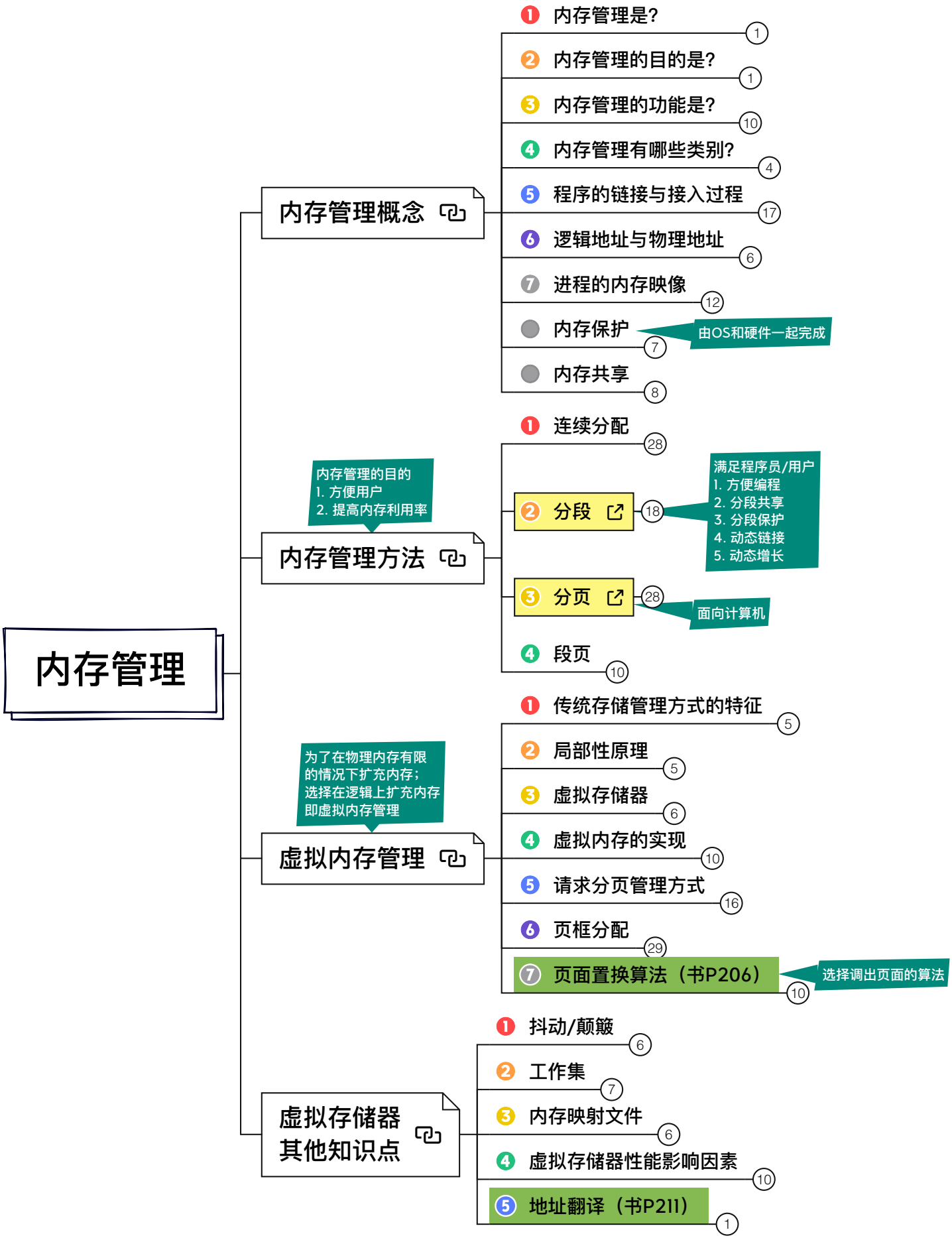
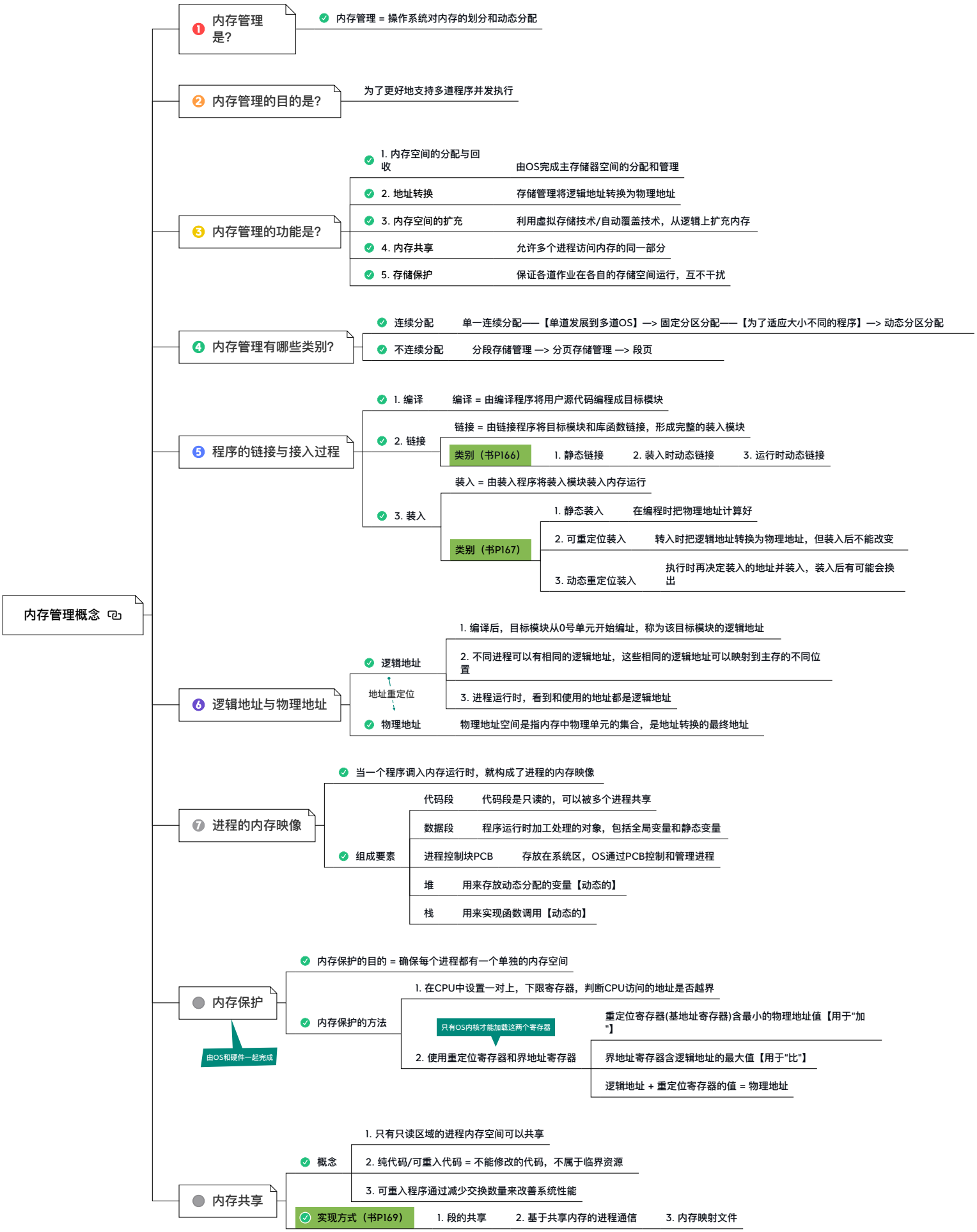
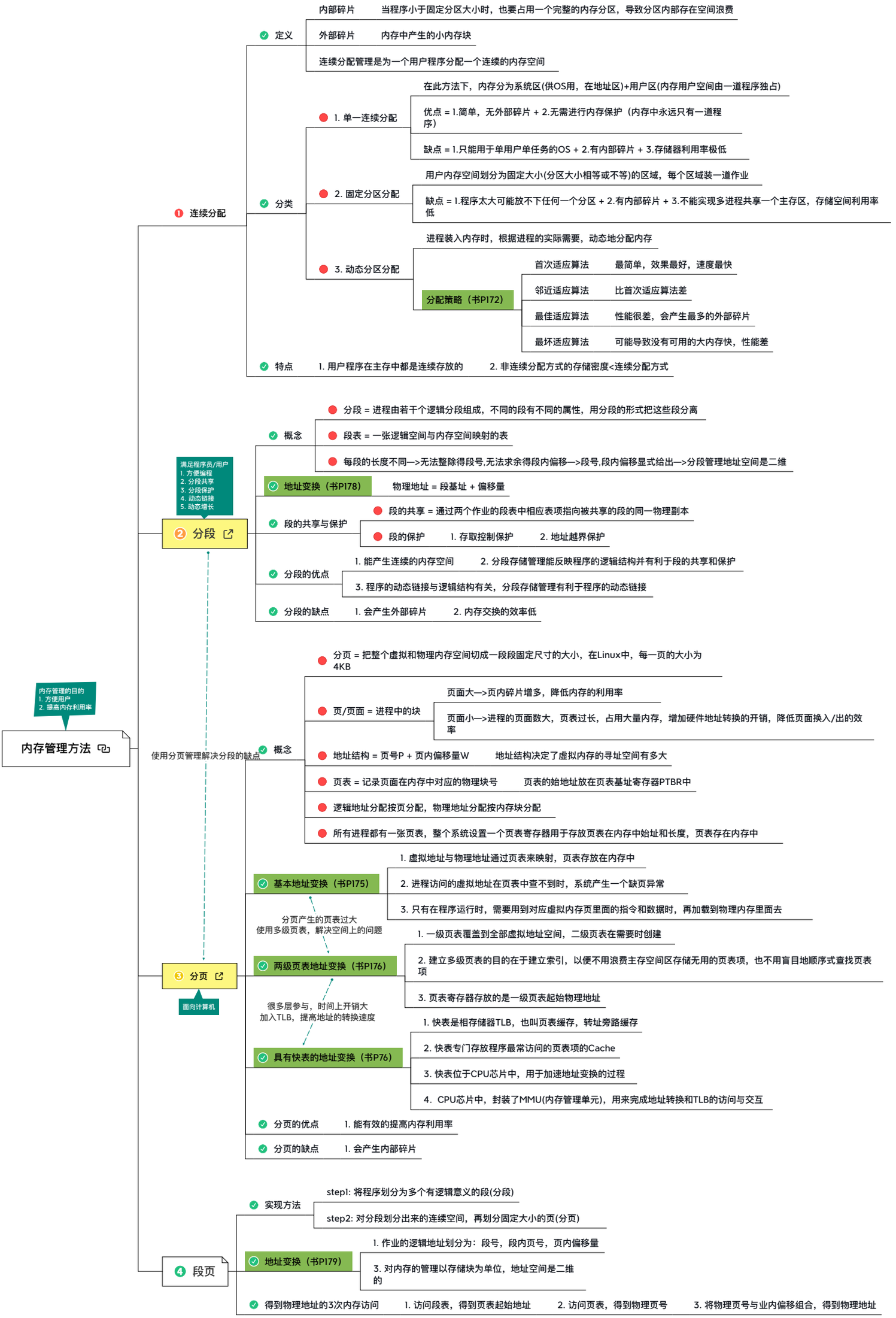
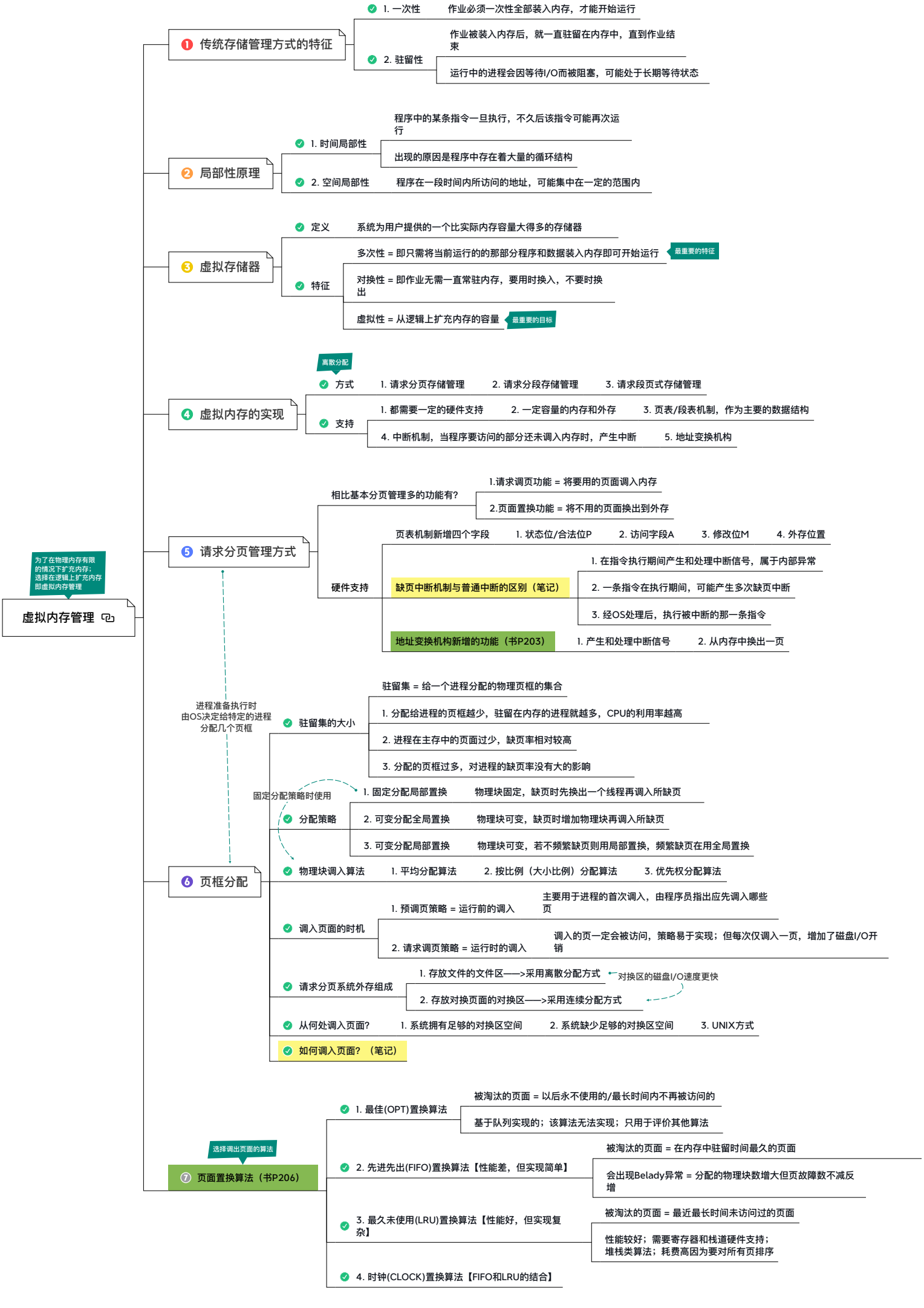
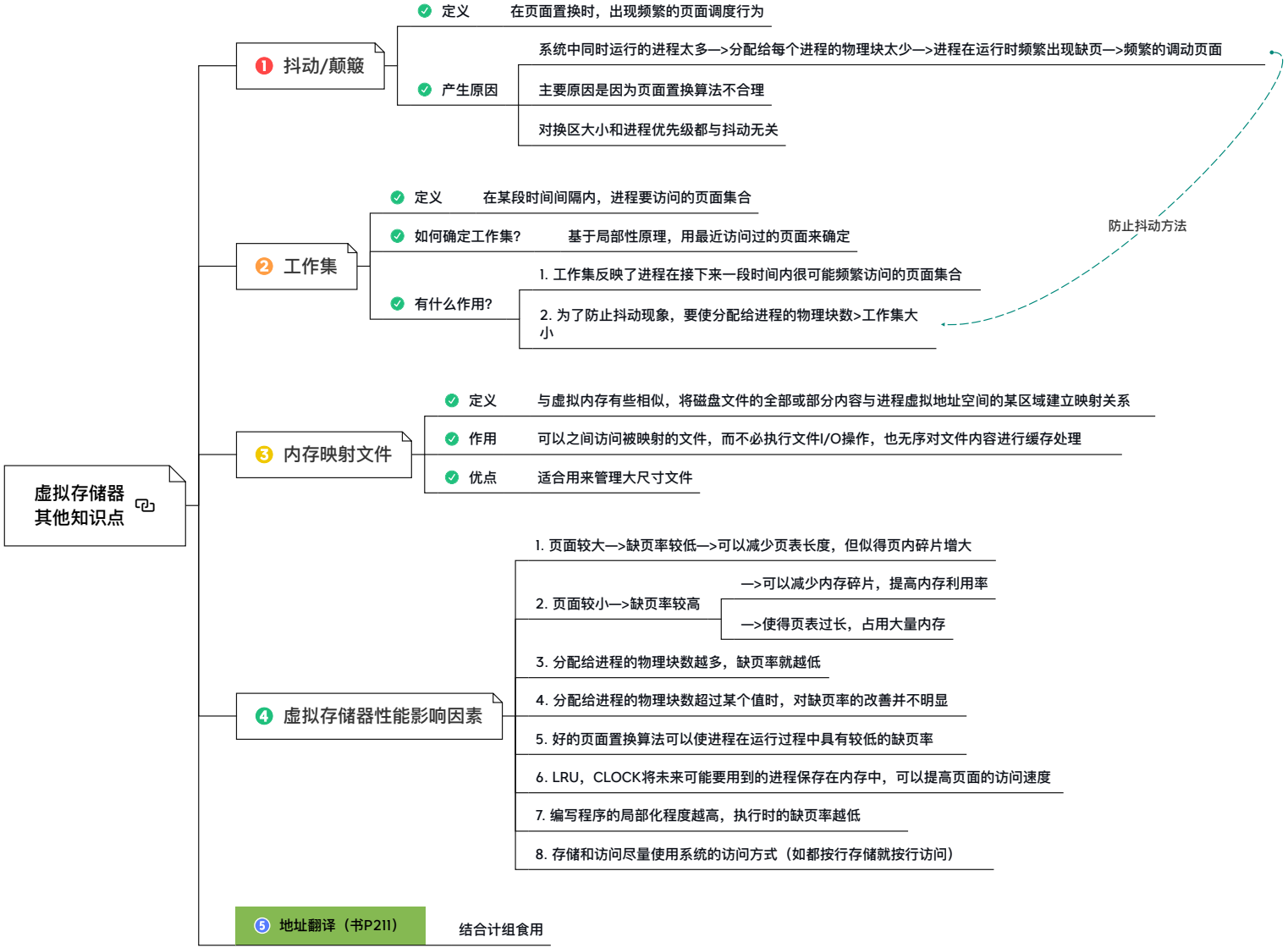

