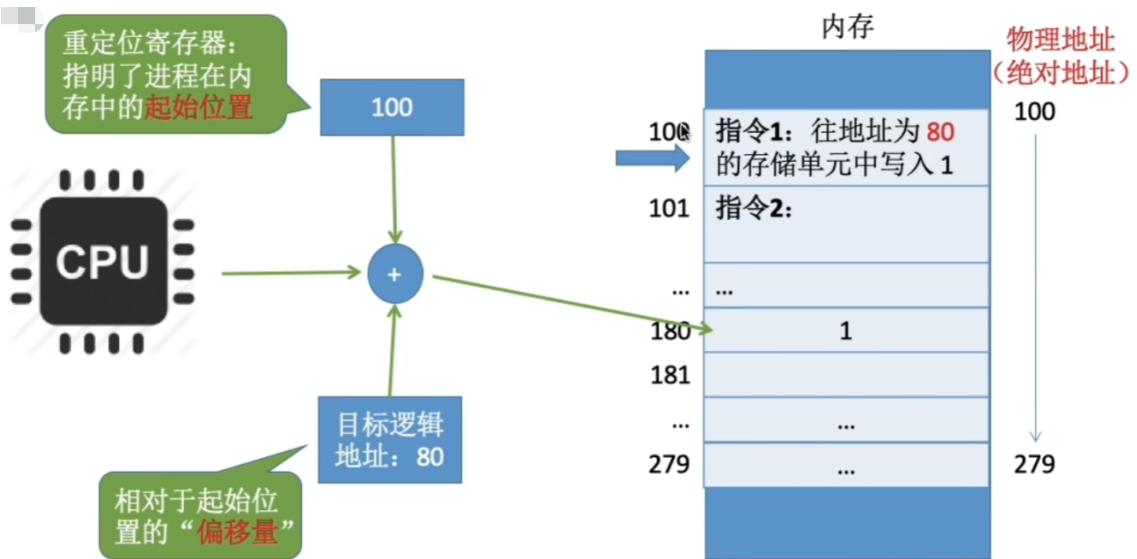
如何找到页号为i的页表项？

$$i\text{号页表项的存放地址} = X + 3*i$$

因此，页表中的页号可以是隐含的，即页号不占用存储空间

2. 如何通过页表实现逻辑地址到物理地址的转换？

进程在内存中连续存放时，操作系统是如何实现逻辑地址到物理地址的转换的？



将进程地址空间分页之后，操作系统该如何实现逻辑地址到物理地址的转换？

特点：虽然进程的各个页面是离散存放的，但是页面内部是连续存放的

如果要访问逻辑地址A，则

1. 确定逻辑地址A对应的“页号”P
2. 找到P号页面在内存中的起始地址（需要查页表）
3. 确定逻辑地址A的“页内偏移量”W

$$\text{逻辑地址 } A \text{ 对应的物理地址} = P \text{ 号页面在内存中的起始地址} + \text{页内偏移量 } W$$

3. 如何确定一个逻辑地址对应的页号、页内偏移量？

在某计算机系统中，页面大小是50B。某进程逻辑地址空间大小为200B，则逻辑地址110对应的页号、页内偏移量是多少？



$$\text{页号} = \text{逻辑地址} / \text{页面长度}$$

$$\text{页内偏移量} = \text{逻辑地址 \% 页面长度}$$

$$\text{页号} = 110 / 50 = 2$$

$$\text{页内偏移量} = 110 \% 50 = 10$$

逻辑地址可以拆分为（页号，页内偏移量）

通过页号查询页表，可知页面在内存中的起始地址

$$\text{页面在内存中的起始地址} + \text{页内偏移量} = \text{实际的物理地址}$$

在计算机内部，地址是用二进制表示的，如果页面大小刚好是2的整数幂，则计算机硬件可以很快的把逻辑地址拆分成（页号，页内偏移量）

假设某计算机用32个二进制位表示逻辑地址，页面大小为 $4KB = 2^{12}B = 4096B$

0号页的逻辑地址范围应该是 0~4095，用二进制表示应该是：

00000000000000000000 ~ 0000000000000000111111111111

1号页的逻辑地址范围应该是 4096~8191，用二进制表示应该是：

0000000000000000100000000000 ~ 0000000000000000111111111111

2号页的逻辑地址范围应该是 8192~12287，用二进制表示应该是：

0000000000000000100000000000 ~ 0000000000000000111111111111

Eg: 逻辑地址 2，用二进制表示应该是 000000000000000000000000000010

页号 = $2/4096 = 0 = 0000000000000000$ ，页内偏移量 = $2\%4096 = 2 = 000000000010$

Eg: 逻辑地址 4097，用二进制表示应该是 00000000000000000000000000001000000000001

页号 = $4097/4096 = 1 = 0000000000000001$ ，页内偏移量 = $4097\%4096 = 1 = 000000000001$

结论：如果每个页面大小为

$$2^K B$$

用二进制数表示逻辑地址，则末尾K位即为页内偏移量，其余部分就是页号

假设某计算机用32个二进制位表示逻辑地址，页面大小为 $4KB = 2^{12}B = 4096B$

Eg: 逻辑地址 4097，用二进制表示应该是 00000000000000000000000000001000000000001

页号 = $4097/4096 = 1 = 0000000000000001$ ，页内偏移量 = $4097\%4096 = 1 = 000000000001$

假设物理地址也用32个二进制位表示，则由于内存块的大小=页面大小，因此：

0号内存块的起始物理地址是 00000000000000000000000000000000

1号内存块的起始物理地址是 00000000000000000000000000000001

2号内存块的起始物理地址是 000000000000000000000000000000010

3号内存块的起始物理地址是 000000000000000000000000000000011

根据页号可以查询页表，而页表中记录的只是内存块号，而不是内存块的起始地址！
J号内存块的起始地址 = J * 内存块大小

假设通过查询页表得知1号页面存放的内存块号是9（1001），则

9号内存块的起始地址 = $9 * 4096 = 00000000000000001001000000000000$

则逻辑地址4097对应的物理地址 = 页面在内存中存放的起始地址 + 页内偏移量

= (000000000000000010010000000001)

