计算机组成原理

虚拟存储器

王浩宇,教授

haoyuwang@hust.edu.cn

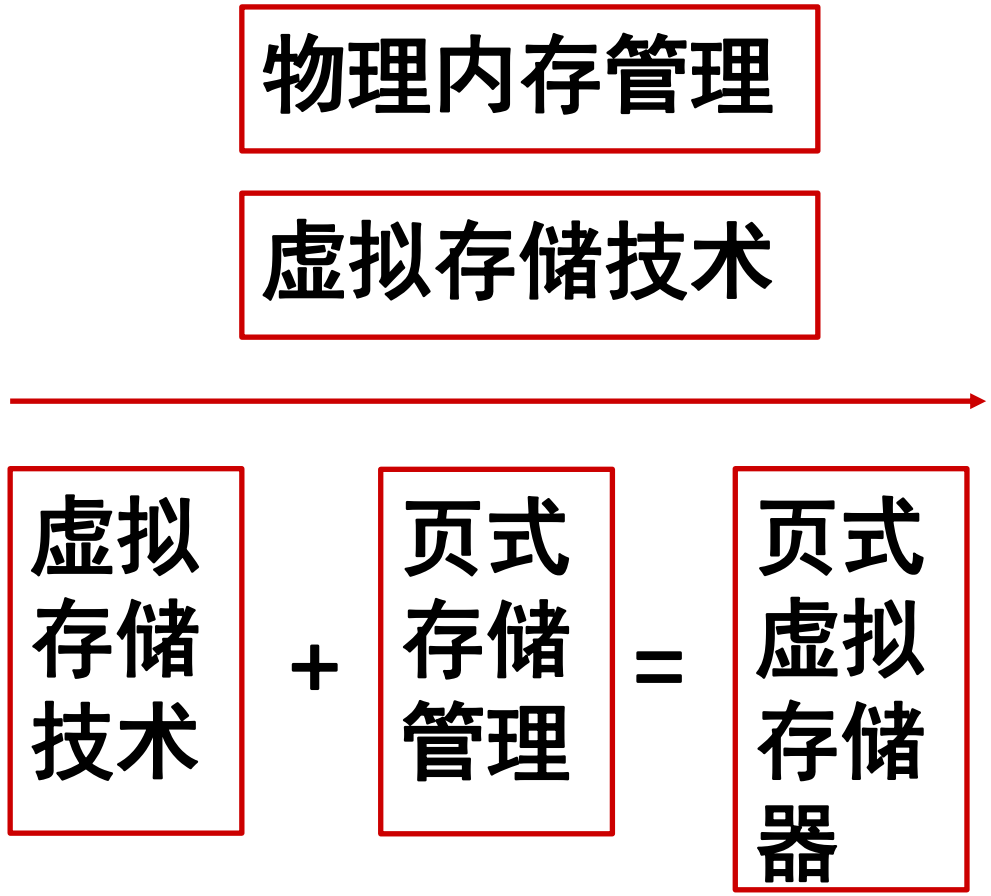
https://howiepku.github.io/

本节目录

- **虚拟存储器**
  - 内存管理方案（部分内容与操作系统课程有重叠，便于理解）
  - 虚拟存储技术（重点）

虚拟存储器主要内容

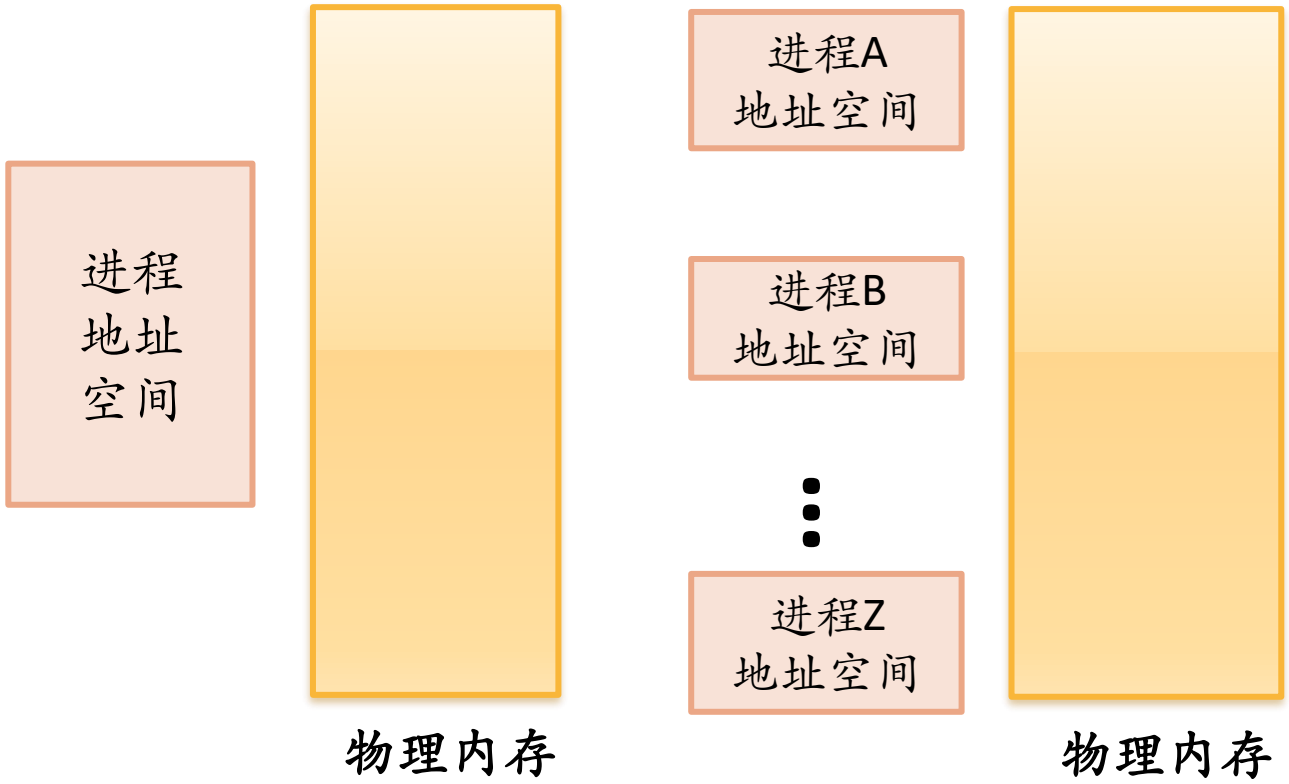
- 基本概念
- 物理内存管理
  - 基本内存管理方案
- 虚拟存储技术
  - 页式虚拟存储器
  - 段式虚拟存储器
  - 段页式虚拟存储器
- 软件相关策略
  - 页面置换算法



背景知识

- 程序装载到内存才可以运行
  - 通常，程序以可执行文件格式保存在磁盘上
- 多道程序设计模型
  - 允许多个程序同时进入内存
- 每个进程有自己的地址空间
  - 一个进程执行时不能访问另一个进程的地址空间
  - 进程不能执行不适合的操作

要解决的问题



如何把进程地址空间的内容装载到内存，合理分配内存，使得每一个进程都能够正确的执行？

讨论

- **进程中的地址不是最终的物理地址**
- **在进程运行前无法计算出物理地址**

因为：不能确定进程被加载到内存什么地方

→→ 需要　地址重定位　的支持

地址转换、地址变换、地址翻译、地址映射　Translation、Mapping

地址重定位

- 逻辑地址（相对地址，虚拟地址）

  用户程序经过编译、汇编后形成目标代码，目标代码通常采用相对地址的形式，其首地址为0，其余地址都相对于首地址而编址

  不能用逻辑地址在内存中读取信息

- 物理地址（绝对地址，实地址）

  内存中存储单元的地址　　可直接寻址

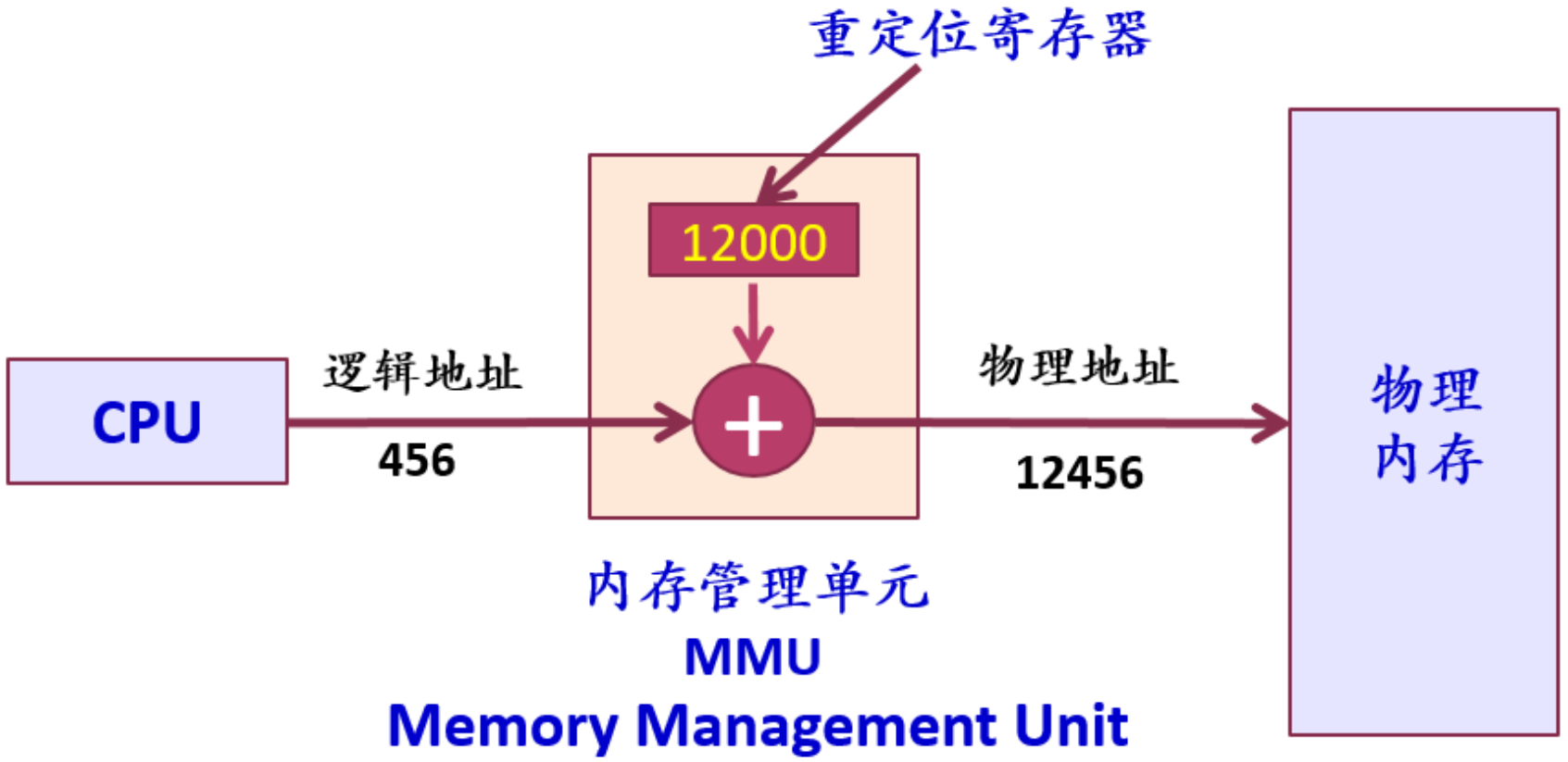
为了保证CPU执行指令时可正确访问内存单元，需要将用户程序中的逻辑地址转换为运行时可由机器直接寻址的物理地址，这一过程称为地址重定位

