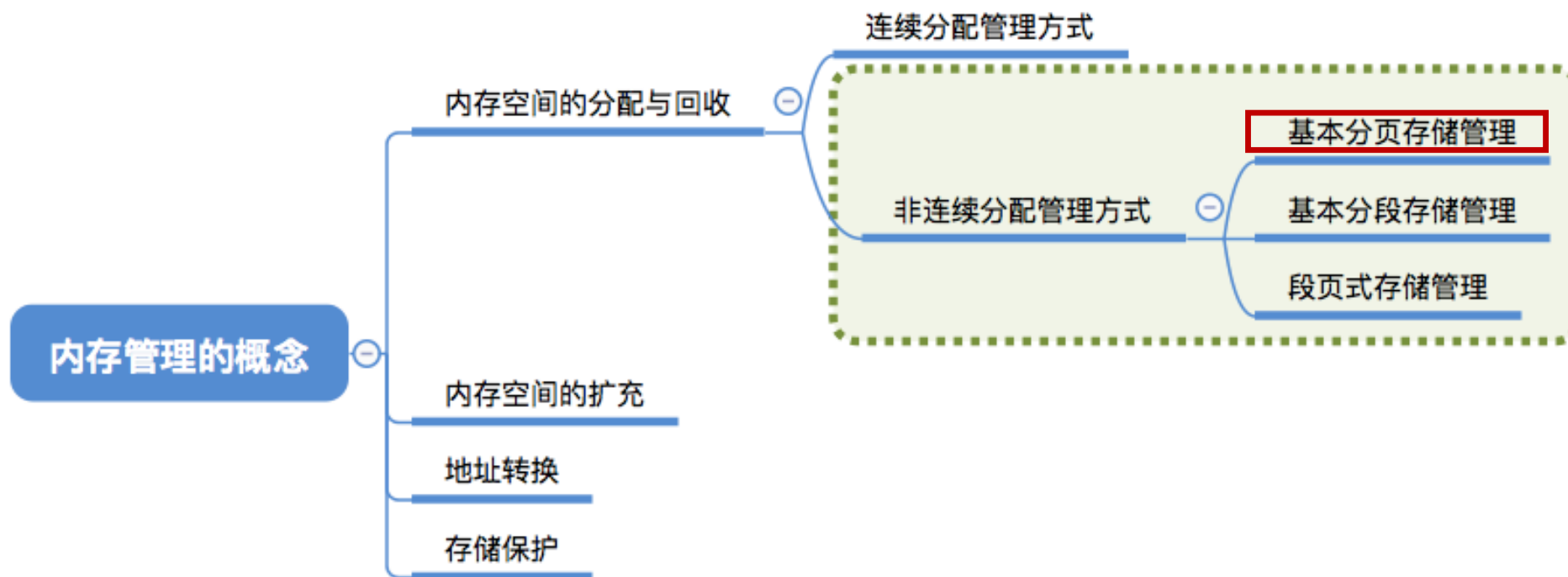


本节内容

基本地址变换机构

知识总览



结合上一小节理解基本地址变换机构（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

基本地址变换机构

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放到页表寄存器中。

页表寄存器

注意：页面大小是2的整数幂

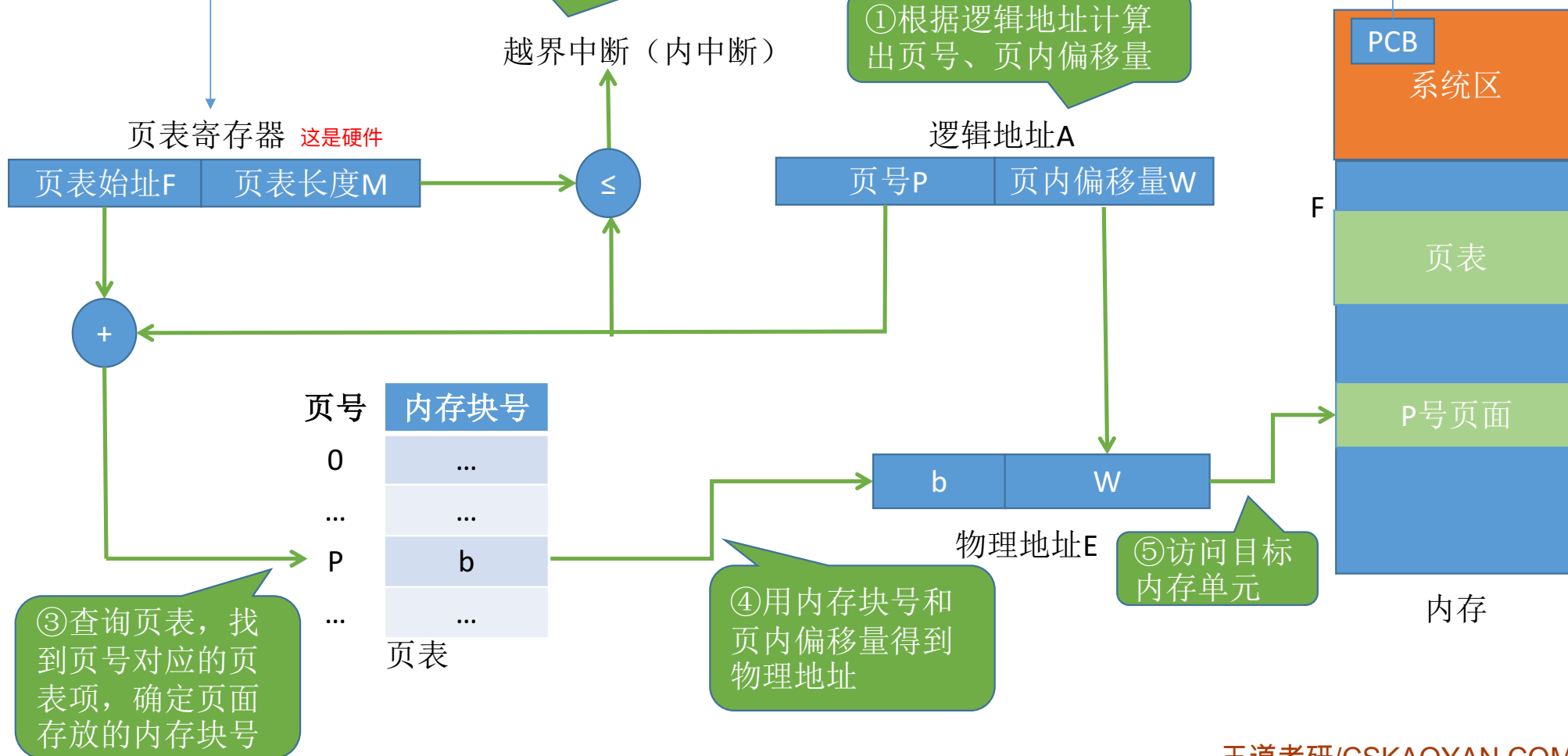
设页面大小为L，逻辑地址A到物理地址E的变换过程如下：

程序计数器PC：指向下一条指令的逻辑地址A

进程切换相关的内核程序负责恢复进程运行环境

②判断页号是否越界

①根据逻辑地址计算出页号、页内偏移量



基本地址变换机构

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和页内偏移量）

②比较页号P和页表长度M，若 $P \geq M$ ，则产生越界中断；否则，P是合法的页号，从页表开始查找，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 + P * 页表项长度**，即为内存块号。（注意区分**页表项长度**、**页表长度**和**页面大小**：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用的存储空间；**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 $L = 1\text{KB}$ ，最终要访问的内存块号 $b = 2$ ，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①②的结果是否一致

基本地址变换机构

例：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 $2^{10} \text{ B} = 1\text{KB}$

①计算页号、页内偏移量

页号 $P = A/L = 2500/1024 = 2$ ； 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 $b = 8$

③物理地址 $E = b * L + W = 8 * 1024 + 425 = 8644$

