@[toc]  


# 基本地址变换机构

* 重点理解、记忆基本地址变换结构（用于实现逻辑地址到物理地址转换的一组硬件机构）的原理和流程
* 基本地址变换结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。

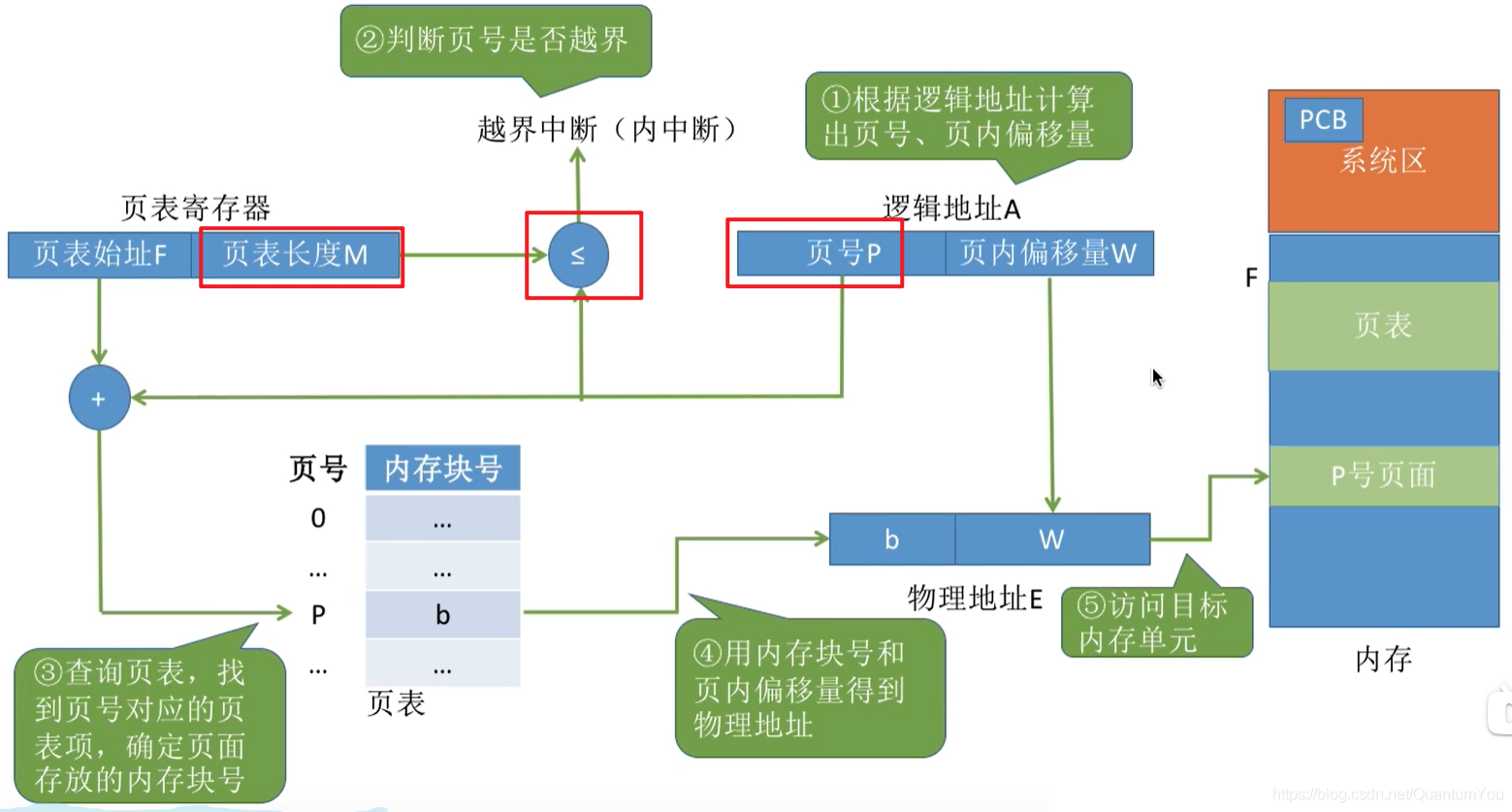
注意：

* 页面大小（页面偏移量）是2的整数幂
* 页表长度---> 页表项的个数，所以需要将页表长度与页号进行比较
* 每个页表项的长度是相同的，页号是“隐含”