静态重定位与动态重定位

静态重定位：
当用户程序加载到内存时，一次性实现逻辑地址到物理地址的转换
一般可以由软件完成

动态重定位：
在进程执行过程中进行地址变换
即逐条指令执行时完成地址转换
需要硬件部件支持

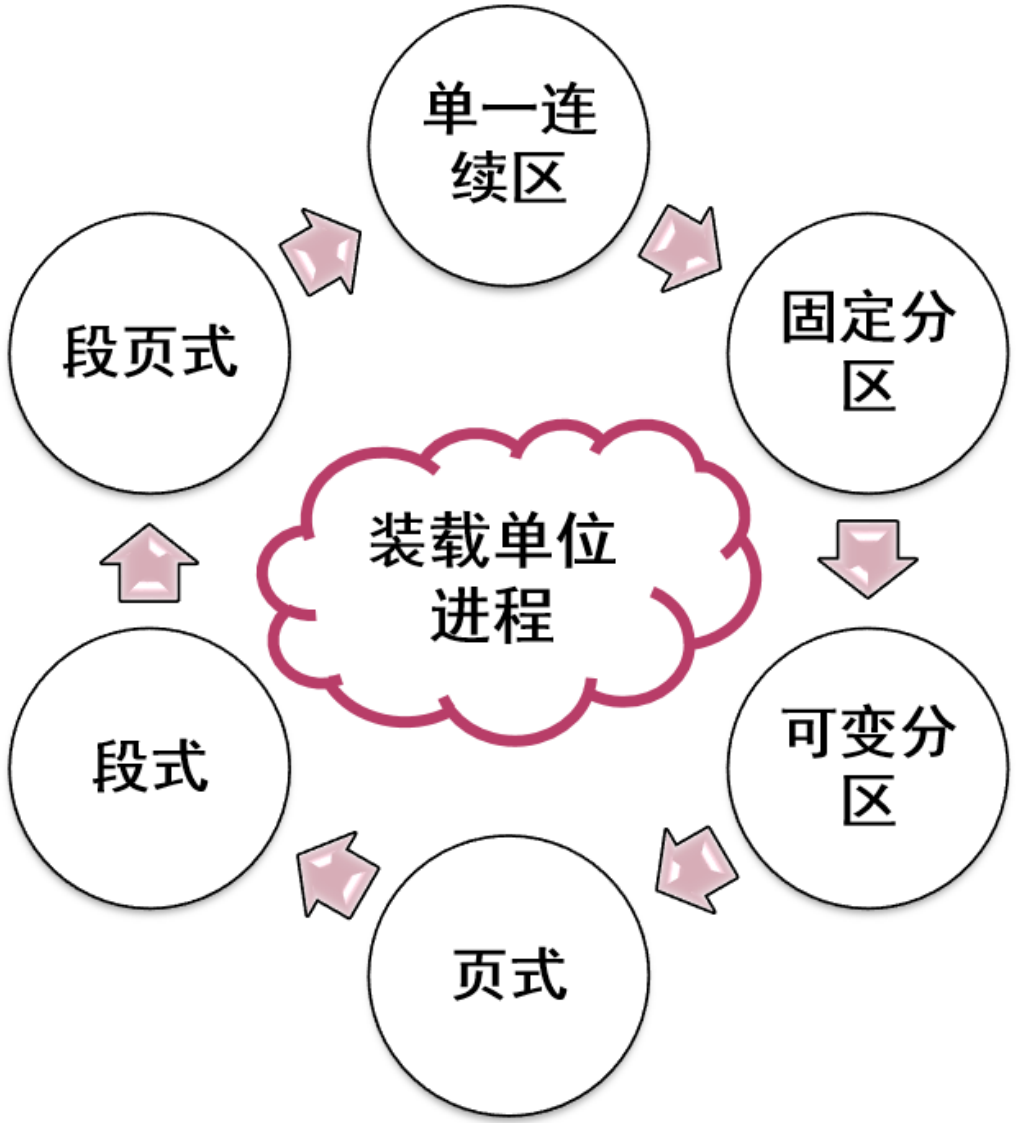
动态重定位实现


重定位寄存器
12000
CPU
逻辑地址
456
物理地址
12456
物理
内存
内存管理单元
MMU
Memory Management Unit

讨论

- 动态重定位　　逻辑地址->物理地址
- 但如何给进程分配物理内存？

基本内存管理方案



单一连续区

特点：一段时间内只有一个进程在内存

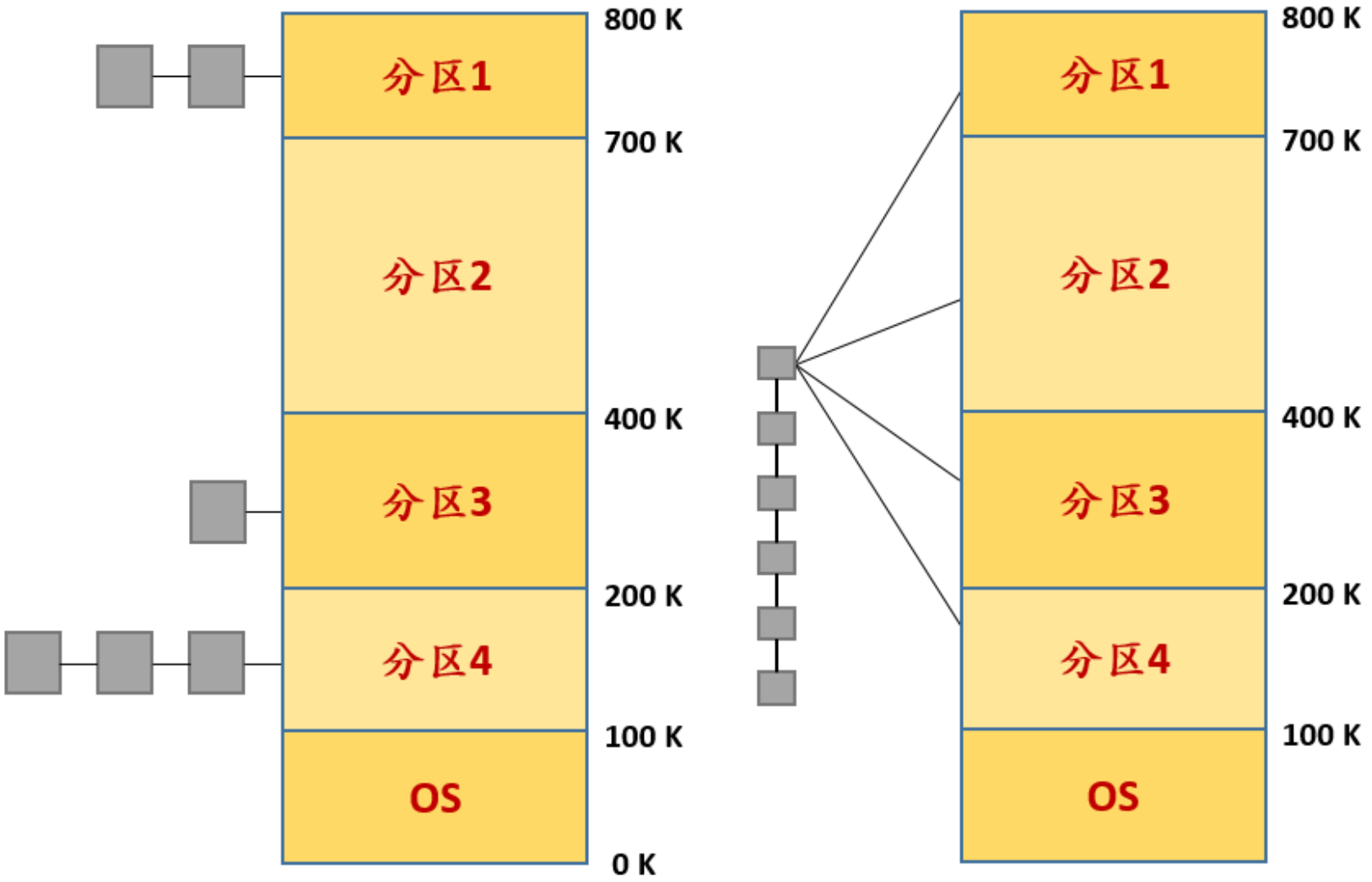
简单，内存利用率低

固定分区

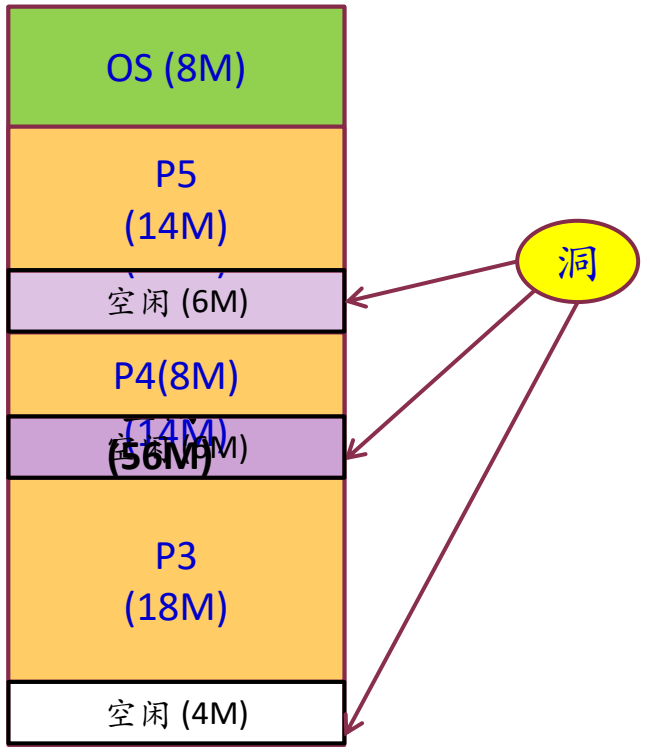
最简单的可运行多道程序的存储管理方式

- 把内存空间分割成若干区域，称为分区
- 每个分区的大小可以相同也可以不同
- 分区大小固定不变
- 每个分区装一个且只能装一个进程

固定分区示例


800 K
分区1
700 K
分区2
400 K
分区3
200 K
分区4
100 K
OS
0 K
800 K
分区1
700 K
分区2
