本节内容

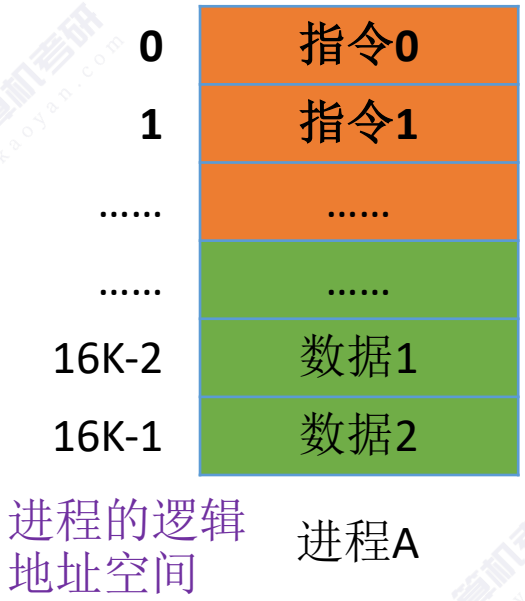
基本分页存储管理的基本概念

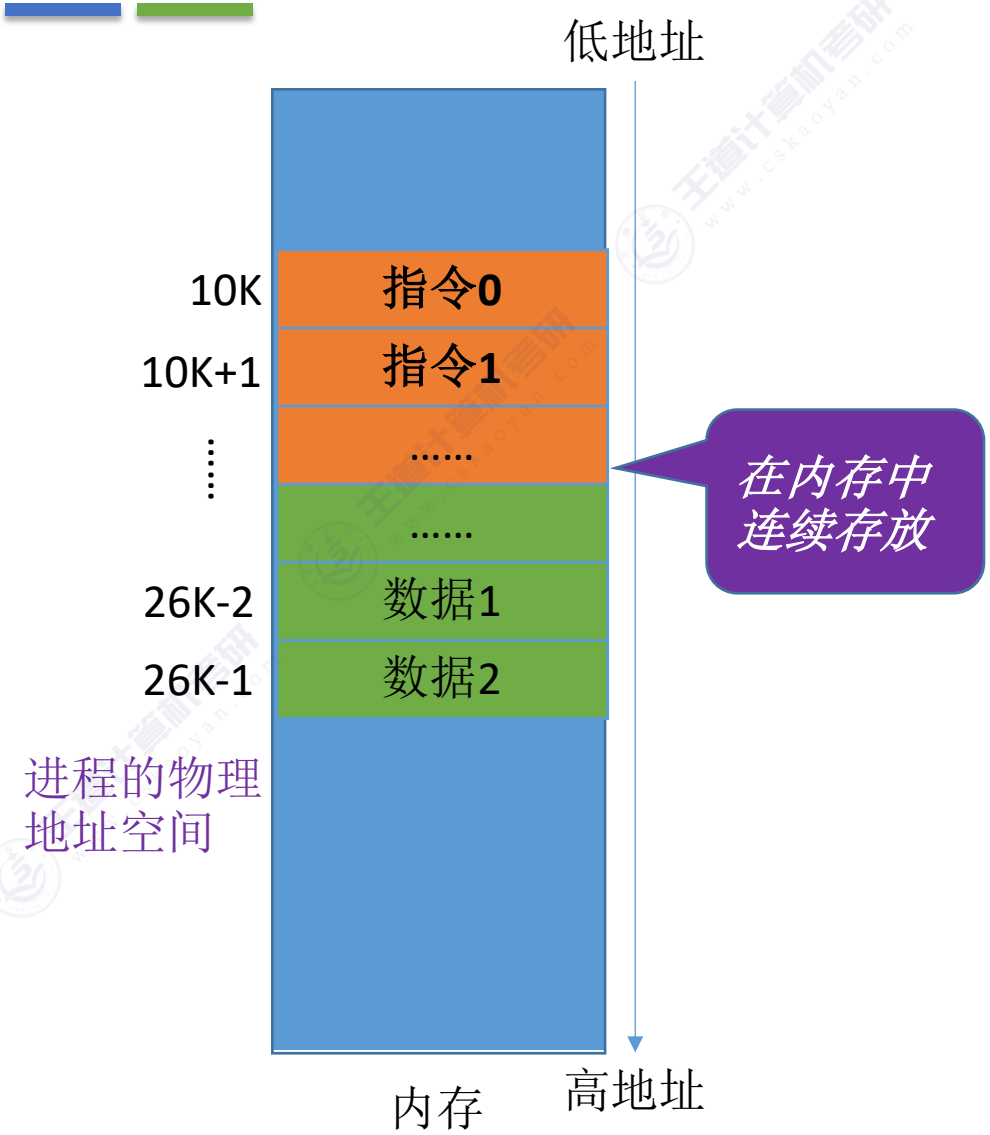
知识总览

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“地址空间”