结论：如果页面大小刚好是2的整数幂，则只需把页表中记录的物理块号拼接上页内偏移量就能得到对应的物理地址

总结：页面大小刚好是2的整数幂有什么好处？

1. 逻辑地址的拆分更加迅速

如果每个页面大小为

$$2^K B$$

用二进制数表示逻辑地址，则末尾K位即为页内偏移量，其余部分就是页号。因此，如果让每个页面的大小为2的整数幂，计算机硬件就可以很方便地得出一个逻辑地址对应的页号和页内偏移量，而无需进行除法运算，从而提升了运行速度。

2. 物理地址的计算更加迅速

根据逻辑地址得到页号，根据页号查询页表从而找到页面存放的内存块号，将二进制表示的内存块号和页内偏移量拼接起来，就可以得到最终的物理地址。

逻辑地址结构

分页存储管理的逻辑地址结构如下所示：

31	12	11	0
页号 P				页内偏移量 W	

地址结构包含两个部分：前一部分为页号，后一部分为页内偏移量W

地址长度为32位，其中0~11位为页内偏移量，或称页内地址

12~31位为页号

如果有K位表示页内偏移量，则说明该系统中一个页面的大小是

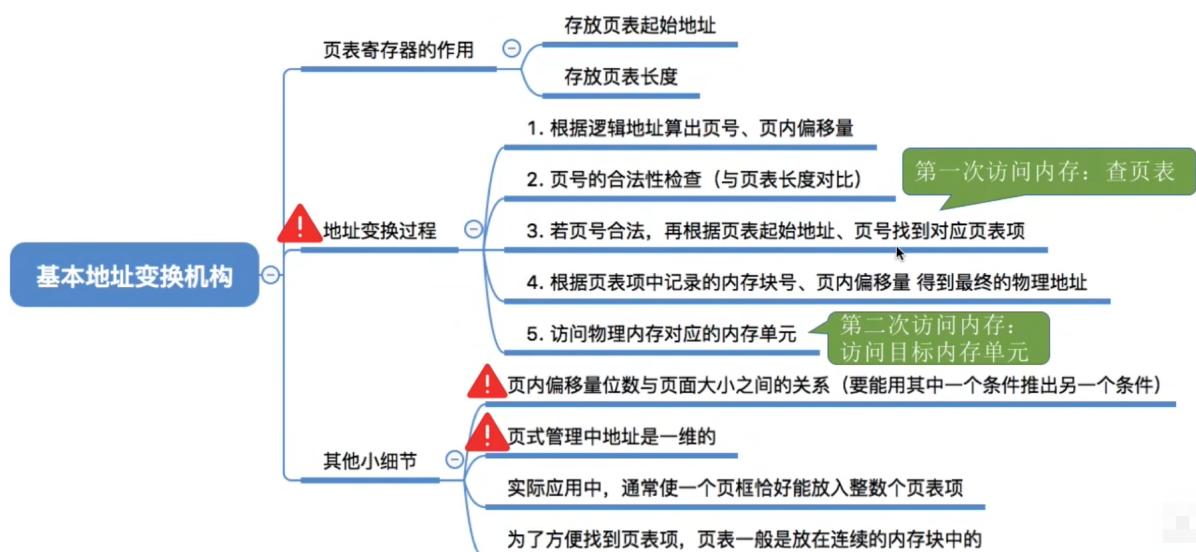
$$2^K \text{个内存单元}$$

如果有M位表示页号，则说明在该系统中，一个进程最多允许有

$$2^M \text{个页面}$$

页面大小 \leftrightarrow 页内偏移量位数 \rightarrow 逻辑地址结构

基本地址变换机构



重点理解、记忆基本地址变换机构（用于实现逻辑地址到物理地址的一组硬件机构）的原理和流程

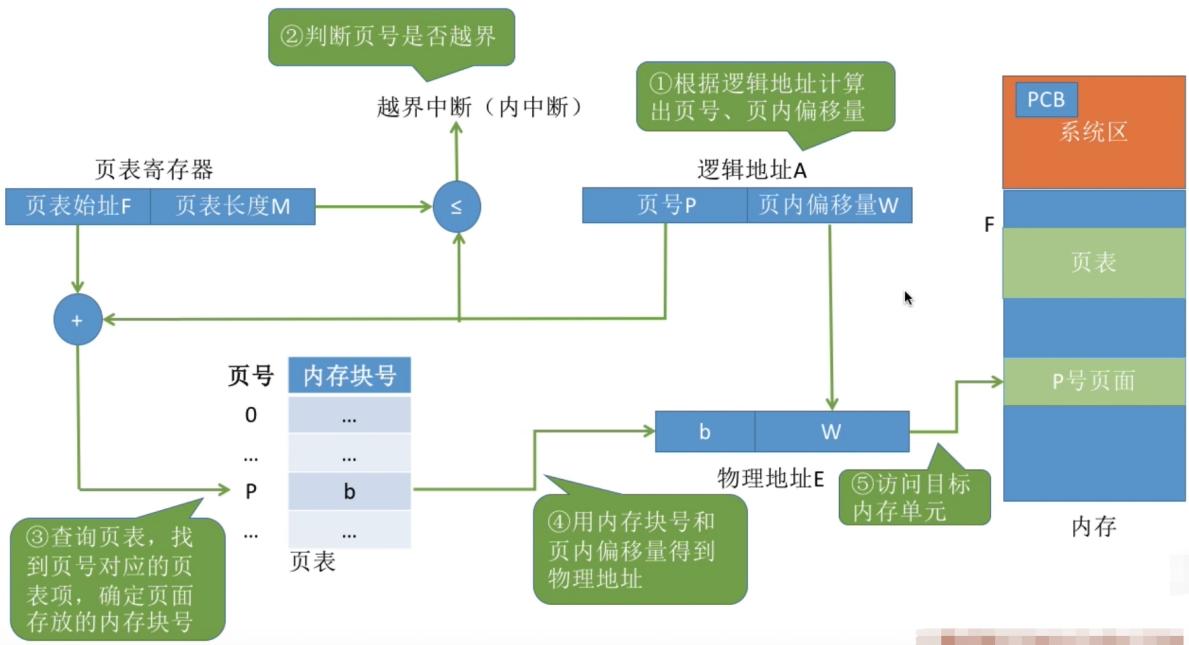
基本地址变换机构可以借助进程的页表将逻辑地址转换为物理地址。

通常会在系统中设置一个页表寄存器（PTR），存放页表在内存中的起始地址F和页表长度M。

进程未执行时，页表的始址和页表长度放在进程控制块（PCB）中，当进程被调度时，操作系统内核会把它们放到页表寄存器中。

注意：页面大小是2的整数幂

设页面大小为L，逻辑地址A到物理地址E的变换过程如下：



1. 计算页号P和页内偏移量W（如果用十进制数手算，则 $P=A/L$, $W=A\%L$;但是在计算机实际运行时，逻辑地址结构是固定不变的，因此计算机硬件可以更快地得到二进制表示的页号、页内偏移量）