400 K
分区3
200 K
分区4
100 K
OS

可变分区



- 根据进程的需要，把内存空闲空间分割出一个分区，分配给该进程
- 剩余部分成为新的空闲区

经过一段时间的分配回收后，内存中存在很多很小的空闲块。它们每一个都很小，不足以满足分配要求；但其总和满足分配要求。这些空闲块被称为碎片

页式存储管理方案

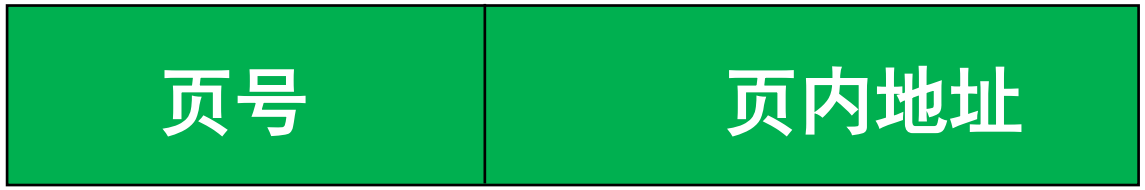
- 设计思想
  - 用户进程地址空间被划分为大小相等的部分，称为页（page）或页面，从0开始编号
  - 内存空间按同样大小划分为大小相等的区域，称为页框（page frame），从0开始编号；也称为物理页面，页帧，内存块
  - 内存分配（规则）

以页为单位进行分配，并按进程需要的页数来分配；逻辑上相邻的页，物理上不一定相邻

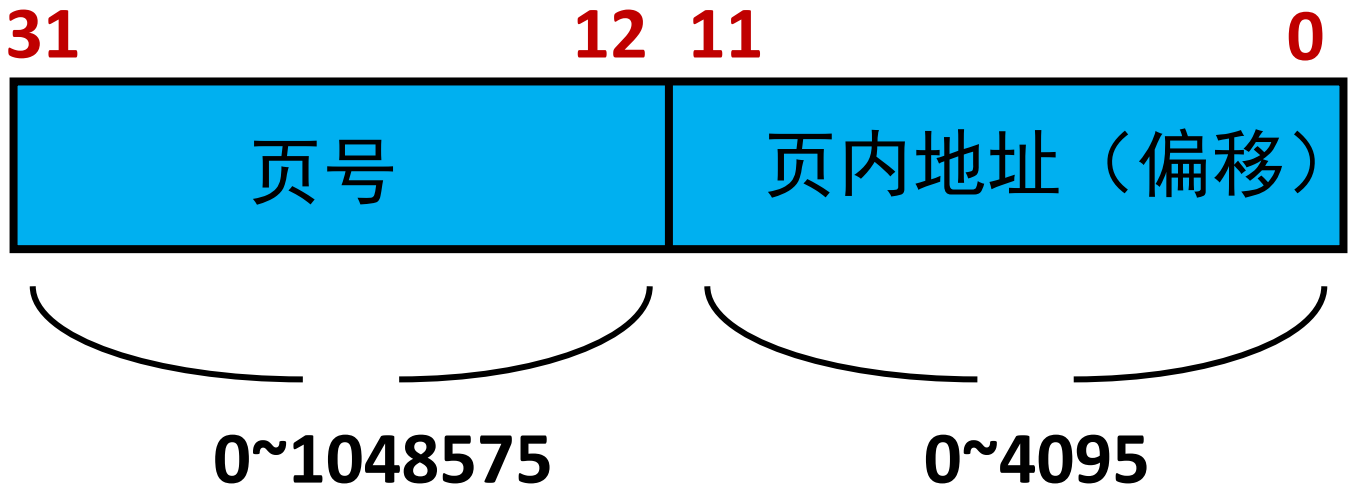
  - 典型页面尺寸：4K 或 4M

页式存储管理方案

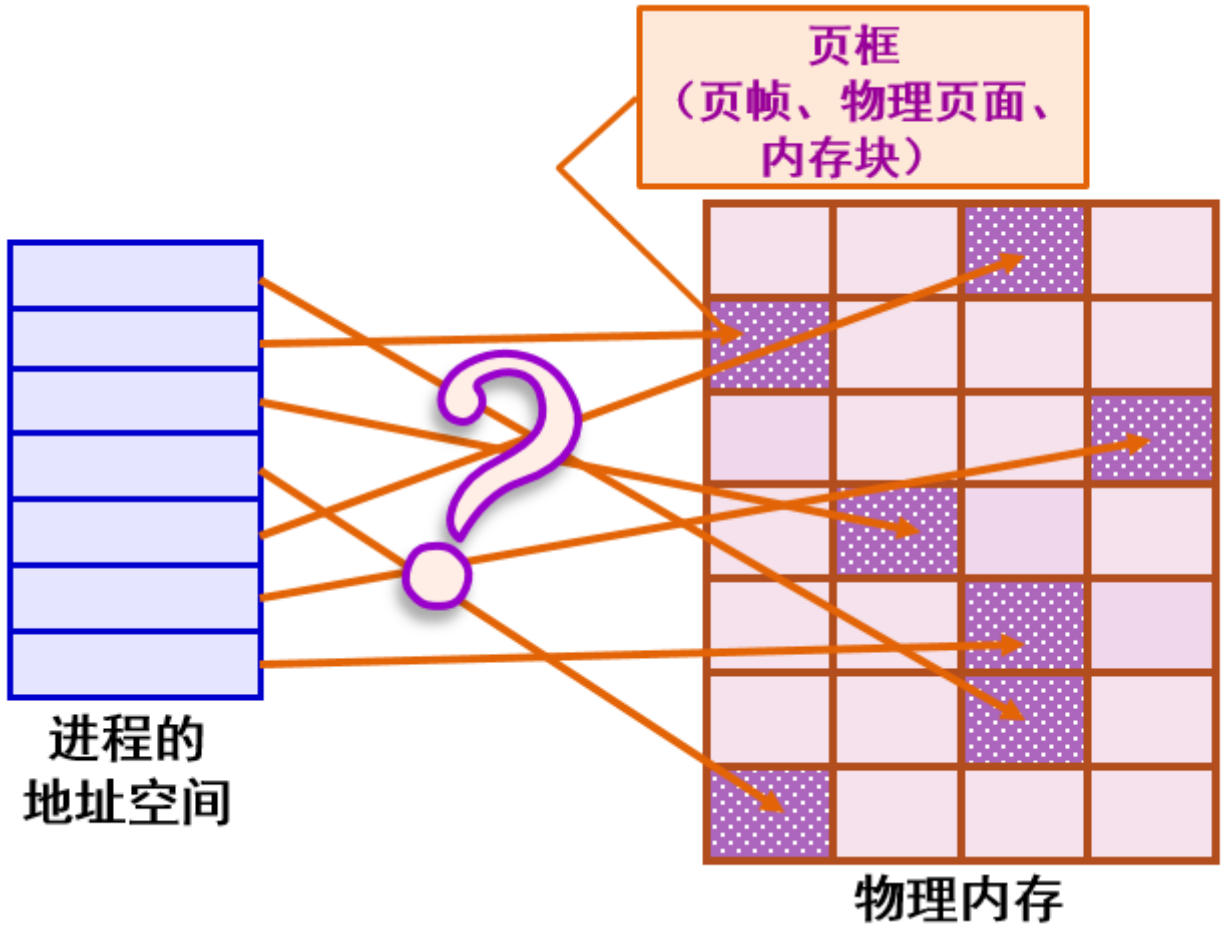
逻辑地址



32位计算机的逻辑地址

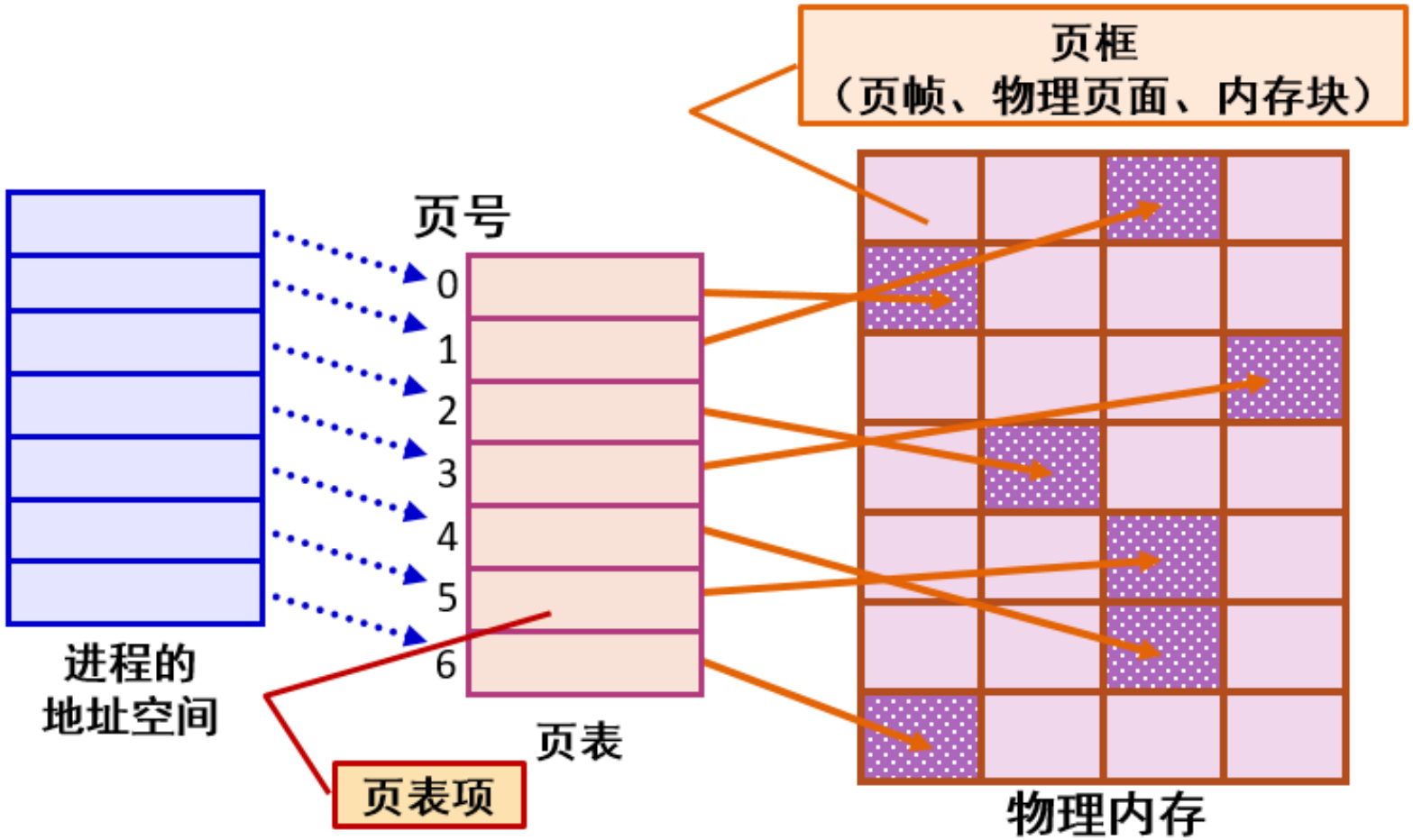


内存分配



逻辑上相邻的页物理上不一定相邻
如何找到下一页的页框?