内存管理
内存管理概念
1 内存管理是? 1
2 内存管理的目的是? 1
3 内存管理的功能是? 10
4 内存管理有哪些类别? 4
5 程序的链接与接入过程 17
6 逻辑地址与物理地址 6
7 进程的内存映像 12
内存保护 7
由OS和硬件一起完成
内存共享 8
内存管理方法
内存管理的目的
1. 方便用户
2. 提高内存利用率
1 连续分配 28
2 分段 18
满足程序员/用户
1. 方便编程
2. 分段共享
3. 分段保护
4. 动态链接
5. 动态增长
3 分页 28
面向计算机
4 段页 10
虚拟内存管理
为了在物理内存有限的情况下扩充内存；选择在逻辑上扩充内存即虚拟内存管理
1 传统存储管理方式的特征 5
2 局部性原理 5
3 虚拟存储器 6
4 虚拟内存的实现 10
5 请求分页管理方式 16
6 页框分配 29
7 页面置换算法（书P206） 10
选择调出页面的算法
虚拟存储器其他知识点
1 抖动/颠簸 6
2 工作集 7
3 内存映射文件 6
4 虚拟存储器性能影响因素 10
5 地址翻译（书P211） 1

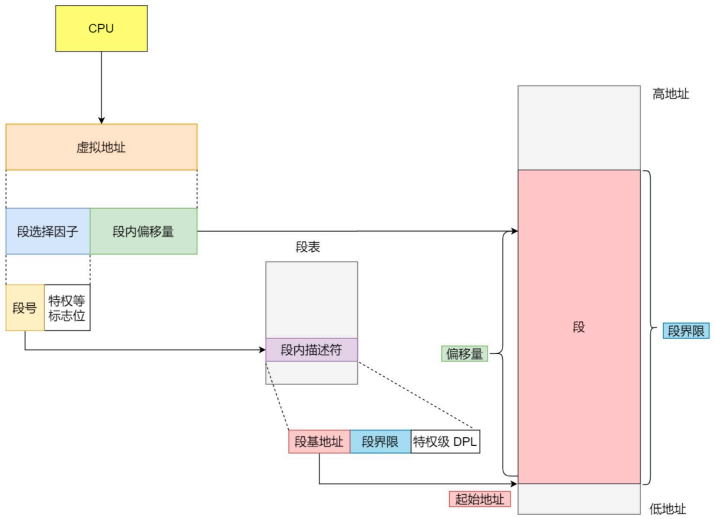

内存管理概念
1 内存管理是?
内存管理 = 操作系统对内存的划分和动态分配
2 内存管理的目的是?
为了更好地支持多道程序并发执行
3 内存管理的功能是?
1. 内存空间的分配与回收
由OS完成主存储器空间的分配和管理
2. 地址转换
存储管理将逻辑地址转换为物理地址
3. 内存空间的扩充
利用虚拟存储技术/自动覆盖技术，从逻辑上扩充内存
4. 内存共享
允许多个进程访问内存的同一部分
5. 存储保护
保证各道作业在各自的存储空间运行，互不干扰
4 内存管理有哪些类别?