2. 比较页号P和页表长度M，若 $P>=M$ ，则产生越界中断，否则继续执行。（注意：页号是从0开始的，而页表长度至少是1，因此 $P=M$ 时也会越界）
3. 页表中页号P对应的页表项地址=页表起始地址F+页号P*页表项长度，取出该页表项内容b，即为内存块号。（注意区分页表项长度、页表长度、页面大小的区别。页表长度指的是这个页表中总共有几个页表项，即总共有几个页；页表项长度指的是每个页表项占多大的存储空间；页面大小指的是一个页面占多大的存储空间）
4. 计算 $E=b*L+W$ ，用得到的物理地址E去访存。（如果内存块号、页面偏移量是用二进制表示的，那么把二者拼接起来就是最终的物理地址了）

动手验证：假设页面大小 $L=1KB$ ，最重要访问的内存块号 $b=2$ ，页内偏移量 $W=1023$ 。

1. 尝试用 $E=b*L+W$ 计算目标物理地址。
2. 尝试把内存块号、页内偏移量用二进制表示，并把它们拼接起来得到物理地址。对比1、2的结果是否一致

若页面大小 L 为1K字节，页号2对应的内存块号 $b=8$ ，将逻辑地址 $A=2500$ 转换为物理地址 E 。

等价描述：某系统按字节寻址，逻辑地址结构中，页内偏移量占10位，页号2对应的内存块号 $b=8$ ，将逻辑地址 $A=2500$ 转换为物理地址 E 。

说明一个页面的大小为

$$2^{10} B = 1KB$$

1. 计算页号、页内偏移量

$$\text{页号 } P = A/L = 2500/1024 = 2$$

$$\text{页内偏移量 } W = A \% L = 2500 \% 1024 = 452$$

2. 根据题中条件可知，页号2没有越界，其存放的内存块号 $b=8$

$$\text{3. 物理地址 } E = b * L + W = 8 * 1024 + 452 = 8644$$

在分页存储管理（页式管理）的系统中，只要确定了每个页面的大小，逻辑地址结构就确定了。因此，页式管理中地址是一维的。即，只要给出一个逻辑地址，系统就可以自动地算出页号、页内偏移量，并不需要显示地告诉系统这个逻辑地址中，页内偏移量占多少位。