内存分配



页表：逻辑页与物理页的映射关系

相关数据结构及地址转换

- 页表
  - 页表项：记录逻辑页号与页框号的对应关系
  - 每个进程一个页表，存放在内存
  - 页表起始地址保存在何处？

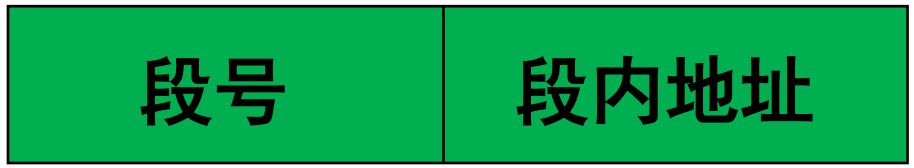
    PCB->页表寄存器
- 空闲内存管理
- 地址转换（硬件支持）

  CPU取到逻辑地址，自动划分为页号和页内地址；用页号查页表，得到页框号，再与页内偏移拼接成为物理地址

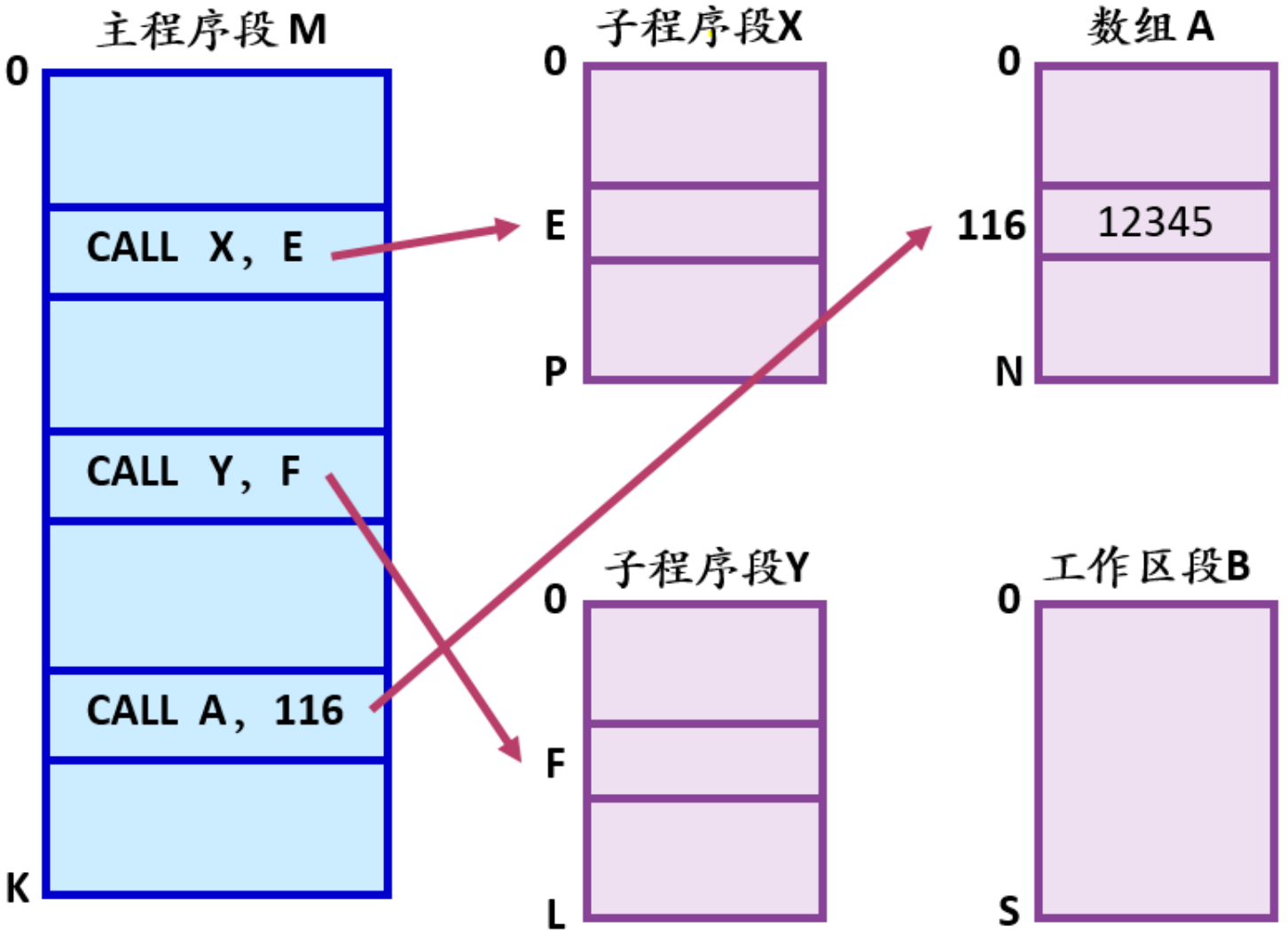
段式存储管理方案

- 设计思想
  - 用户进程地址空间：按程序自身的逻辑关系划分为若干个程序段，每个程序段都有一个段名
  - 内存空间被动态划分为若干长度不相同的区域，称为物理段，每个物理段由起始地址和长度确定
  - 内存分配（规则）：以段为单位进行分配，每段在内存中占据连续空间，但各段之间可以不相邻

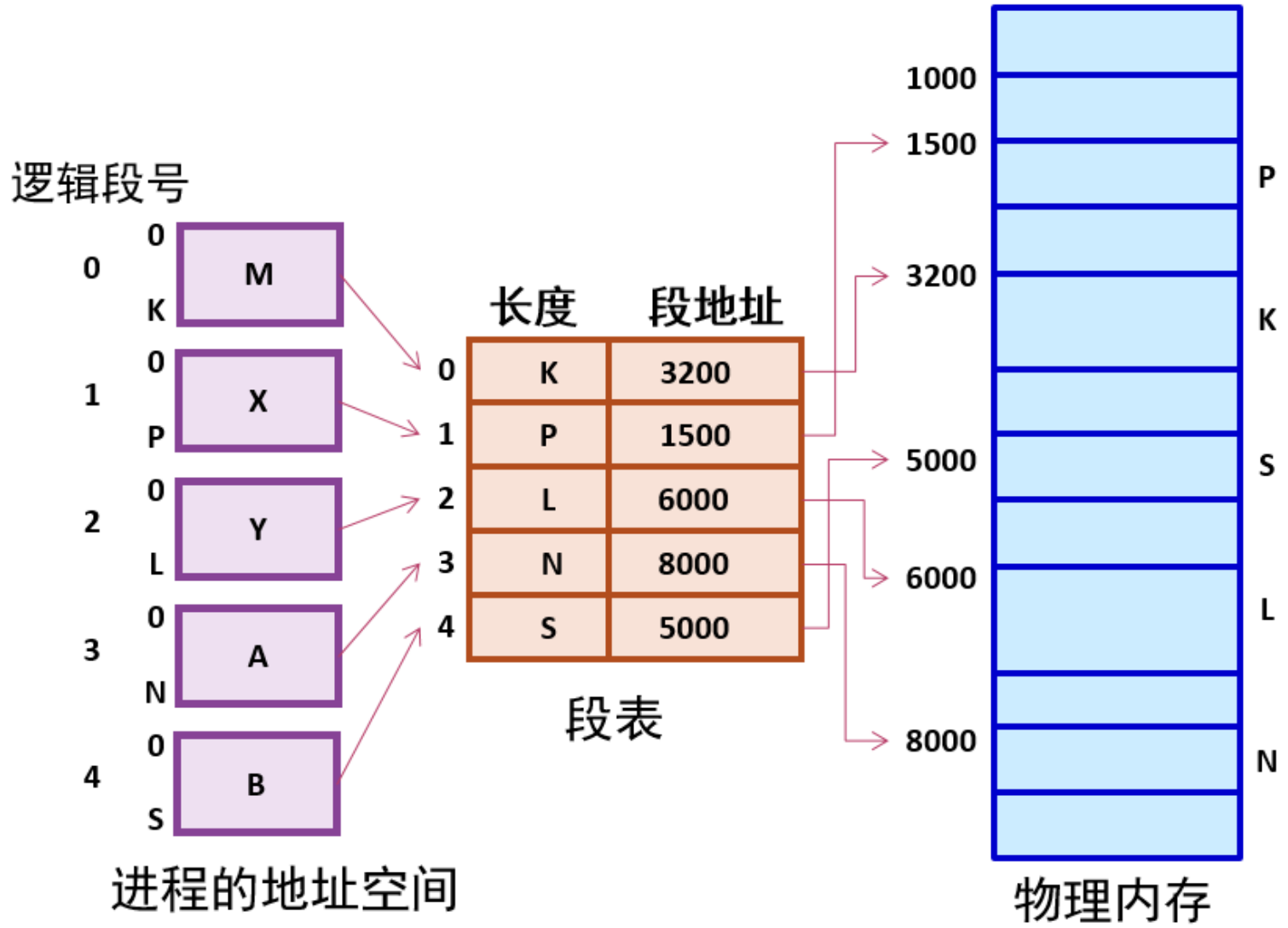
- **逻辑地址**



示意图(1/2)