[第三个参考链接](https://blog.csdn.net/QuantumYou/article/details/118651568?spm=1001.2014.3001.5501)

## 逻辑地址物理地址变换过程

* 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：

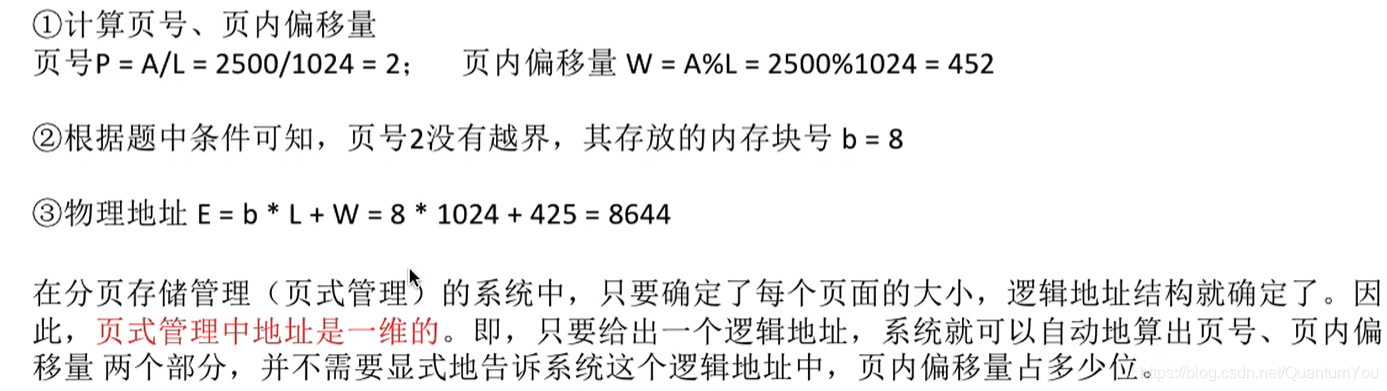
一、图示描述  


二、文字描述：

* 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。
* 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，最终要访问的内存块号b=2，页内偏移量W=1023.①尝试用E=bL+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①②的结果是否一致

## 实际运用

* 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（说明一个页面大小为210B=1KB），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.