连续分配
单一连续分配——【单道发展到多道OS】—> 固定分区分配——【为了适应大小不同的程序】—> 动态分区分配
不连续分配
分段存储管理 —> 分页存储管理 —> 段页
5 程序的链接与接入过程
1. 编译
编译 = 由编译程序将用户源代码编程成目标模块
2. 链接
链接 = 由链接程序将目标模块和库函数链接，形成完整的装入模块
类别（书P166）
1. 静态链接
2. 装入时动态链接
3. 运行时动态链接
3. 装入
装入 = 由装入程序将装入模块装入内存运行
类别（书P167）
1. 静态装入
在编程时把物理地址计算好
2. 可重定位装入
转入时把逻辑地址转换为物理地址，但装入后不能改变
3. 动态重定位装入
执行时再决定装入的地址并装入，装入后有可能会换出
6 逻辑地址与物理地址
逻辑地址
1. 编译后，目标模块从0号单元开始编址，称为该目标模块的逻辑地址
2. 不同进程可以有相同的逻辑地址，这些相同的逻辑地址可以映射到主存的不同位置
3. 进程运行时，看到和使用的地址都是逻辑地址
地址重定位
物理地址
物理地址空间是指内存中物理单元的集合，是地址转换的最终地址
7 进程的内存映像
当一个程序调入内存运行时，就构成了进程的内存映像
组成要素
代码段
代码段是只读的，可以被多个进程共享
数据段
程序运行时加工处理的对象，包括全局变量和静态变量
进程控制块PCB
存放在系统区，OS通过PCB控制和管理进程
堆
用来存放动态分配的变量【动态的】
栈
用来实现函数调用【动态的】
内存保护
由OS和硬件一起完成
内存保护的目的 = 确保每个进程都有一个单独的内存空间
内存保护的方法
1. 在CPU中设置一对上，下限寄存器，判断CPU访问的地址是否越界
只有OS内核才能加载这两个寄存器
2. 使用重定位寄存器和界地址寄存器
重定位寄存器(基地址寄存器)含最小的物理地址值【用于"加"】
界地址寄存器含逻辑地址的最大值【用于"比"】
逻辑地址 + 重定位寄存器的值 = 物理地址
内存共享
概念
1. 只有只读区域的进程内存空间可以共享
2. 纯代码/可重入代码 = 不能修改的代码，不属于临界资源
3. 可重入程序通过减少交换数量来改善系统性能
实现方式（书P169）
1. 段的共享
2. 基于共享内存的进程通信
3. 内存映射文件


内存管理的目的
1. 方便用户
2. 提高内存利用率
内存管理方法
使用分页管理解决分段的缺点
❶ 连续分配
定义
内部碎片　当程序小于固定分区大小时，也要占用一个完整的内存分区，导致分区内部存在空间浪费
外部碎片　内存中产生的小内存块
连续分配管理是为一个用户程序分配一个连续的内存空间
分类
1. 单一连续分配
在此方法下，内存分为系统区(供OS用，在地址区)+用户区(内存用户空间由一道程序独占)
优点 = 1.简单，无外部碎片 + 2.无需进行内存保护（内存中永远只有一道程序)
缺点 = 1.只能用于单用户单任务的OS + 2.有内部碎片 + 3.存储器利用率极低
2. 固定分区分配
用户内存空间划分为固定大小(分区大小相等或不等)的区域，每个区域装一道作业
缺点 = 1.程序太大可能放不下任何一个分区 + 2.有内部碎片 + 3.不能实现多进程共享一个主存区，存储空间利用率低
3. 动态分区分配
进程装入内存时，根据进程的实际需要，动态地分配内存
分配策略（书P172）
首次适应算法　最简单，效果最好，速度最快
邻近适应算法　比首次适应算法差
最佳适应算法　性能很差，会产生最多的外部碎片
最坏适应算法　可能导致没有可用的大内存快，性能差
特点　1. 用户程序在主存中都是连续存放的　2. 非连续分配方式的存储密度<连续分配方式
满足程序员/用户
1. 方便编程
2. 分段共享
3. 分段保护
4. 动态链接
5. 动态增长
❷ 分段
概念
分段 = 进程由若干个逻辑分段组成，不同的段有不同的属性，用分段的形式把这些段分离
段表 = 一张逻辑空间与内存空间映射的表
每段的长度不同—>无法整除得段号,无法求余得段内偏移—>段号,段内偏移显式给出—>分段管理地址空间是二维
地址变换（书P178）　物理地址 = 段基址 + 偏移量