主程序段 M
0
CALL X, E
CALL Y, F
CALL A, 116
K
子程序段X
0
E
P
数组 A
0
116
12345
N
子程序段Y
0
F
L
工作区段B
0
S

示意图(2/2)


逻辑段号
0
0
K
M
1
0
P
X
2
0
L
Y
3
0
N
A
4
0
S
B
长度
段地址
0
K
3200
1
P
1500
2
L
6000
3
N
8000
4
S
5000
段表
1000
1500
3200
5000
6000
8000
P
K
S
L
N
进程的地址空间
物理内存

相关数据结构及地址转换

- 段表
  - 每项记录了段号、段首地址和段长度之间的关系
  - 每个进程一个段表，存放在内存
  - 段表起始地址保存在何处？

- 地址转换（硬件）

CPU取到逻辑地址，用段号查段表，得到该段在内存的起始地址，与段内偏移地址计算出物理地址

段页式存储管理方案

- 产生背景

  综合页式、段式方案的优点，克服二者的缺点

- 设计思想

  用户进程划分：

  先按段划分，每一段再按页面划分

  逻辑地址：

  内存划分：同页式存储管理方案

  内存分配：以页为单位进行分配

段页式存储管理

- **数据结构及有关操作**
  - 段表：记录了每一段的页表始址和页表长度
  - 页表：记录了逻辑页号与页框号的对应关系

    每一段有一张页表，一个进程有多个页表

  - 空闲区管理：同页式管理
  - 内存分配、回收：同页式管理

- **地址转换**

段号	段内地址	
	页号	页内地址

本节目录

- **虚拟存储器**
  - 内存管理方案（与操作系统课程有重叠，便于理解）
  - **虚拟存储技术**

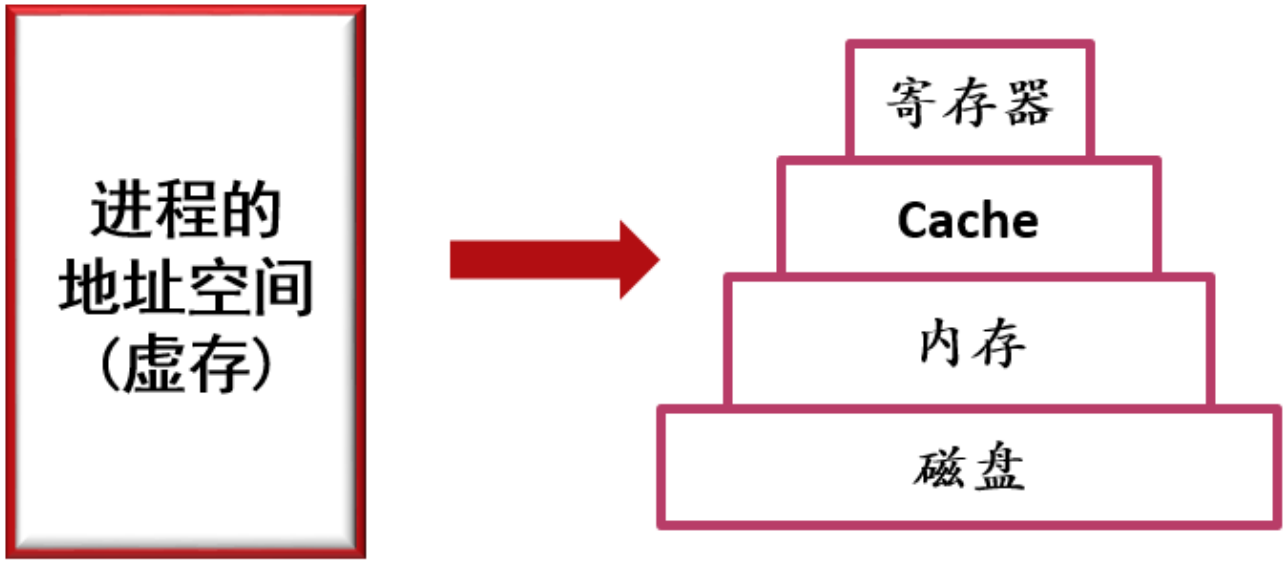
虚拟存储技术

- 虚拟存储技术：当进程运行时，先将其一部分装入内存，另一部分暂留在磁盘，当要执行的指令或访问的数据不在内存时，由操作系统自动完成将它们从磁盘调入内存的工作

- 虚拟地址空间：分配给进程的虚拟内存
- 虚拟地址：在虚拟内存中指令或数据的位置，该位置可以被访问，仿佛它是内存的一部分

思考：虚拟内存在哪里？？？

虚存 与 存储体系



- 把内存与磁盘有机地结合起来使用，从而得到一个容量很大的“内存”，即虚存
- 虚存是对内存的抽象，构建在存储体系之上，由操作系统协调各存储器的使用
- 虚存提供了一个比物理内存空间大得多的地址空间

虚拟页式（paging）

虚拟存储技术 + 页式存储管理方案

→ 虚拟页式存储管理系统

具体有两种方式
1、请求调页（demand paging）
2、预先调页（prepaging）

- 基本思想
  - 进程开始运行之前，不是装入全部页面，而是装入一个或零个页面
  - 之后，根据进程运行的需要，动态装入其他页面
  - 当内存空间已满，而又需要装入新的页面时，则根据某种算法置换内存中的某个页面，以便装入新的页面

以CPU时间和磁盘空间换取昂贵内存空间，这是操作系统中的资源转换技术

页表表项设计

- 页表由页表项组成
- 页框号、有效位、访问位、修改位、保护位
  - 页框号（内存块号、物理页面号、页帧号）
  - 有效位（驻留位、中断位）：表示该页是在内存还是在磁盘
  - 访问位：引用位
  - 修改位：此页在内存中是否被修改过
  - 保护位：读/可读写

关于页表

- **32位虚拟地址空间的页表规模？**

  页面大小为4K；页表项大小为4字节

  则： 一个进程地址空间有 ？页 2^{20}

  其页表需要占 ？页（页表页） 1024

- **64位虚拟地址空间**

  页面大小为4K；页表项大小为8字节

  页表规模： 32,000 TB

- **页表页在内存中若不连续存放，则需要引入页表页的地址索引表**

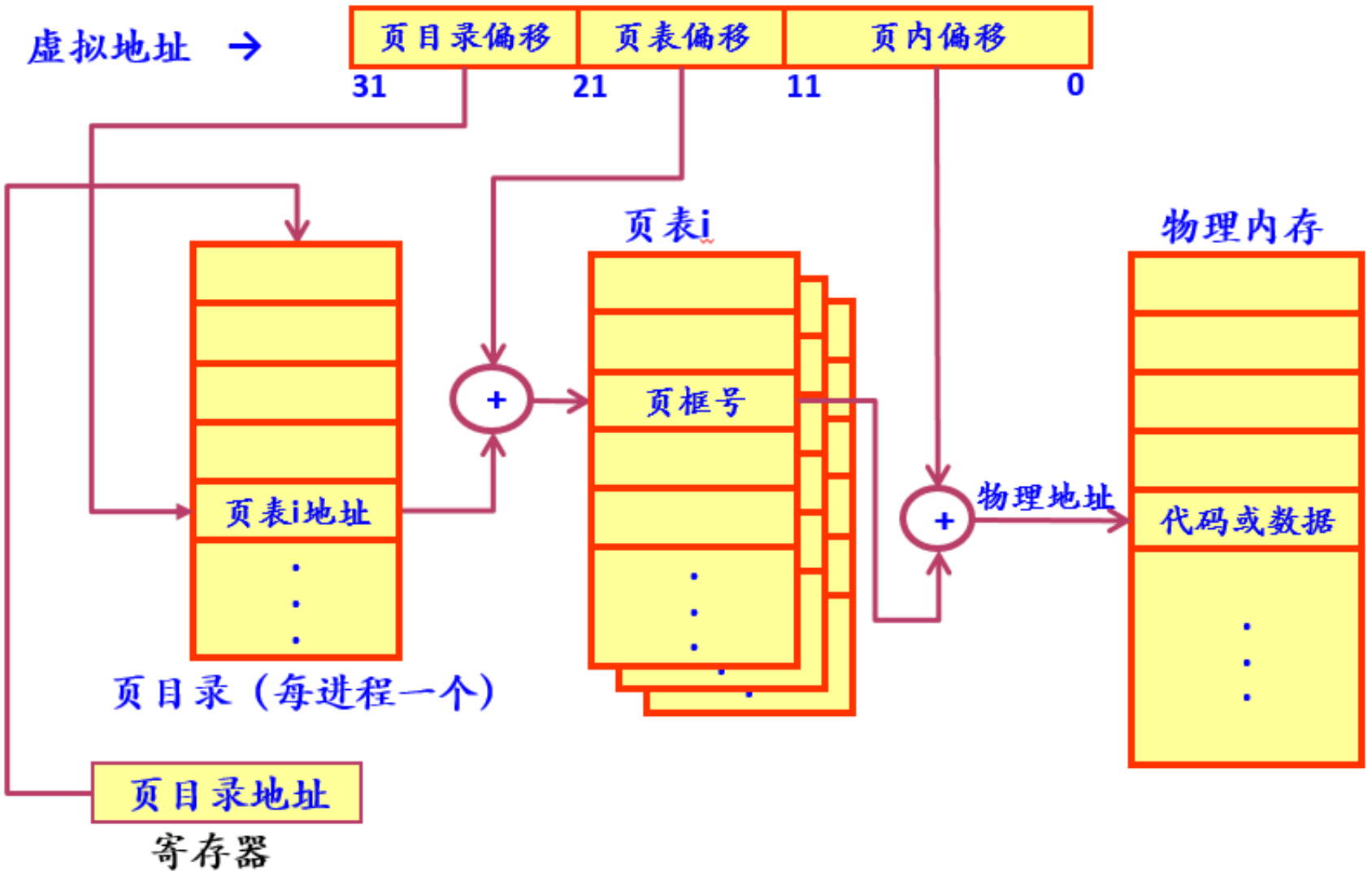
  →

  页目录（Page Directory）

为什么要引入二级/多级页表?