对页表项大小的进一步探讨

每个页表项的长度是相同的，页号是“隐含”的

假设某系统内存大小为4GB，页面大小为4KB，物理内存总共会被分为

$$2^{32}/2^{12} = 2^{20}$$

个内存块，因此内存块号的范围应该是

$$[0, 2^{20} - 1]$$

因此至少要20个二进制位才能表示这么多的内存块号，因此至少要3个字节才够（每个字节8个二进制位，3个字节共24个二进制位）

页号	块号
0	3字节
1	3字节
.....	3字节
n	3字节

页表

各页表项会按顺序连续地存放在内存中

如果该页表在内存中存放的起始地址为X，则M号页对应的页表项是存放在内存地址为X+3*M

一个页面为4KB，则每个页框可以存放 $4096/3=1365$ 个页表项，但是这个页框会剩余 $4096\%3=1$ B页内碎片

因此，1365号页表项存放的地址为X+3*1365+1

如果每个页表项占4个字节，则每个页框刚好可存放1024个页表项

1024号页表项虽然是存放在下一个页框中的，但是它的地址依然可以用X+4*1024得出

结论：理论上，页表项长度为3B即可表示内存块号的范围，但是，为了方便页表的查询，常常会让一个页表项占更多的字节，使得每个页面恰好可以装得下整个页表项。

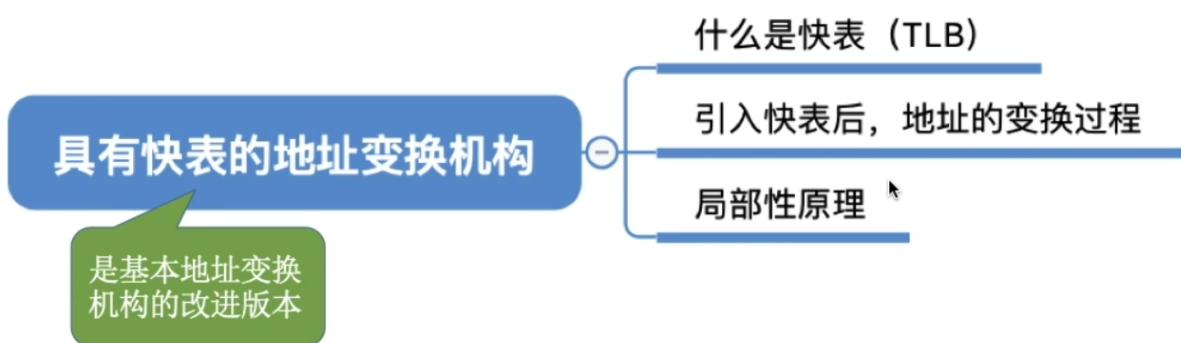


进程页表通常是装在连续的内存块中的

具有快表的地址变换机构

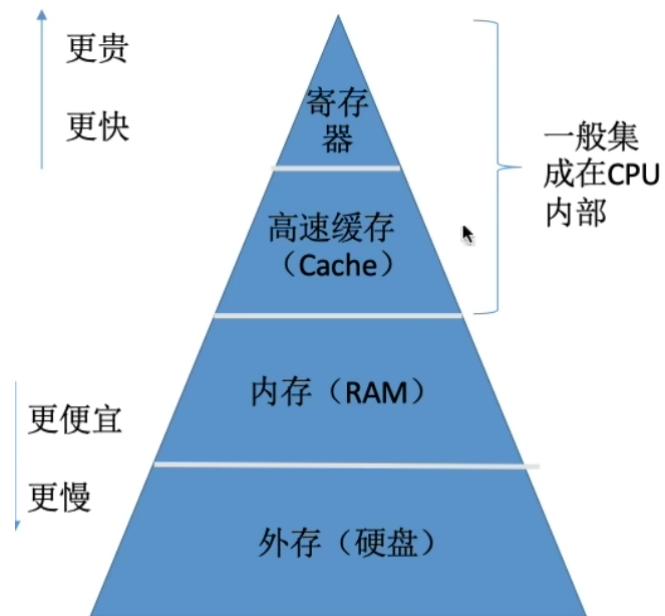
	地址变换过程	访问一个逻辑地址的访存次数
基本地址变换机构	①算页号、页内偏移量 ②检查页号合法性 ③查页表，找到页面存放的内存块号 ④根据内存块号与页内偏移量得到物理地址 ⑤访问目标内存单元	两次访存
具有快表的地址变换机构	①算页号、页内偏移量 ②检查页号合法性 ③查快表。若命中，即可知道页面存放的内存块号，可直接进行⑤；若未命中则进行④ ④查页表，找到页面存放的内存块号，并且将页表项复制到快表中 ⑤根据内存块号与页内偏移量得到物理地址 ⑥访问目标内存单元	快表命中，只需一次访存 快表未命中，需要两次访存

TLB和普通Cache的区别——TLB中只有页表项的副本，而普通Cache中可能会有其他各种数据的副本



什么是快表 (TLB)

快表，又称联想寄存器 (TLB, translation lookaside buffer)，是一种访问速度比内存快很多的高速缓存 (TLB不是内存！)，用来存放最近访问的页表项的副本，可以加速地址变换的速度，于此对应，内存中的页表常称为慢表。