段的共享与保护
段的共享 = 通过两个作业的段表中相应表项指向被共享的段的同一物理副本
段的保护　1. 存取控制保护　2. 地址越界保护
分段的优点
1. 能产生连续的内存空间　2. 分段存储管理能反映程序的逻辑结构并有利于段的共享和保护
3. 程序的动态链接与逻辑结构有关，分段存储管理有利于程序的动态链接
分段的缺点　1. 会产生外部碎片　2. 内存交换的效率低
❸ 分页
面向计算机
概念
分页 = 把整个虚拟和物理内存空间切成一段段固定尺寸的大小，在Linux中，每一页的大小为4KB
页/页面 = 进程中的块
页面大—>页内碎片增多，降低内存的利用率
页面小—>进程的页面数大，页表过长，占用大量内存，增加硬件地址转换的开销，降低页面换入/出的效率
地址结构 = 页号P + 页内偏移量W　地址结构决定了虚拟内存的寻址空间有多大
页表 = 记录页面在内存中对应的物理块号　页表的始地址放在页表基址寄存器PTBR中
逻辑地址分配按页分配，物理地址分配按内存块分配
所有进程都有一张页表，整个系统设置一个页表寄存器用于存放页表在内存中始址和长度，页表存在内存中
基本地址变换（书P175）
1. 虚拟地址与物理地址通过页表来映射，页表存放在内存中
2. 进程访问的虚拟地址在页表中查不到时，系统产生一个缺页异常
3. 只有在程序运行时，需要用到对应虚拟内存页里面的指令和数据时，再加载到物理内存里面去
分页产生的页表过大
使用多级页表，解决空间上的问题
两级页表地址变换（书P176）
1. 一级页表覆盖到全部虚拟地址空间，二级页表在需要时创建
2. 建立多级页表的目的在于建立索引，以便不用浪费主存空间区存储无用的页表项，也不用盲目地顺序式查找页表项
3. 页表寄存器存放的是一级页表起始物理地址
很多层参与，时间上开销大
加入TLB，提高地址的转换速度
具有快表的地址变换（书P76）
1. 快表是相存储器TLB，也叫页表缓存，转址旁路缓存
2. 快表专门存放程序最常访问的页表项的Cache
3. 快表位于CPU芯片中，用于加速地址变换的过程
4. CPU芯片中，封装了MMU(内存管理单元)，用来完成地址转换和TLB的访问与交互
分页的优点　1. 能有效的提高内存利用率
分页的缺点　1. 会产生内部碎片
❹ 段页
实现方法
step1: 将程序划分为多个有逻辑意义的段(分段)
step2: 对分段划分出来的连续空间，再划分固定大小的页(分页)
地址变换（书P179）
1. 作业的逻辑地址划分为：段号，段内页号，页内偏移量
3. 对内存的管理以存储块为单位，地址空间是二维的
得到物理地址的3次内存访问　1. 访问段表，得到页表起始地址　2. 访问页表，得到物理页号　3. 将物理页号与业内偏移组合，得到物理地址

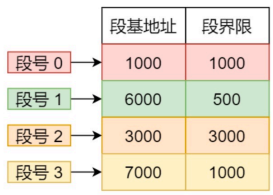

虚拟内存管理
为了在物理内存有限的情况下扩充内存；选择在逻辑上扩充内存即虚拟内存管理
1 传统存储管理方式的特征
1. 一次性 作业必须一次性全部装入内存，才能开始运行
2. 驻留性
作业被装入内存后，就一直驻留在内存中，直到作业结束
运行中的进程会因等待I/O而被阻塞，可能处于长期等待状态
2 局部性原理
1. 时间局部性
程序中的某条指令一旦执行，不久后该指令可能再次运行
出现的原因是程序中存在着大量的循环结构
2. 空间局部性 程序在一段时间内所访问的地址，可能集中在一定的范围内
3 虚拟存储器
定义 系统为用户提供的一个比实际内存容量大得多的存储器
特征
多次性 = 即只需将当前运行的的那部分程序和数据装入内存即可开始运行
最重要的特征
对换性 = 即作业无需一直常驻内存，要用时换入，不要时换出
虚拟性 = 从逻辑上扩充内存的容量
最重要的目标
4 虚拟内存的实现
离散分配
方式 1. 请求分页存储管理 2. 请求分段存储管理 3. 请求段页式存储管理
支持
1. 都需要一定的硬件支持 2. 一定容量的内存和外存 3. 页表/段表机制，作为主要的数据结构
4. 中断机制，当程序要访问的部分还未调入内存时，产生中断 5. 地址变换机构
5 请求分页管理方式
相比基本分页管理多的功能有?