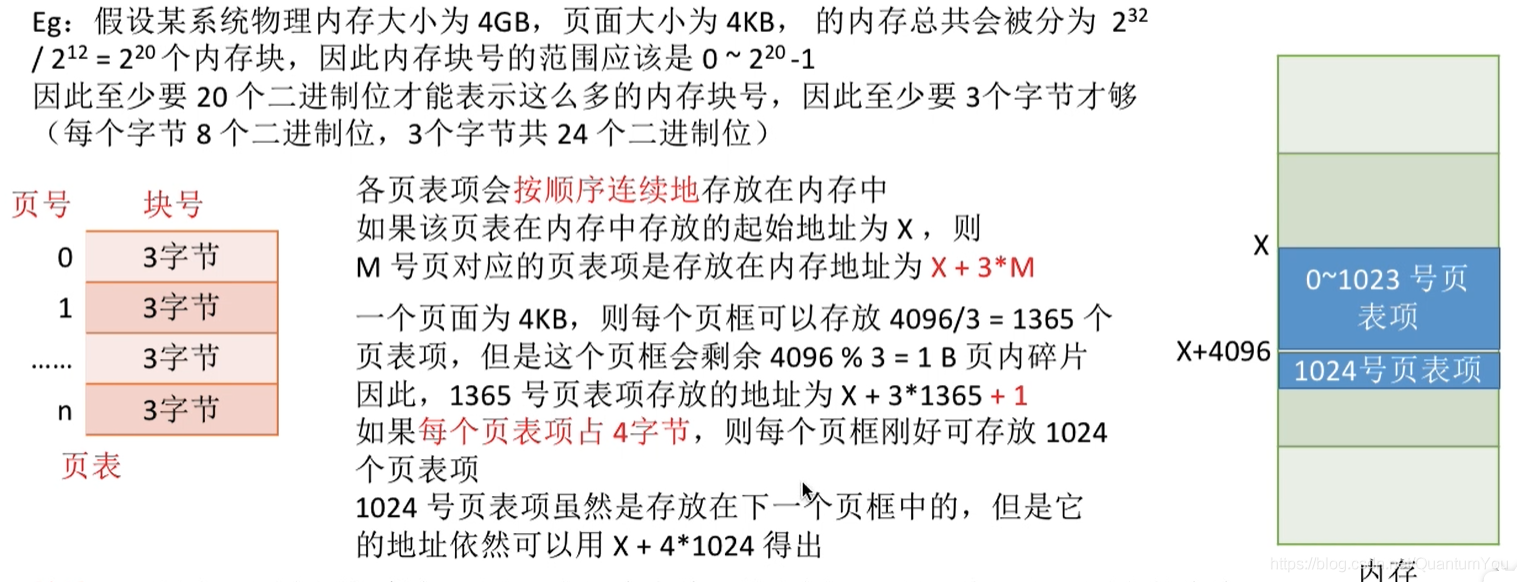
## 易混淆概念

* 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**”，或称“页帧”、“内存块”、“物理块”.每个页框有一个编号，即“**页框号**”（或者“内存块号”、“页帧号”、“物理块号”）页框号从0开始。
* 将用户进程的地址空间也分为与页框大小相等的一个个区域称为“**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。（注：进程的最后一个页面可能没有一个页框那么大。因此，页框不能太大，否则可能产生过大的内部碎片）
* 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，也不必按先后顺序来，可以放到不相邻的各个页框中。

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

### 页表项解读



* **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数个页表项
* 进程页表通常是装在连续的内存块中的