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、页内偏移量两个部分，并不需要显式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

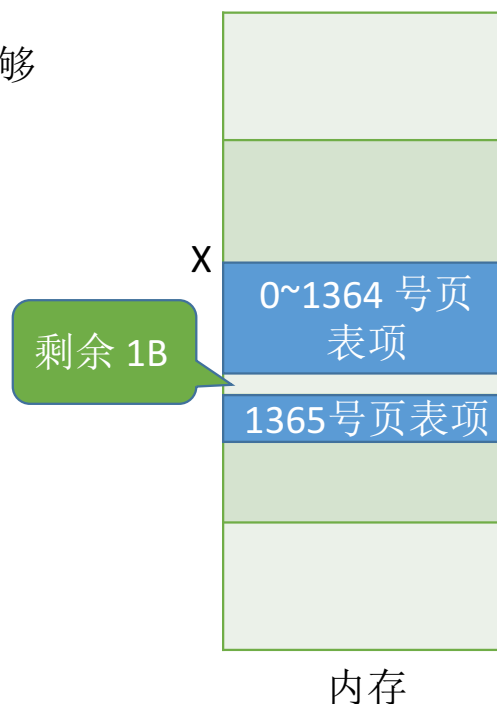
Eg: 假设某系统物理内存大小为 4GB，页面大小为 4KB，的内存总共会被分为 $2^{32} / 2^{12} = 2^{20}$ 个内存块，因此内存块号的范围应该是 $0 \sim 2^{20}-1$
因此至少要 20 个二进制位才能表示这么多的内存块号，因此至少要 3 个字节才够
(每个字节 8 个二进制位，3 个字节共 24 个二进制位)

| 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 $4096 \% 3 = 1 \text{ B}$ 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 字节，则每个页框刚好可存放 1024 个页表项



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

Eg: 假设某系统物理内存大小为 4GB，页面大小为 4KB，的内存总共会被分为 $2^{32} / 2^{12} = 2^{20}$ 个内存块，因此内存块号的范围应该是 $0 \sim 2^{20}-1$
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

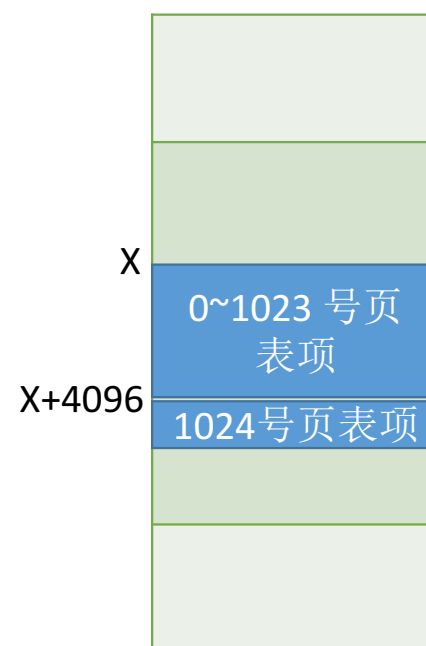
M 号页对应的页表项是存放在内存地址为 $X + 3 * M$

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 $4096 \% 3 = 1 \text{ B}$ 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 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

进程页表通常是装在连续的内存块中的



内存

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。

知识回顾与重要考点