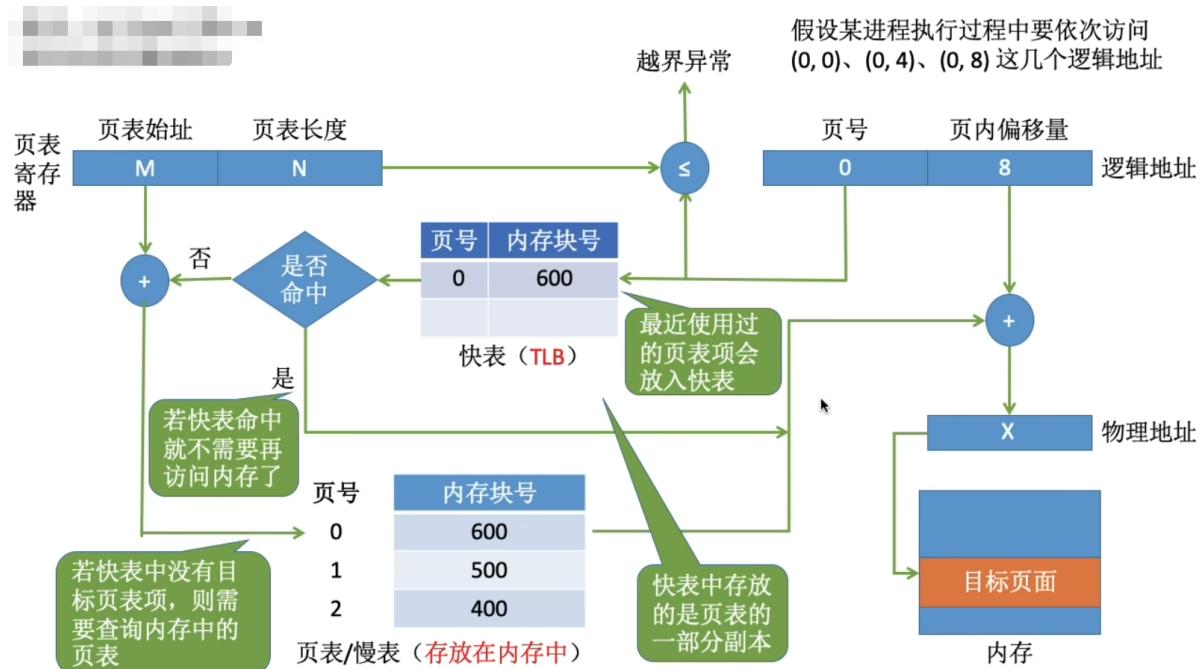


思考：能否把整个页表都放在TLB中？

贵

假设：访问TLB只需1微秒

访问内存需要100微秒



引入快表后，地址的变换过程

1. CPU给出逻辑地址，由某个硬件算得页号、页内偏移量，将页号与快表中的所有页号进行比较。
2. 如果找到匹配的页号，说明要访问的页表项在快表中有副本，则直接从中取出该页对应的内存块号，再将内存块号与页内偏移量拼接形成物理地址，最后，访问该物理地址对应的内存单元。因此，若快表命中，则访问某个逻辑地址仅需一次访存即可。
3. 如果没有找到匹配的页号，则需要访问内存中的页表，找到对应页表项，得到页面存放的内存块号，再将内存块号与页内偏移量拼接形成物理地址，最后，访问该物理地址对应的内存单元。因此，若快表未命中，则访问某个逻辑地址需要两次访存（注意：在找到页表项后，应同时将其存入快表，以便后面可能的再次访问。但若快表已满，则必须按照一定的算法对旧的页表项进行替换）

由于查询快表的速度比查询页表的速度快很多，因此只要快表命中，就可以节省很多时间。

由于局部性原理，一般来说快表的命中率可以达到90%以上。

例：某系统使用基本分页存储管理，并采用了具有快表的地址变换机构。访问一次快表耗时1微秒，访问一次内存耗时100微秒。若快表的命中率为90%，那么访问一个逻辑地址的平均耗时是多少？