## 小结

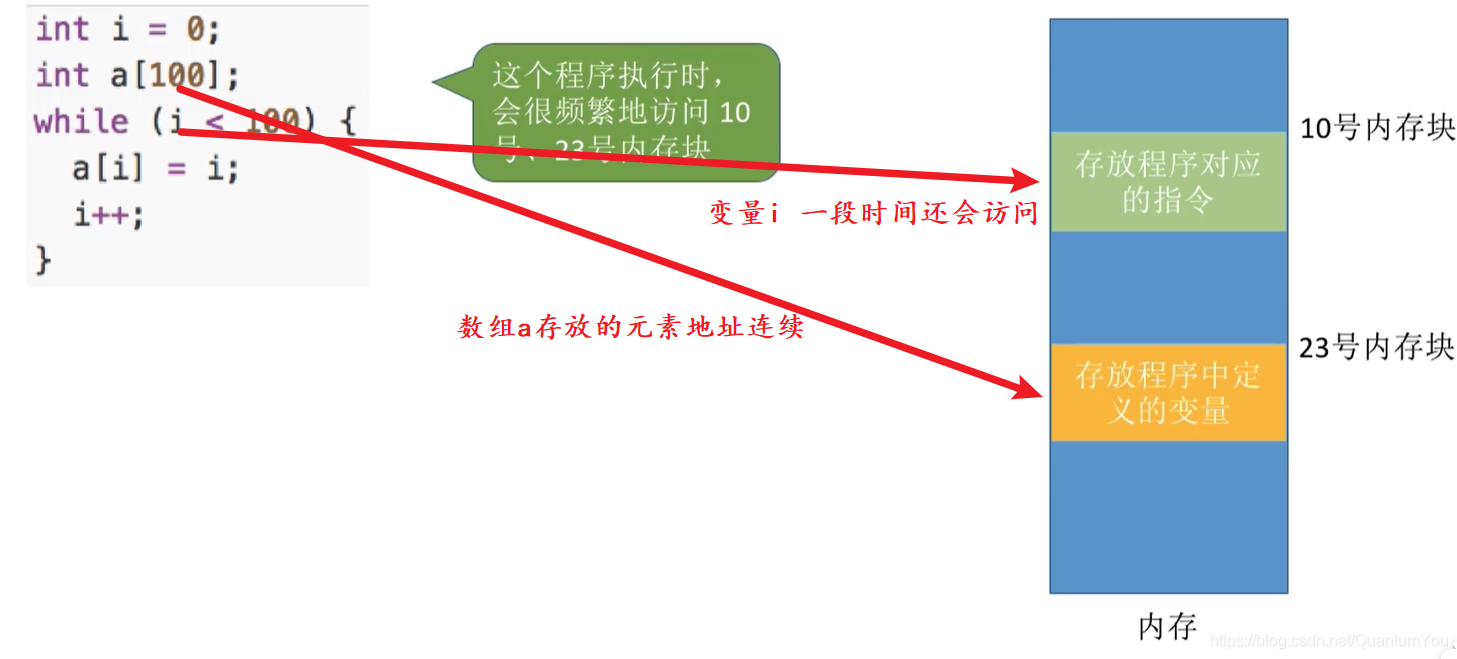


# 具有快表的地址变换机构

* 是基本地址变换机构的改进版本

## 局部性原理

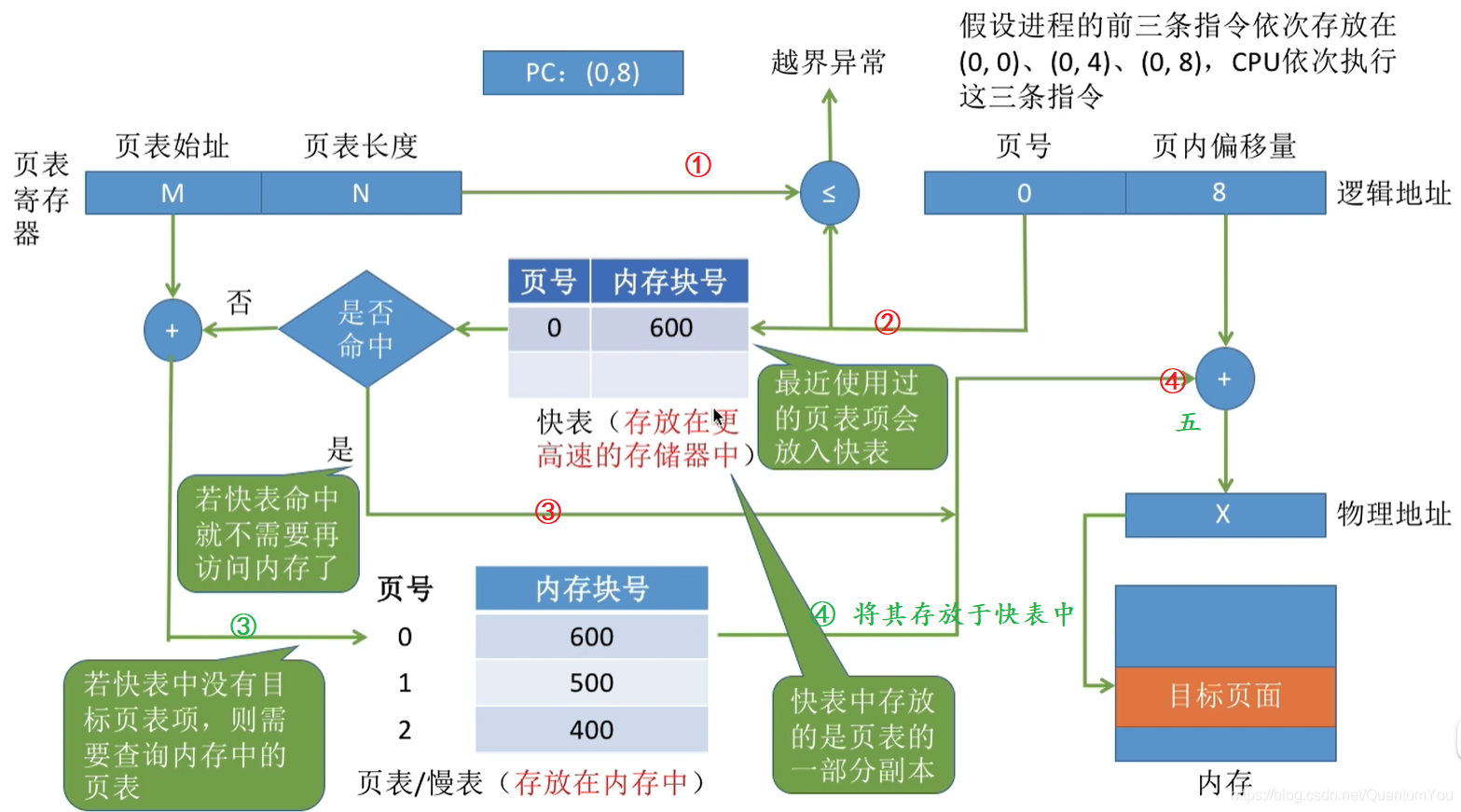
* **时间局部性**：如果执行了程序中的某条指令，那么不久后这条指令很有可能再次执行；如果某个数据被访问过，不久之后该数据很可能再次被访问。（因为程序中存在大量的循环）
* **空间局部性**：一旦程序访问了某个存储单元，在不久之后，其附近的存储单元也很有可能被访问。（因为很多数据在内存中都是连续存放的）