概念回顾：
逻辑地址（相对地址）
物理地址（绝对地址）





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*。

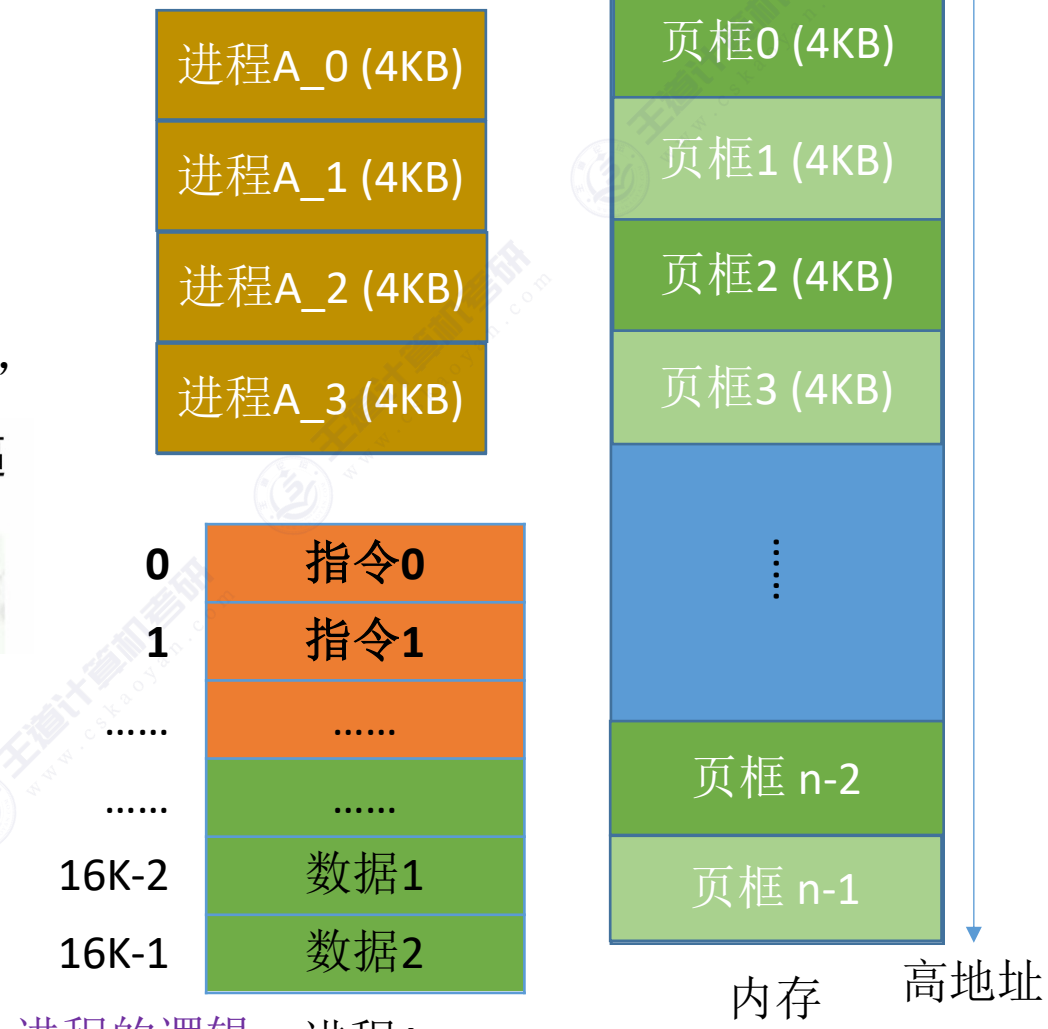
将进程的逻辑地址空间也分为*与页框大小相等*的一个个部分，每个部分称为一个“页”或“页面”。每个页面也有一个编号，即“页号”，页号也是*从0开始*。