- 如果一个页表的大小超过了一个页面（假如是两个页面），那么分页存储管理系统只能分配两个离散的页面存储这个页表，可是PTR（页表寄存器）只有页表始址和页表长度，它默认页表是连续的
- 多级页表可以节省内存空间
  - 进程未使用的页暂时可以不用为其建立页表，如果使用一级页表，必须为所有虚拟地址分配页表

二级页表结构及地址映射



课堂练习

- 已知某系统页面长为4KB，页表项为4B，采用多层分页策略映射64位虚拟地址空间，若限定最高层页表占1页，问需要采用几层分页策略？

- 64位虚拟地址空间共有页面2^{64}/4KB=2^{52}个

  一个页面有4KB/4B = 1K个页表项

  6层分层策略

引入反转(倒排)页表

- **地址转换问题**

  从虚拟地址空间出发：虚拟地址 → 查页表 → 得到页框号 → 形成物理地址

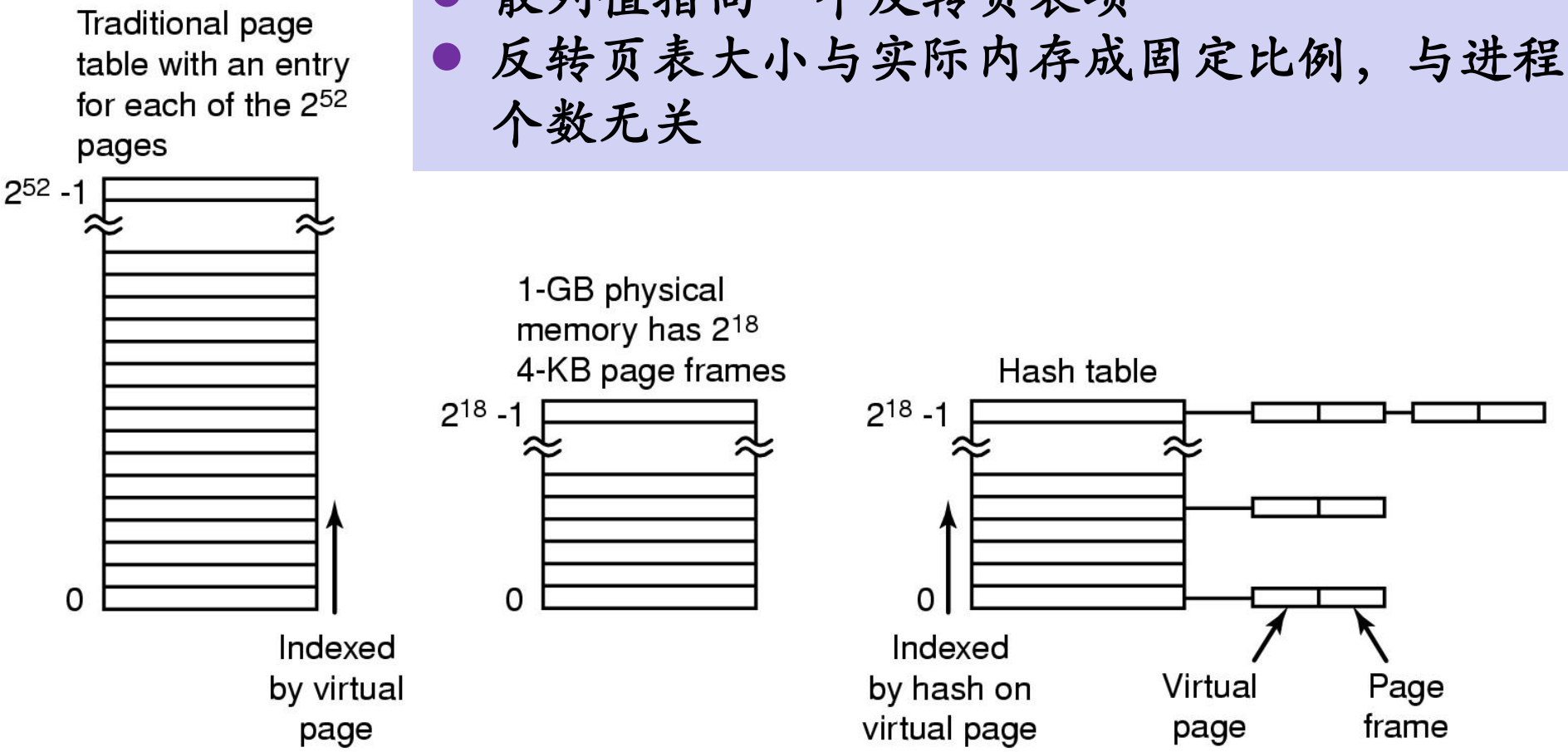
  每个进程一张页表

传统的页表的大小都是和进程的虚拟地址空间成正比的，从而页表非常大

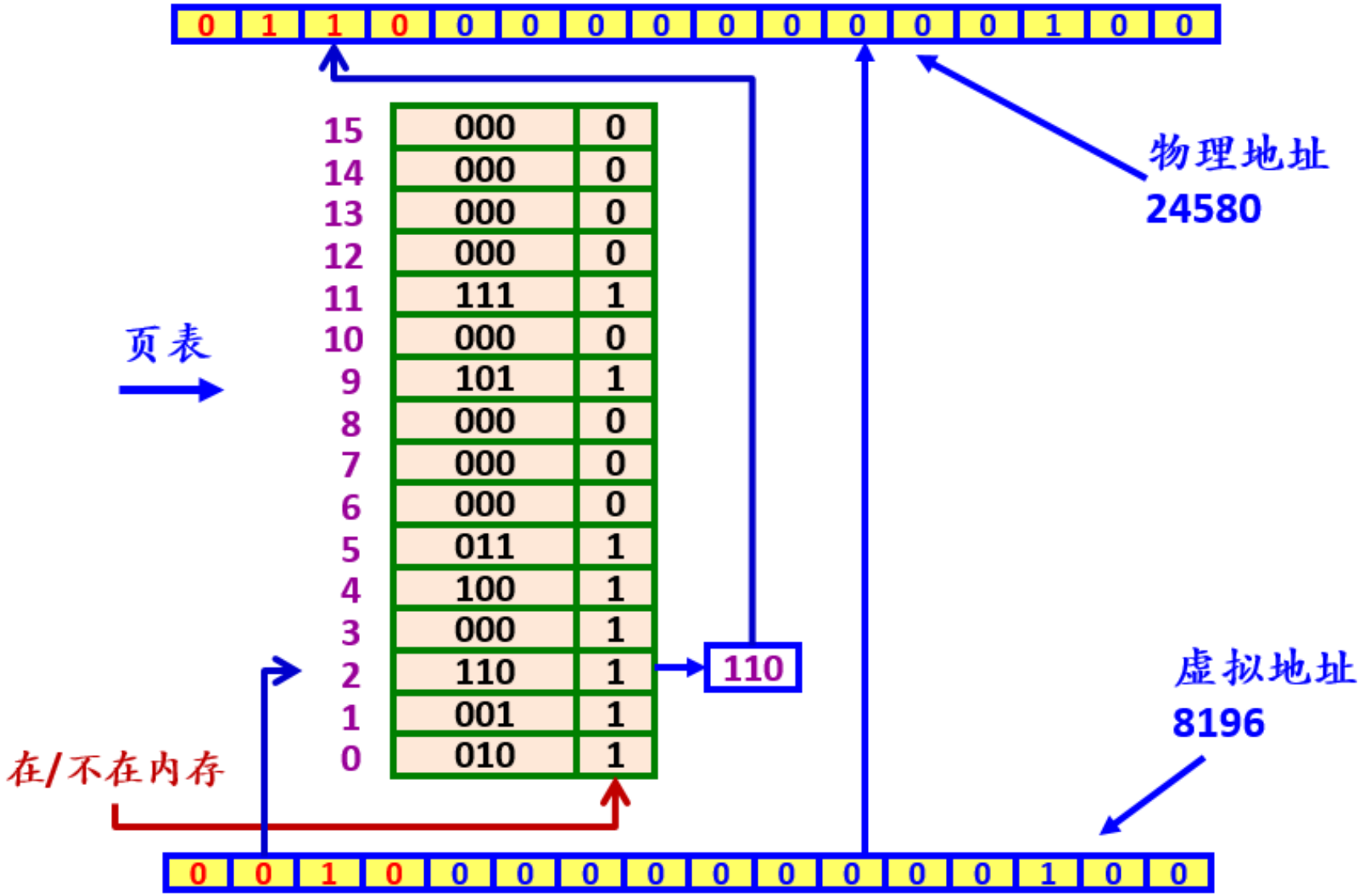
- **解决思路**
  - 从物理地址空间出发，系统建立一张页表
  - 页表项记录进程i的某虚拟地址(虚页号)与页框号的映射关系

反转(倒排)页表设计

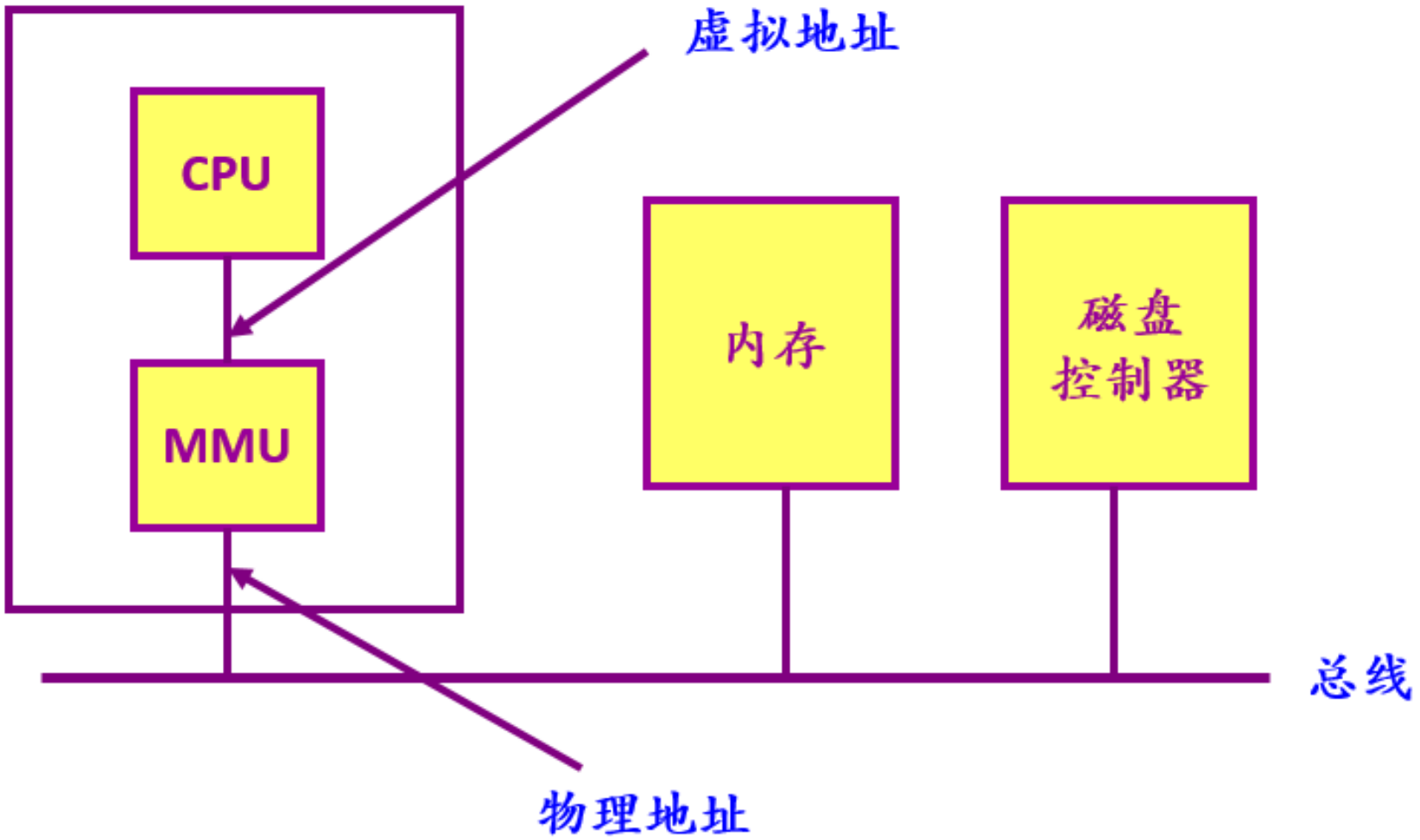
- PowerPC、UltraSPARC和IA-64 等体系结构采用
- 将虚拟地址的页号部分映射到一个散列值
- 散列值指向一个反转页表项
- 反转页表大小与实际内存成固定比例，与进程个数无关