## 快表（TLB）

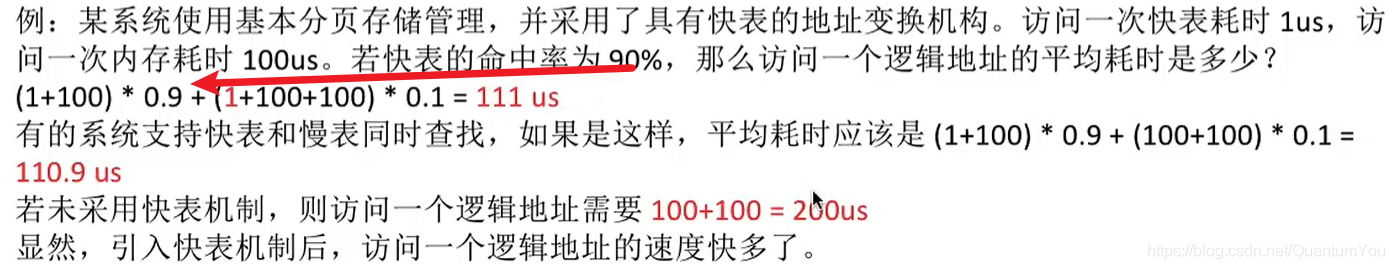
问题提出

* 在基本地址变换机构中，每次要访问一个逻辑地址，都需要査询内存中的页表。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。既然如此，能香利用这个特性减少访问页表的次数呢？
* **快表的定义**：快表，又称联想寄存器（TLB），是一种访问速度比内存快很多的高速缓冲存储器，用来存放当前访问的若干页表项，以加速地址变换的过程。与此对应，内存中的页表常称为慢表。

  
文字描述：

* ①CPU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(慢表)，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）

因为局部性原理，一般来说快表的命中率可以达到90%以上



## 小结