一脸懵逼

Tips：初学易混——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

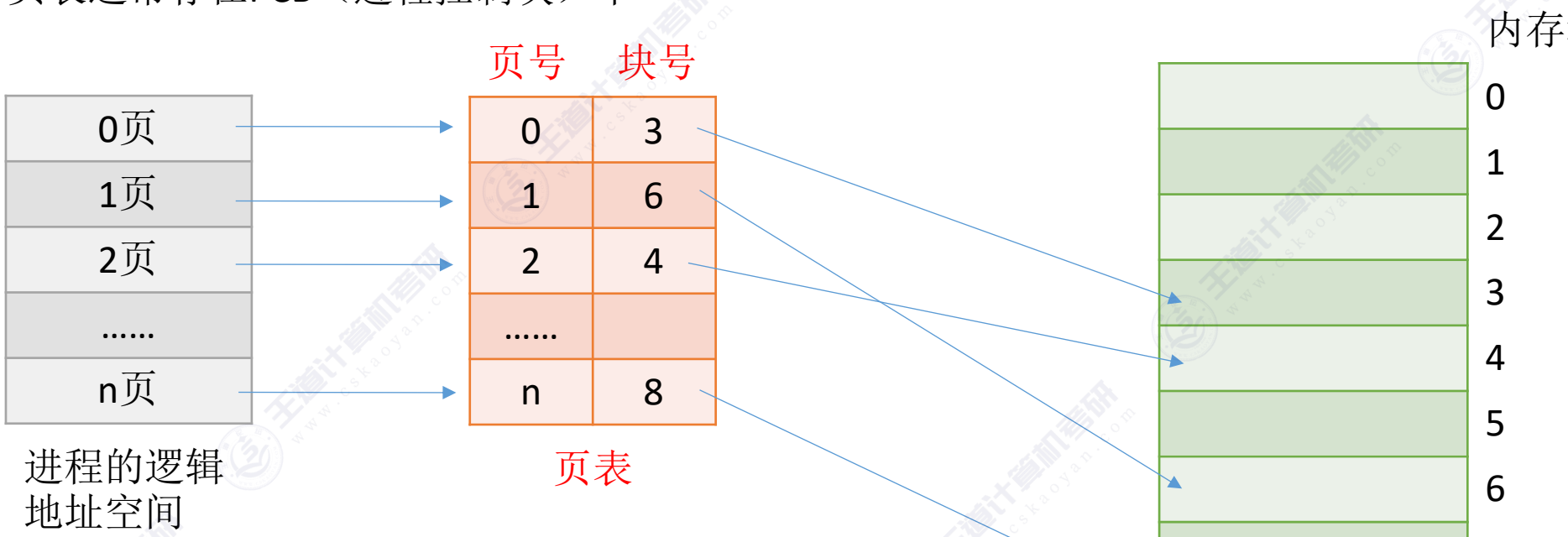
操作系统以页框为单位为各个进程分配内存空间。进程的每个页面分别放入一个页框中。也就是说，进程的页面与内存的页框有一一对应的关系。
各个页面不必连续存放，可以放到不相邻的各个页框中。