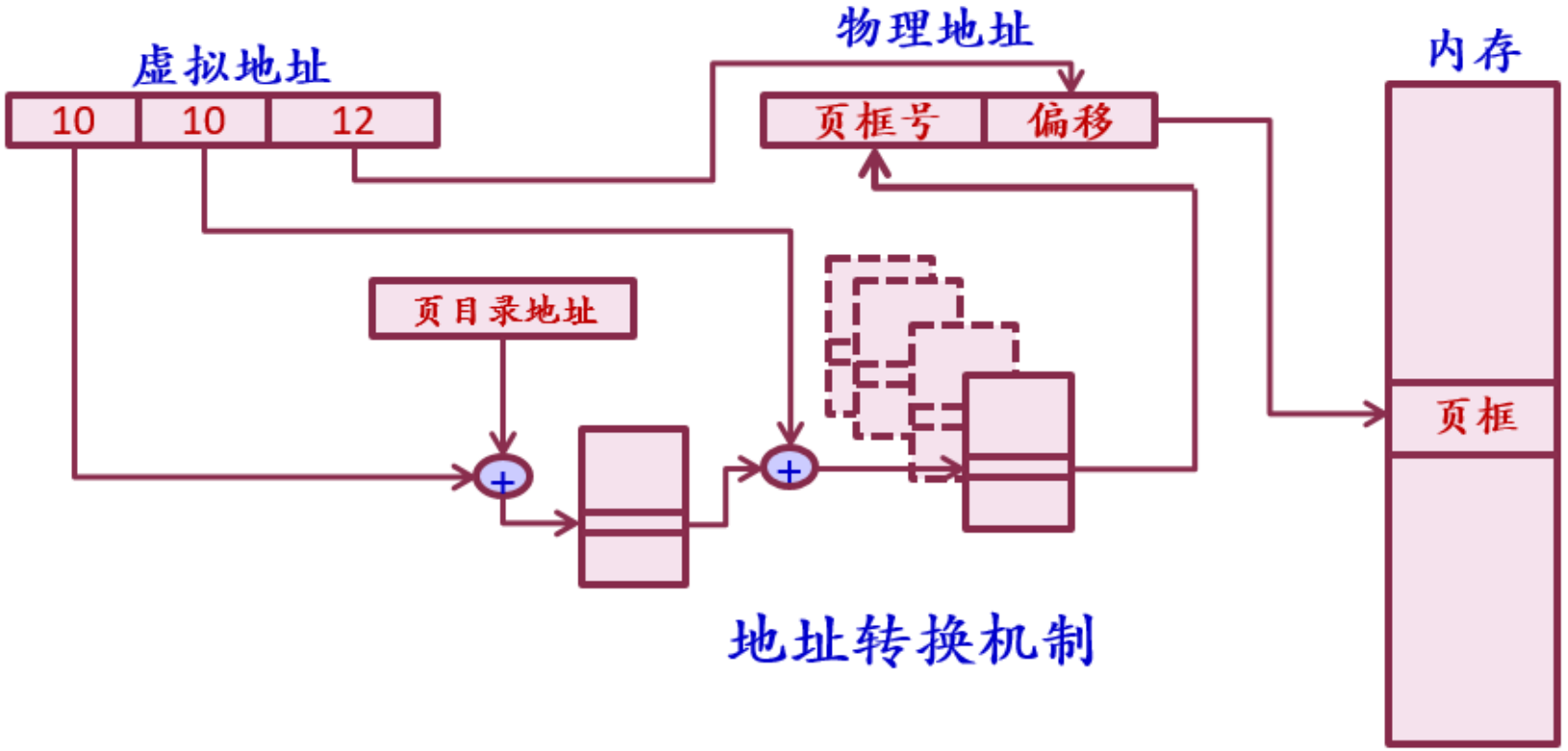
地址转换过程示意


0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0
物理地址
24580
页表
15 000 0
14 000 0
13 000 0
12 000 0
11 111 1
10 000 0
9 101 1
8 000 0
7 000 0
6 000 0
5 011 1
4 100 1
3 000 1
2 110 1
1 001 1
0 010 1
110
虚拟地址
8196
在/不在内存
0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0

MMU：内存管理单元



地址转换(映射)



快表(TLB)的引入

问题

- 虚存访问 → 两次或两次以上的内存访问
  - 一次取相应的页表项
  - 一次取需要的数据
- CPU的指令处理速度与内存指令的访问速度差异大，CPU的速度得不到充分利用
- 如何加快地址映射速度，以改善系统性能？

程序访问的局部性原理 → 引入快表(TLB)

快表是什么?

- TLB — Translation Look-aside Buffers

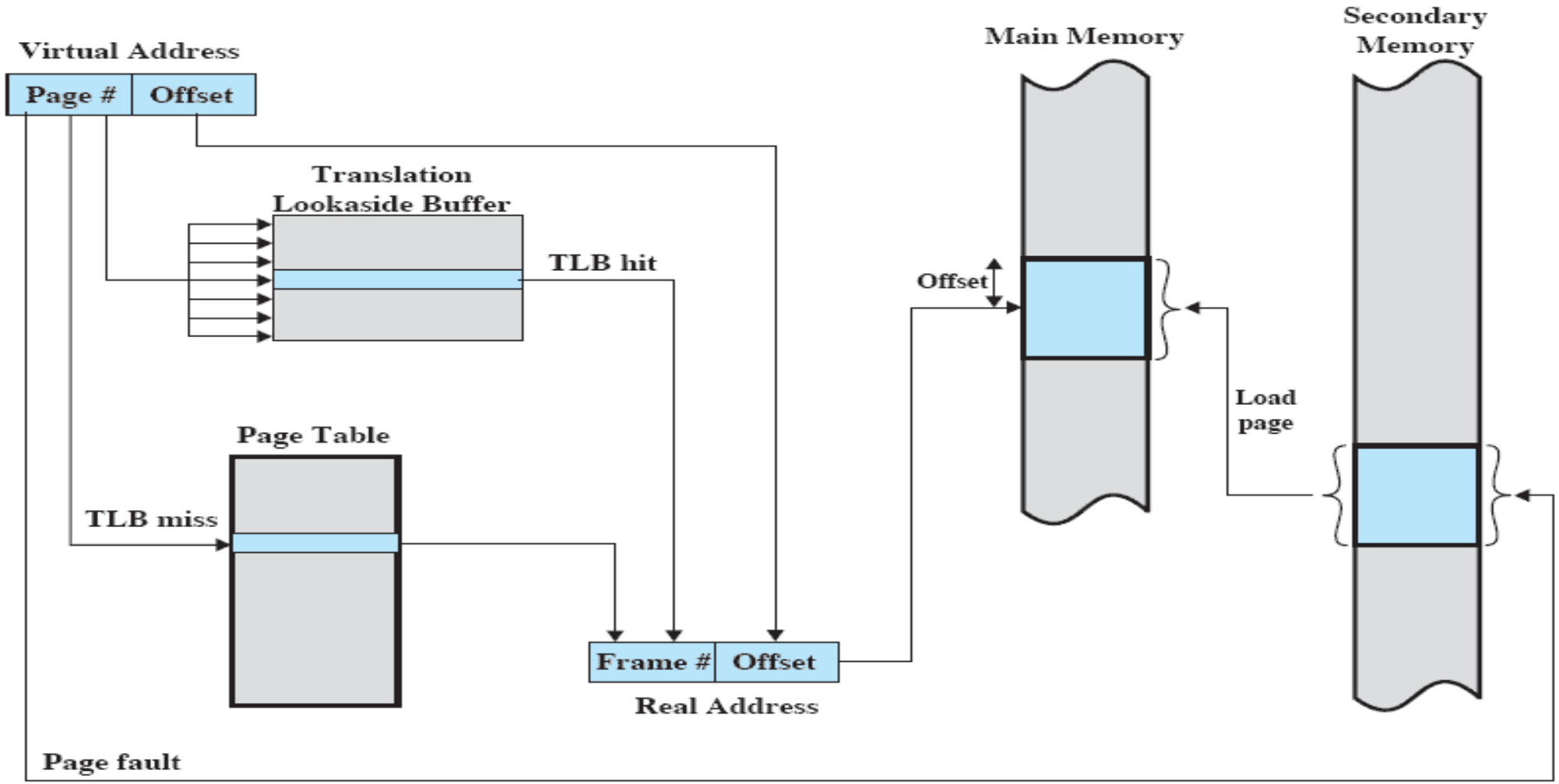
  - 在CPU中引入的高速缓存（Cache），可以匹配CPU的处理速率和内存的访问速度
  - 一种随机存取型存储器，除连线寻址机制外，还有接线逻辑，能按特定的匹配标志在一个存储周期内对所有的字同时进行比较

- 相联存储器（associative memory）

  特点：按内容并行查找，在一个存储周期内同时进行比较

- 保存正在运行进程的页表的子集(部分页表项)