$$(1 + 100) * 0.9 + (1 + 100 + 100) * 0.1 = 111\mu s$$

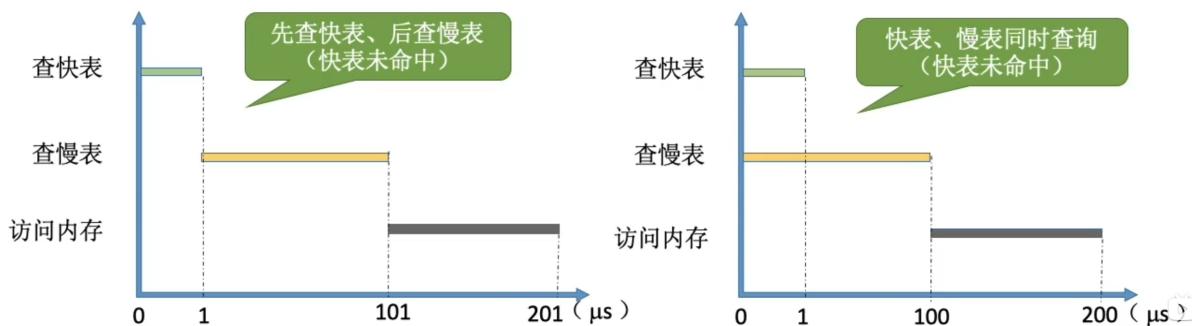
有的系统支持快表和慢表同时查找，如果是这样，平均耗时应该是

$$(1 + 100) * 0.9 + (100 + 100) * 0.1 = 110.9\mu s$$

若未采用快表机制，则访问一个逻辑地址需要

$$100 + 100 = 200\mu s$$

显然，引入快表机制后，访问一个逻辑地址的速度快多了。



局部性原理

```
int i=0;
int a[100];
while(i<100){
    a[i]=i;
    i++;
}
```

这个程序执行时，会很频繁地访问10号页面、23号页面



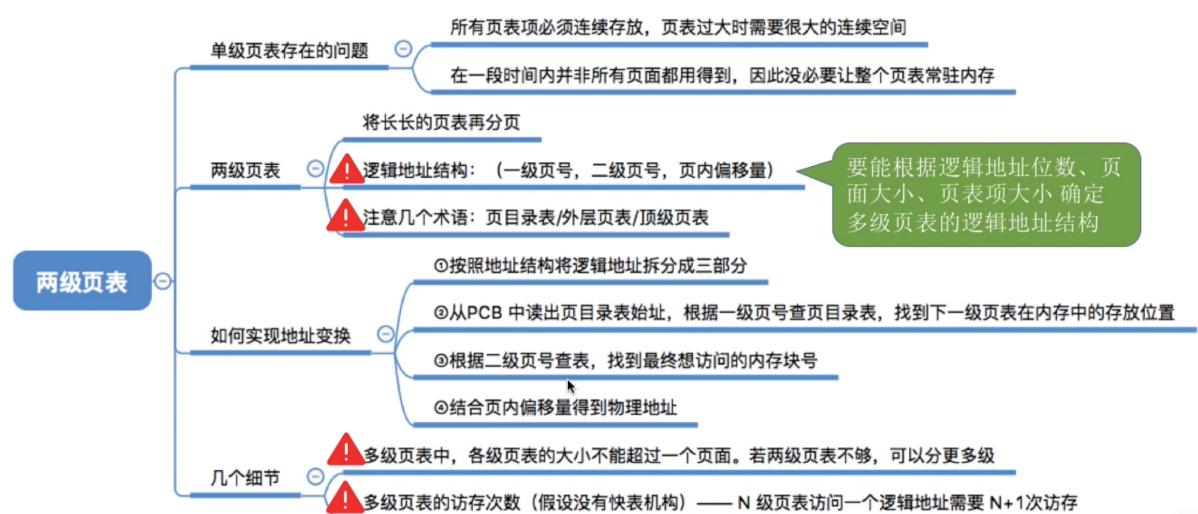
内存

时间局部性：如果执行了程序中的某条指令，那么不久后这条指令很有可能再次执行；如果某个数据被访问过，不久之后该数据很可能再次被访问。（因为程序中存在大量的循环）

空间局部性：一旦程序访问了某个存储单元，在不久之后，其附近的存储单元也很有可能被访问。（因为很多数据在内存中都是连续存放的）

基本地址变换机构中，每次要访问一个逻辑地址，都需要查询内存中的页表。由于局部性原理，可能连续很多次查到的都是同一个页表项

两级页表



单级页表存在什么问题？如何解决？

两级页表

两级页表的原理、逻辑地址结构

如何实现地址变换？

两级页表问题需要注意的几个细节

单级页表存在的问题

某计算机系统按字节寻址，支持32位的逻辑地址，采用分页存储管理，页面大小为4KB，页表项长度为4B。

$$4KB = 2^{12}B$$

因此页内地址要用12位表示，剩余20位表示页号。

因此，该系统中用户进程最多有

$$2^{20} \text{页}$$