（注：进程的最后一个页面可能没有一个页框那么大。也就是说，分页存储有可能产生内部碎片，因此页框不能太大，否则可能产生过大的内部碎片造成浪费）



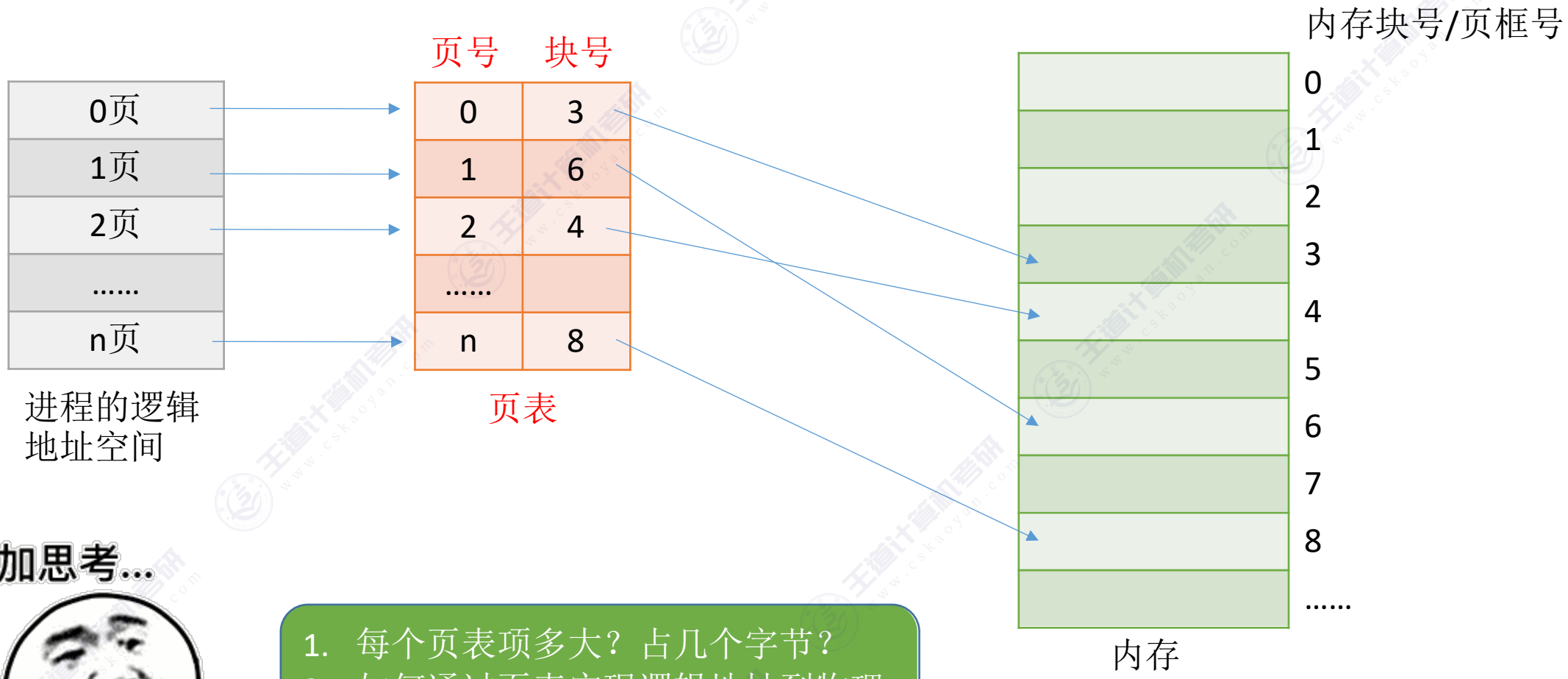
重要的数据结构——页表

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
5. 每个页表项的长度是相同的

重要的数据结构——页表

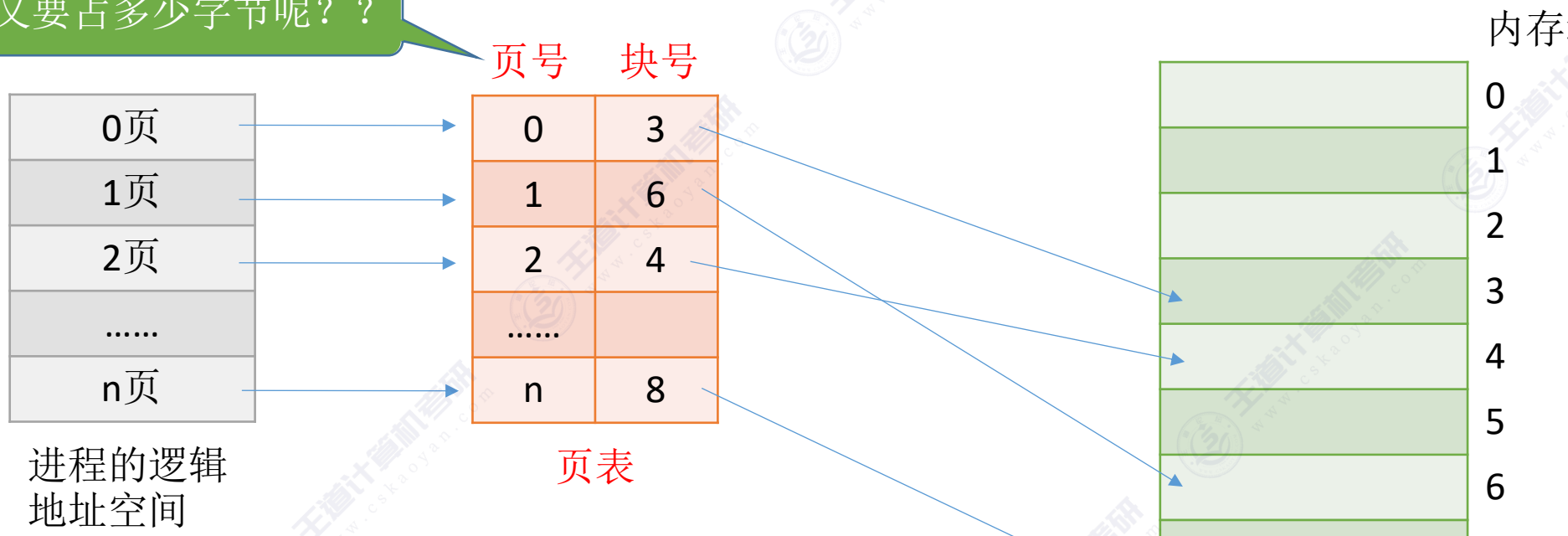


稍加思考...