加入TLB后地址转换过程示意



cite from William Stallings

页错误Page fault

- 页面错误、页故障、页面失效
- 地址转换过程中硬件产生的异常
- 具体原因
  - 所访问的虚拟页面没有调入物理内存

    → 缺页异常
  - 页面访问违反权限（读/写、用户/内核）
  - 错误的访问地址
  - … …

缺页异常处理

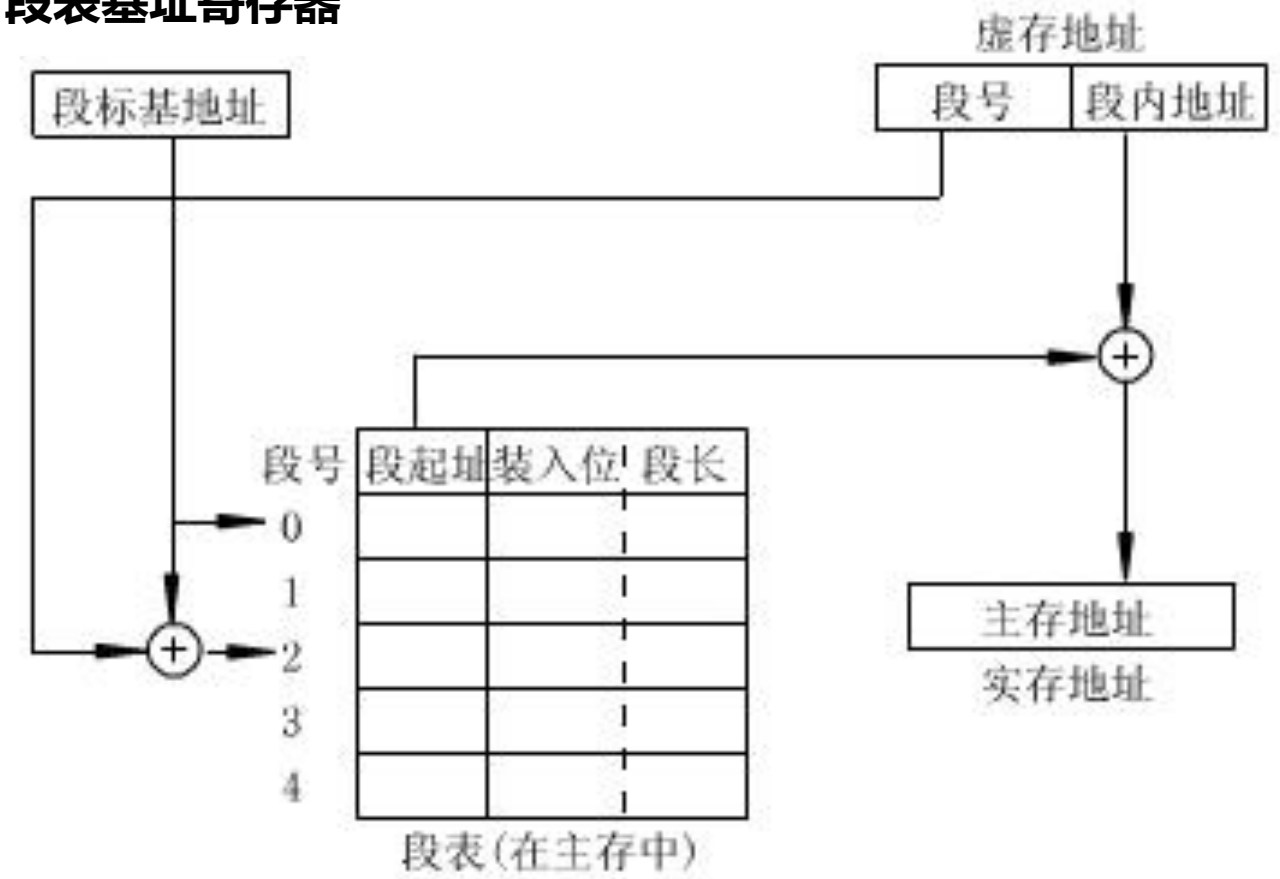
- 是一种Page Fault
- 在地址映射过程中，硬件检查页表时发现所要访问的页面不在内存，则产生该异常——缺页异常
- 操作系统执行缺页异常处理程序：获得磁盘地址，启动磁盘，将该页调入内存
  - 如果内存中有空闲页框，则分配一个页框，将新调入页装入，并修改页表中相应页表项的有效位及相应的页框号
  - 若内存中没有空闲页框，则要置换内存中某一页框；若该页框内容被修改过，则要将其写回磁盘

段式虚拟存储器

- **段式虚拟存储器**
  - 段长因程序而异
  - 地址变换方法
  - 段表的结构（段始址、装入位、段长）
  - 分析：
    - 优点：与程序的自然分段相对应
    - 缺点：存储管理麻烦，碎片

段式虚拟存储器

段表基址寄存器

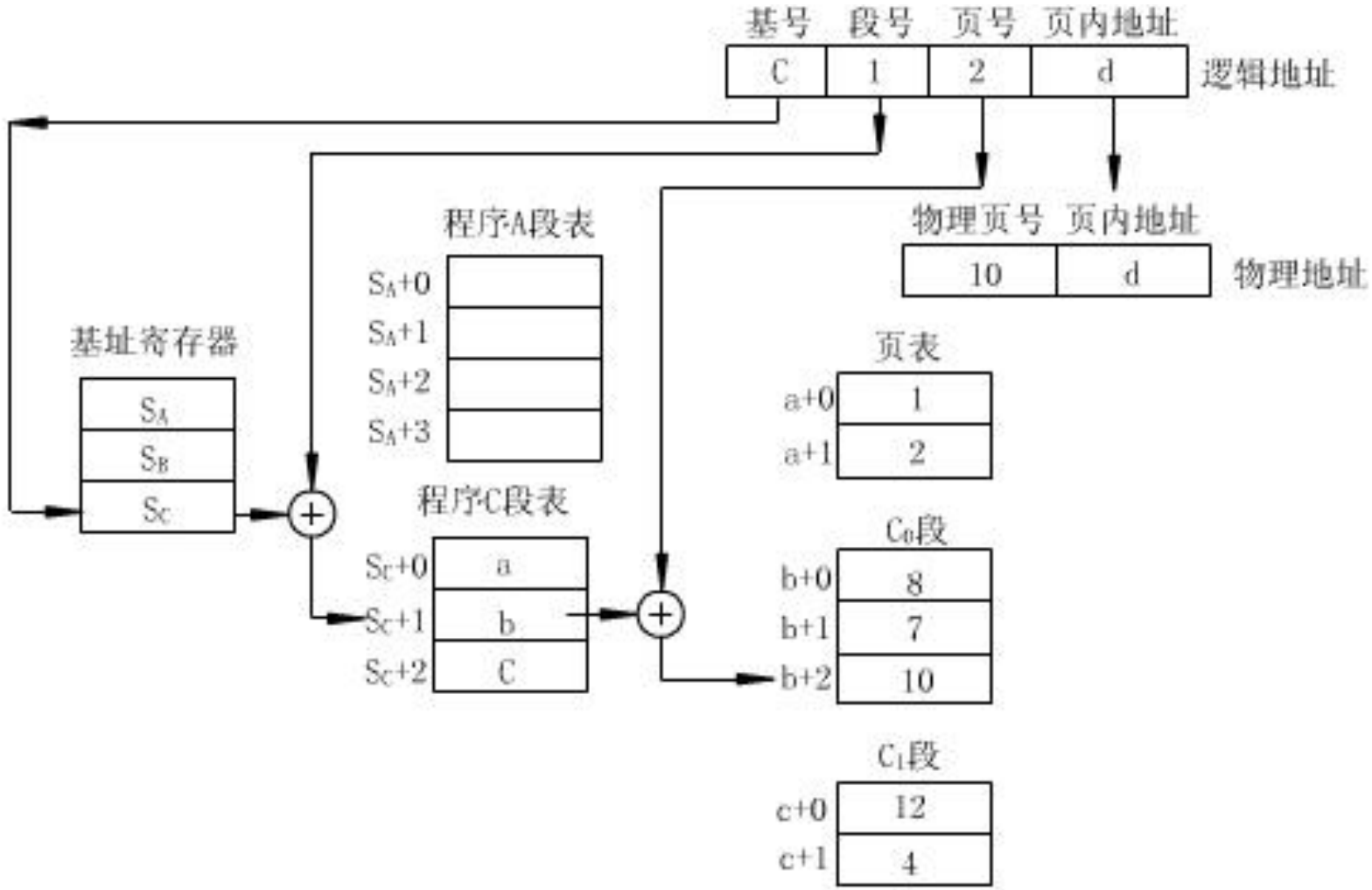


段页式虚拟存储器

- **前两者的结合**
  - 把程序按照逻辑单位分段，在把各个段分成等长的页。调入和调出是按照页面进行的，也可以实现段的共享和保护
  - 每道程序一个段表和一组页表进行定位
  - 引入了多道程序以后，每道程序要一个基（地址）号地址格式变为：

基号	段号	页号	页内地址

段页式虚拟存储器


基号
段号
页号
页内地址
C
1
2
d
逻辑地址
物理页号
页内地址
10
d
物理地址
程序A段表
S_A+0
S_A+1
S_A+2
S_A+3
基址寄存器
S_A
S_B
S_C
程序C段表
S_C+0
S_C+1
S_C+2
a
b
C
页表
a+0
a+1
1
2
C_0段
b+0
b+1
b+2
8
7
10
C_1段
c+0
c+1
I2
4

课堂练习

- 某计算机系统的逻辑地址空间由128个段构成，每个段可具有最多32个页，每页4K字，主存容量1M字，指出逻辑地址和物理地址的格式。

逻辑地址：	**段号**	**$\log_2 128 = 7$**
	段内地址	**$\log_2 32 = 5$**
	页内地址	**$\log_2 4K = 12$**
物理地址：	**主存地址**	**$\log_2 1M = 20$**
	页内地址	**12**
	实页号为	**20-12=8**

虚存与Cache的异同

- 主存—外存层次和cache—主存层次用的地址变换映射方法和替换策略是相同的，都基于程序的局部性原则
  - 把程序中最常用的那部分放在高速存储器中
  - 一旦某部分变得不常用了，就送回到低速存储器
  - 换入对用户是透明的（不可见）
  - 目的是使性能接近高速存储器，价格和容量接近低速存储器

虚存与Cache的异同

- Cache 主要解决主存与CPU的速度差异问题；
  虚存主要是解决存储容量问题
- CPU与Cache和主存之间均有直接访问通路，Cache不命中时可直接访问主存；辅存与CPU之间不存在直接的数据通路，主存不命中时只能通过调页解决，CPU最终还要访问主存。
- Cache的管理由硬件完成，虚存管理由软件（操作系统）和硬件共同完成
- 在主存—外存层次中，如果没有命中，则性能损失要大于cache-主存层次未命中的损失