相应的，一个进程的页表中，最多会有

$$2^{20} = 1M = 1048576 \text{个页表项}$$

所以一个页表最大需要

$$2^{20} * 4B = 2^{22}B$$

共需要

$$2^{22}/2^{12} = 2^{10}$$

个页框存储该页表。

根据页号查询页表的方法：

$$K \text{号页对应的页表项存放位置} = \text{页表始址} + K * 4$$

要在所有的页表项都连续存放的基础上才能用这种方法找到页表项

需要专门给进程分配

$$2^{10} = 1024 \text{个连续的页框}$$

来存放它的页表

根据局部性原理可知，很多时候，进程在一段时间内只需要访问某几个页面就可以正常运行了。因此没有必要让整个页表都常驻内存。

如何解决单级页表的问题

问题一：页表必须连续存放，因此当页表很大时，需要占用很多个连续的页框。

问题二：没有必要让整个页表常驻内存，因为进程在一段时间内可能只需要访问某几个特定的页面。

思考：我们是如何解决进程在内存中必须连续存储的问题？

将进程地址空间分页，并为其建立一张页表，记录各页面的存放位置

同样的思路也可用于解决“页表必须连续存放”的问题，把必须连续存放的页表再分页

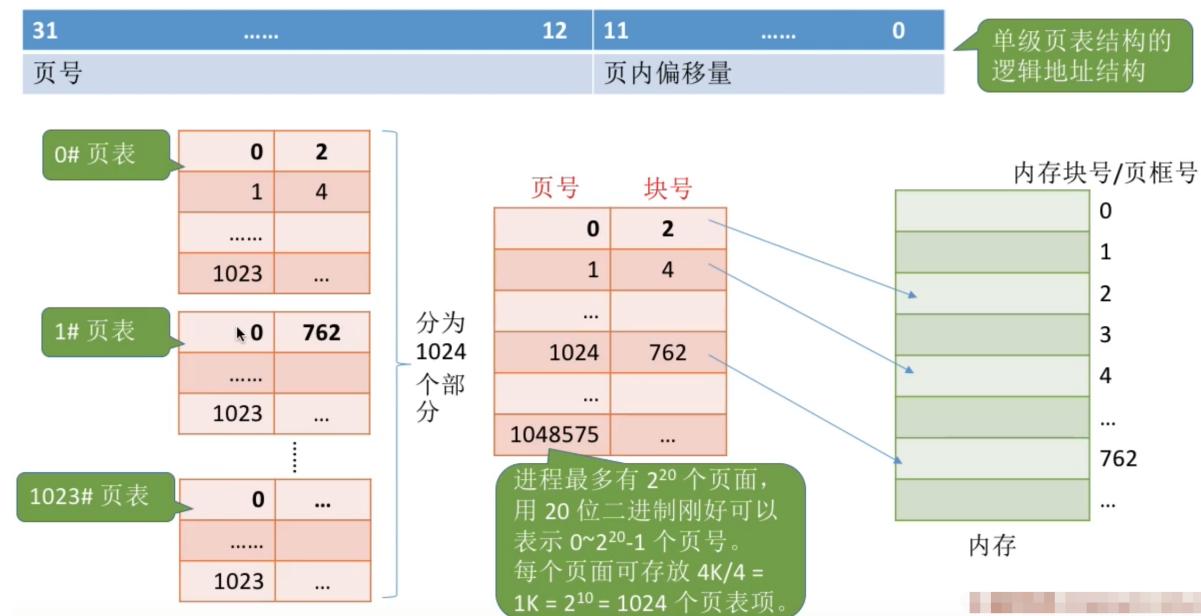
可将长长的页表进行分组，使每个内存块刚好可以放入一个分组

比如上个例子中，页面大小4KB，每个页表项4B，每个页面可存放1K个页表项，因此每1K个连续的页表项为一组，每组刚好占一个内存块，再将各组离散地放到各个内存块中

另外，再为离散分配的页表再建立一张页表，称为页目录表，或称外层页表，或称顶层页表

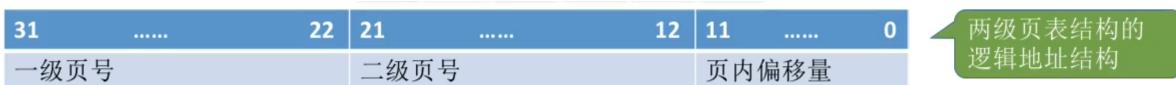
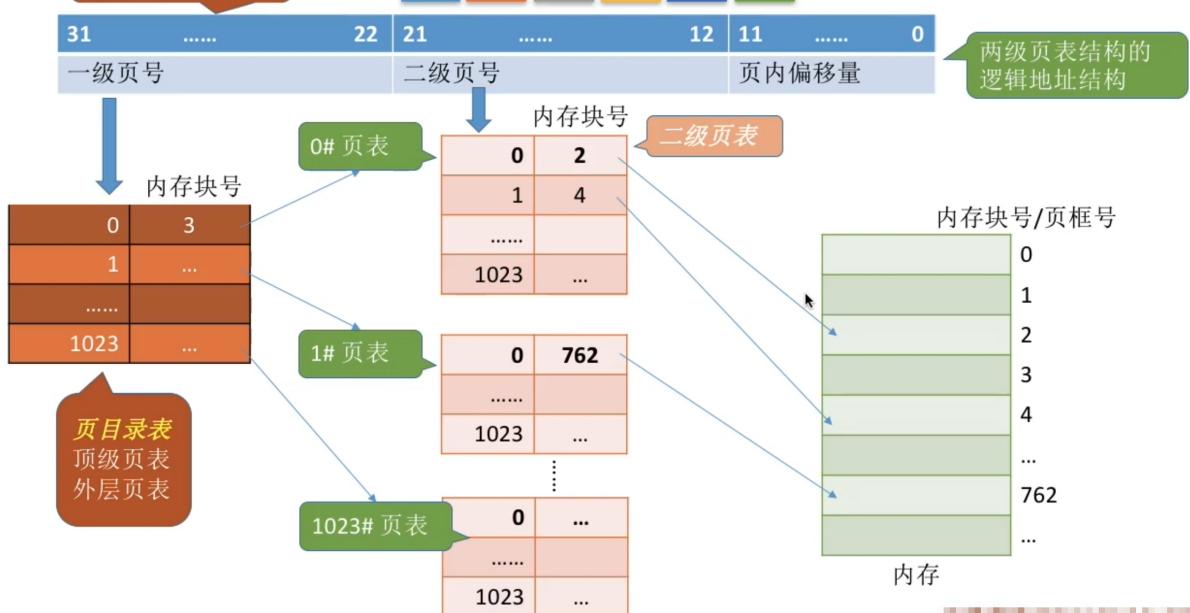
两级页表的原理、地址结构（问题一）