1. 每个页表项多大？占几个字节？
2. 如何通过页表实现逻辑地址到物理地址的转换？

问题一：每个页表项占多少字节？

那页号又要占多少字节呢？？



Eg：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？

内存块大小=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

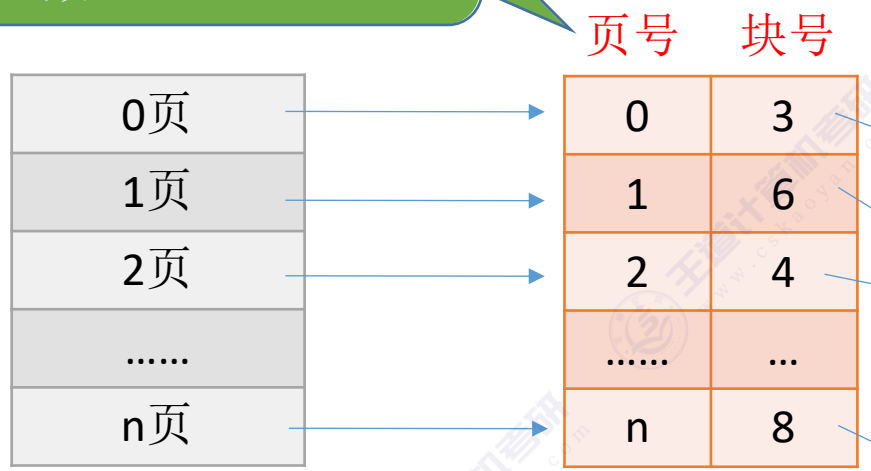
→内存块号至少要用 20 bit 来表示

→至少要用3B来表示块号（3*8=24bit）

重要重要重要考点：
计算机中内存块的数量→页表项中块号至少占多少字节

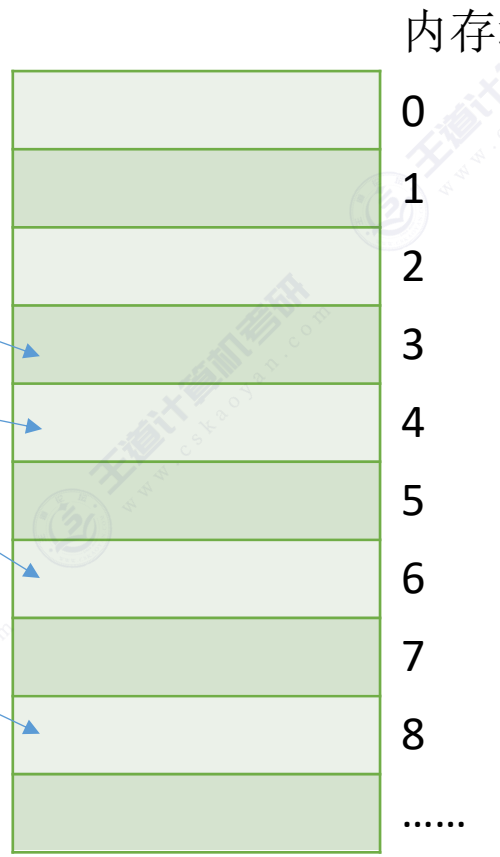
问题一：每个页表项占多少字节？

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）

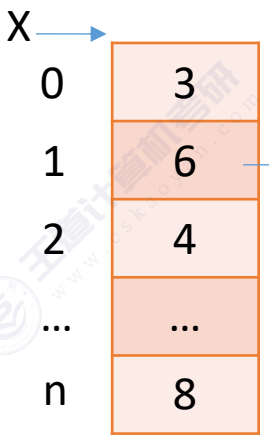


进程的逻辑地址空间

页表



内存



每个页表项占3B，且都是连续存放

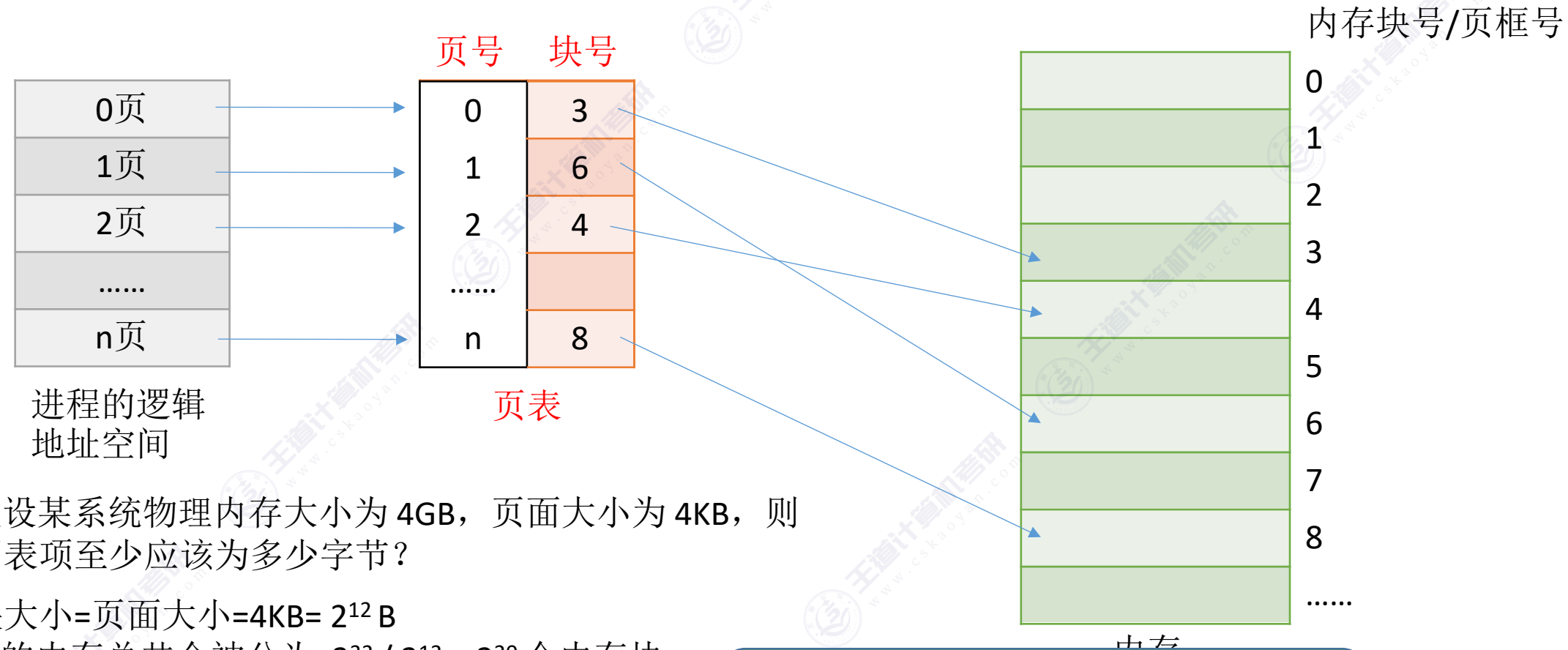
假设页表中的各页表项从内存地址为 X 的地方开始连续存放…

如何找到页号为 i 的页表项？

i 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

问题一：每个页表项占多少字节？



Eg：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？

内存块大小=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 20 bit 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 号内存块的起始地址 = J * 内存块大小