32位逻辑地址空间，页表项大小为4B，页面大小为4KB，则页内地址占12位

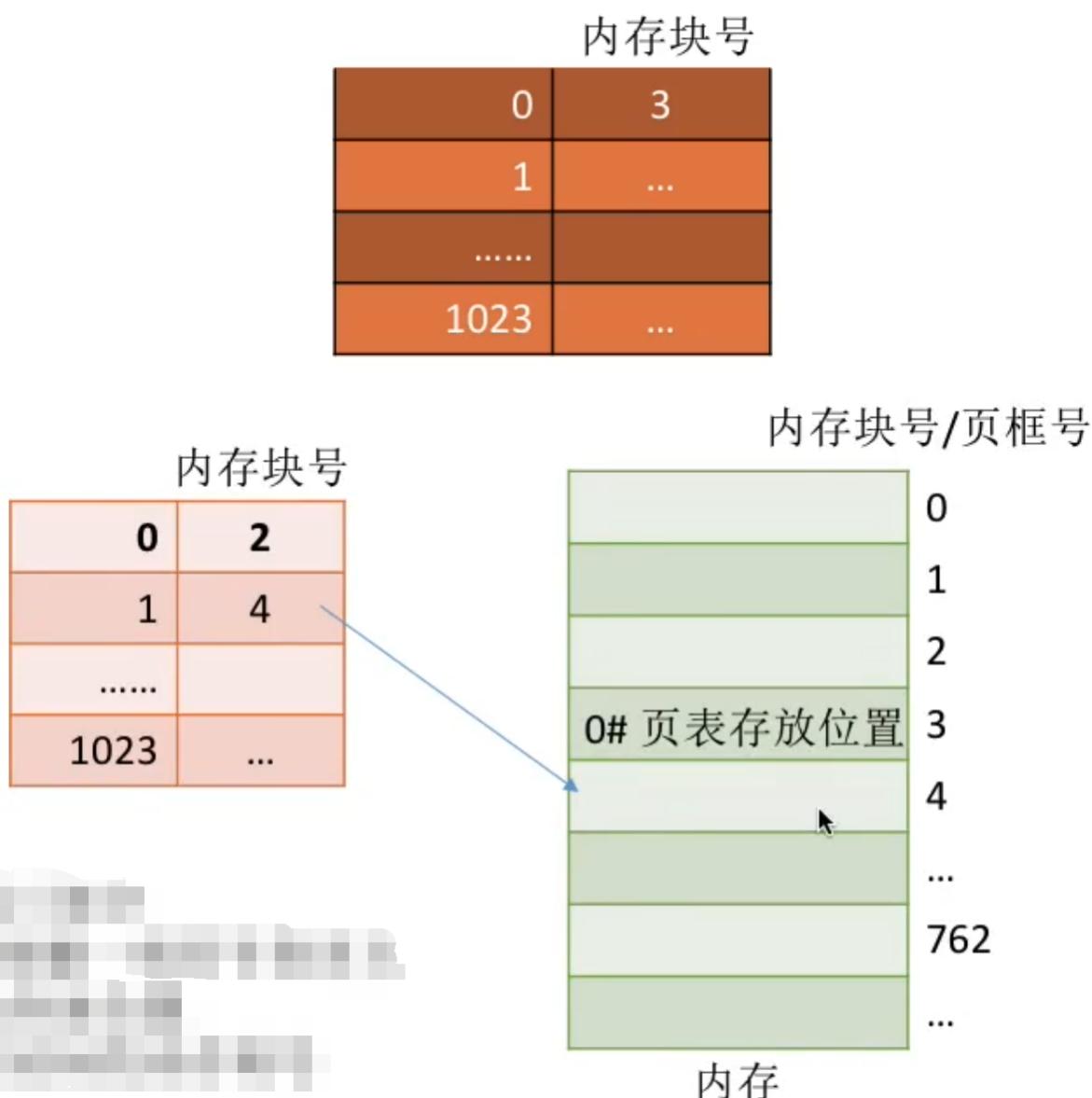


道地址 11 页号刚
好可表示 0~1023

两级页表的原理、地址结构



例：将逻辑地址(0000000000,0000000001,111111111111)转化为物理地址 (用十进制表示)



1. 按照地址结构将逻辑地址拆分成三部分
2. 从PCB中读出页目录表始址，再根据一级页号查页目录表，找到下一级页表在内存中的存放位置
3. 根据二级页号查表，找到最终想访问的内存块号
4. 结合页内偏移量得到物理地址

最终要访问的内存块为4

该内存块的起始地址为 $4 \times 4096 = 16384$

页内偏移量为1023

最终的物理地址为

$16384 + 1023 = 17407$

(问题二)

可以在需要访问页面时才把页面调入内存（虚拟存储技术）。可以在页表项中增加一个标志位，用于表示该页面是否已经调入内存



需要注意的几个细节

1. 若采用多级页表机制，则各级页表的大小不能超过一个页面

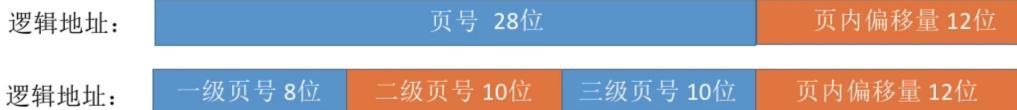
例：某系统按字节编址，采用40位逻辑地址，页面大小为4KB，页表项大小为4B，假设采用纯页式存储，则要采用()级页表，页内偏移量为()位？

页面大小 = $4KB = 2^{12}B$ ，按字节编址，因此页内偏移量为12位

页号 = $40 - 12 = 28$ 位

页面大小 = $2^{12}B$ ，页表项大小 = 4B，则每个页面可存放 $2^{12}/4 = 2^{10}$ 个页表项

因此各级页表最多包含 2^{10} 个页表项，需要10位二进制位才能映射到 2^{10} 个页表项，因此每一级的页表对应页号应为10位。总共28位的页号至少要分为三级



如果只分为两级页表，则一级页号占18位，也就是说页目录表中最多可能有 2^{18} 个页表项，显然，一个页面是放不下这么多页表项的。

2. 两级页表的访存次数分析（假设没有快表机构）

第一次访存：访问内存中的页目录表

第二次访存：访问内存中的二级页表

第三次访存：访问目标内存单元