问题二：如何实现地址的转换

*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？*

重定位寄存器：指明了进程在内存中的起始位置

100

CPU

+

目标逻辑地址：80

相对于起始位置的“偏移量”

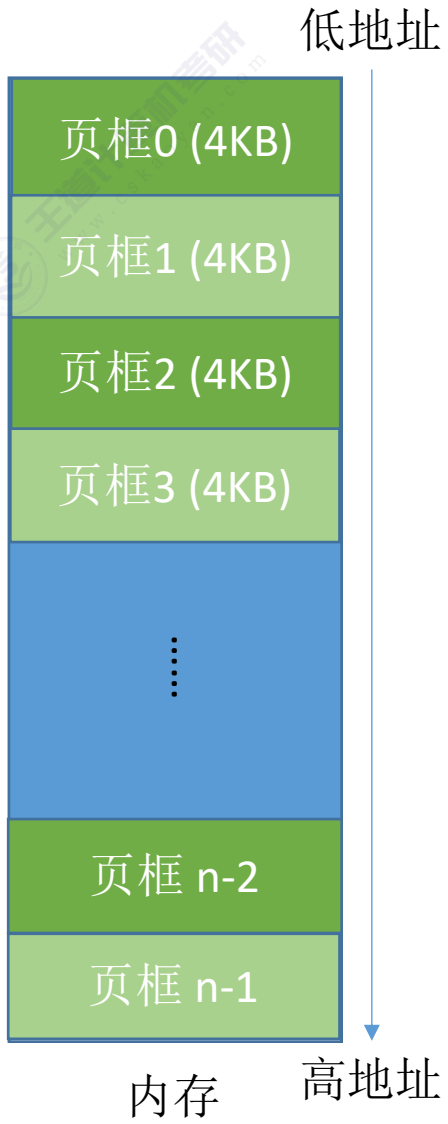
内存

物理地址（绝对地址）

		100
100	**指令1**：往地址为 80 的存储单元中写入 1	
101	**指令2**：	
…	…	
180	1	
181		
…	…	
279	…	279

问题二：如何实现地址的转换

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？



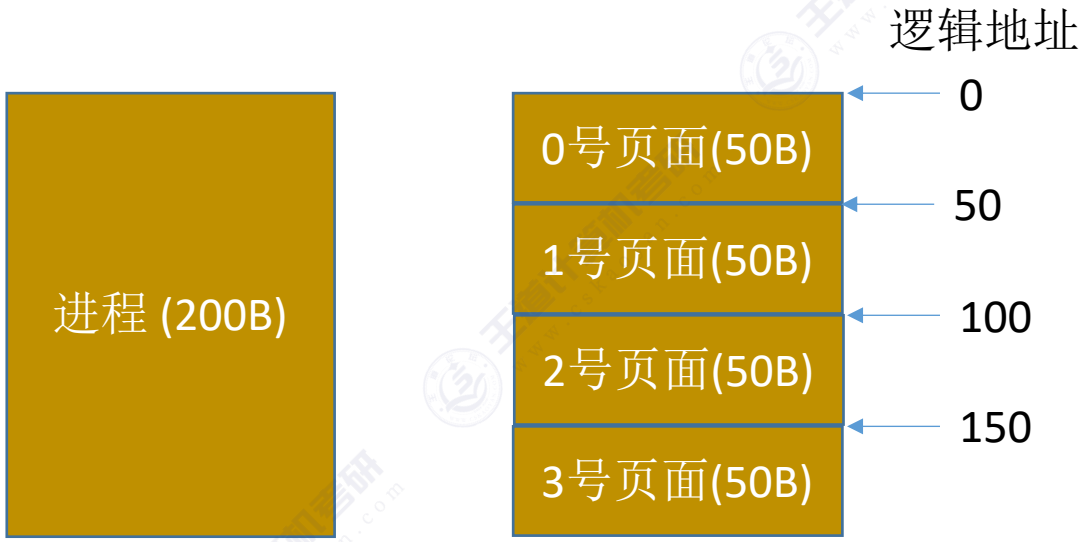
特点：虽然进程的各个页面是离散存放的，但是页面内部是连续存放的

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

逻辑地址A 对应的物理地址 = P号页面在内存中的起始地址+页内偏移量W

子问题：如何确定一个逻辑地址对应的页号、页内偏移量？

Eg：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？



如何计算：

页号 = 逻辑地址 / 页面长度 （取除法的整数部分）

页内偏移量 = 逻辑地址 % 页面长度（取除法的余数部分）

页号 = 110 / 50 = 2

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

页面在内存中的起始地址+页内偏移量 = 实际的物理地址

子问题：如何确定一个逻辑地址对应的页号、页内偏移量？

页号 = 逻辑地址 / 页面长度 （取除法的整数部分）
页内偏移量 = 逻辑地址 % 页面长度（取除法的余数部分）

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速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：
00000000000000000000000000000000 ~ 00000000000000000000111111111111

1号页的逻辑地址范围应该是 4096~8191，用二进制表示应该是：
00000000000000000001000000000000 ~ 00000000000000000001111111111111

2号页的逻辑地址范围应该是 8192~12287，用二进制表示应该是：
00000000000000000010000000000000 ~ 00000000000000000010111111111111

结论：如果每个页面大小为 2^KB，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

Eg：逻辑地址 2，用二进制表示应该是 00000000000000000000000000000010
页号 = 2/4096 = 0 = 00000000000000000000，页内偏移量 = 2%4096 = 2 = 000000000010

Eg：逻辑地址 4097，用二进制表示应该是 00000000000000000001000000000001
页号 = 4097/4096 = 1 = 00000000000000000001，页内偏移量 = 4097%4096 = 1 = 000000000001

子问题：如何确定一个逻辑地址对应的页号、页内偏移量？

页号 = 逻辑地址 / 页面长度 （取除法的整数部分）
页内偏移量 = 逻辑地址 % 页面长度（取除法的余数部分）

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速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

Eg：逻辑地址 4097，用二进制表示应该是 00000000000000000001000000000001
页号 = 4097/4096 = 1 = 00000000000000000001，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 00000000000000000001000000000000
2号内存块的起始物理地址是 00000000000000000010000000000000
3号内存块的起始物理地址是 00000000000000000011000000000000

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
=（00000000000000001001000000000001）

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

子问题：为何页面大小要取2的整数幂？

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速的把逻辑地址拆分成（页号，页内偏移量）

如何计算：
页号 = 逻辑地址 / 页面长度 （取除法的整数部分）
页内偏移量 = 逻辑地址 % 页面长度（取除法的余数部分）

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^KB，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。

Tips：学有余力的同学建议看看二进制数的运算（计组内容），才更能理解本质原因

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。在上图所示的例子中，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 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 个页面

重要重要重要！！！
页面大小 ↔ 页内偏移量位数
➔逻辑地址结构

Tips：有些奇葩题目中页面大小有可能不是2的整数次幂，这种情况还是得用最原始的方法计算：
页号 = 逻辑地址 / 页面长度 （取除法的整数部分）
页内偏移量 = 逻辑地址 % 页面长度（取除法的余数部分）

知识回顾与重要考点

基本分页存储管理的基本概念

- 基本分页存储管理的思想：把进程分页，各个页面可离散地放到各个的内存块中
- 易混概念
  - “页框、页帧、内存块、物理块、物理页” VS “页、页面”
  - “页框号、页帧号、内存块号、物理块号、物理页号” VS “页号、页面号”
- 页表
  - 页表记录了页面和实际存放的内存块之间的映射关系
  - 一个进程对应一张页表，进程的每一页对应一个页表项，每个页表项由“页号”和“块号”组成
  - 每个页表项的大小是相同的，页号是“隐含”的
  - i 号页表项存放地址 = 页表始址 + i * 页表项大小

内存块的数量 ➔ 页表项大小

- 逻辑地址结构——可拆分为【页号P，页内偏移量W】
  - 页号 = 逻辑地址/页面大小；页内偏移量 = 逻辑地址%页面大小
  - 如果页面大小刚好是2的整数次幂呢?
- 如何实现地址转换
  1. 计算出逻辑地址对应的【页号，页内偏移量】
  2. 找到对应页面在内存中的存放位置（查页表）
  3. 物理地址 = 页面始址 + 页内偏移量

公众号：王道在线

b站：王道计算机教育

抖音：王道计算机考研