1.请求调页功能 = 将要用的页面调入内存
2.页面置换功能 = 将不用的页面换出到外存
硬件支持
页表机制新增四个字段 1. 状态位/合法位P 2. 访问字段A 3. 修改位M 4. 外存位置
缺页中断机制与普通中断的区别（笔记）
1. 在指令执行期间产生和处理中断信号，属于内部异常
2. 一条指令在执行期间，可能产生多次缺页中断
3. 经OS处理后，执行被中断的那一条指令
地址变换机构新增的功能（书P203） 1. 产生和处理中断信号 2. 从内存中换出一页
6 页框分配
进程准备执行时由OS决定给特定的进程分配几个页框
驻留集的大小
驻留集 = 给一个进程分配的物理页框的集合
1. 分配给进程的页框越少，驻留在内存的进程就越多，CPU的利用率越高
2. 进程在主存中的页面过少，缺页率相对较高
3. 分配的页框过多，对进程的缺页率没有大的影响
分配策略
固定分配策略时使用
1. 固定分配局部置换 物理块固定，缺页时先换出一个线程再调入所缺页
2. 可变分配全局置换 物理块可变，缺页时增加物理块再调入所缺页
3. 可变分配局部置换 物理块可变，若不频繁缺页则用局部置换，频繁缺页在用全局置换
物理块调入算法 1. 平均分配算法 2. 按比例（大小比例）分配算法 3. 优先权分配算法
调入页面的时机
1. 预调页策略 = 运行前的调入
主要用于进程的首次调入，由程序员指出应先调入哪些页
2. 请求调页策略 = 运行时的调入
调入的页一定会被访问，策略易于实现；但每次仅调入一页，增加了磁盘I/O开销
请求分页系统外存组成
1. 存放文件的文件区——>采用离散分配方式
对换区的磁盘I/O速度更快
2. 存放对换页面的对换区——>采用连续分配方式
从何处调入页面? 1. 系统拥有足够的对换区空间 2. 系统缺少足够的对换区空间 3. UNIX方式
如何调入页面？（笔记）
7 页面置换算法（书P206）
选择调出页面的算法
1. 最佳(OPT)置换算法
被淘汰的页面 = 以后永不使用的/最长时间内不再被访问的
基于队列实现的；该算法无法实现；只用于评价其他算法
2. 先进先出(FIFO)置换算法【性能差，但实现简单】
被淘汰的页面 = 在内存中驻留时间最久的页面
会出现Belady异常 = 分配的物理块数增大但页故障数不减反增
3. 最久未使用(LRU)置换算法【性能好，但实现复杂】
被淘汰的页面 = 最近最长时间未访问过的页面
性能较好；需要寄存器和栈道硬件支持；堆栈类算法；耗费高因为要对所有页排序
4. 时钟(CLOCK)置换算法【FIFO和LRU的结合】

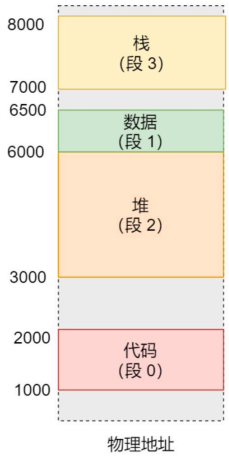

虚拟存储器
其他知识点
1 抖动/颠簸
定义
在页面置换时，出现频繁的页面调度行为
产生原因
系统中同时运行的进程太多—>分配给每个进程的物理块太少—>进程在运行时频繁出现缺页—>频繁的调动页面
主要原因是因为页面置换算法不合理
对换区大小和进程优先级都与抖动无关
2 工作集
定义
在某段时间间隔内，进程要访问的页面集合
如何确定工作集?
基于局部性原理，用最近访问过的页面来确定
有什么作用?
1. 工作集反映了进程在接下来一段时间内很可能频繁访问的页面集合
2. 为了防止抖动现象，要使分配给进程的物理块数>工作集大小
防止抖动方法
3 内存映射文件
定义
与虚拟内存有些相似，将磁盘文件的全部或部分内容与进程虚拟地址空间的某区域建立映射关系
作用
可以之间访问被映射的文件，而不必执行文件I/O操作，也无序对文件内容进行缓存处理
优点
适合用来管理大尺寸文件
4 虚拟存储器性能影响因素
1. 页面较大—>缺页率较低—>可以减少页表长度，但似得页内碎片增大
2. 页面较小—>缺页率较高
—>可以减少内存碎片，提高内存利用率
—>使得页表过长，占用大量内存
3. 分配给进程的物理块数越多，缺页率就越低
4. 分配给进程的物理块数超过某个值时，对缺页率的改善并不明显
5. 好的页面置换算法可以使进程在运行过程中具有较低的缺页率
6. LRU，CLOCK将未来可能要用到的进程保存在内存中，可以提高页面的访问速度
7. 编写程序的局部化程度越高，执行时的缺页率越低
8. 存储和访问尽量使用系统的访问方式（如都按行存储就按行访问）
5 地址翻译（书P211）
结合计组食用

分段

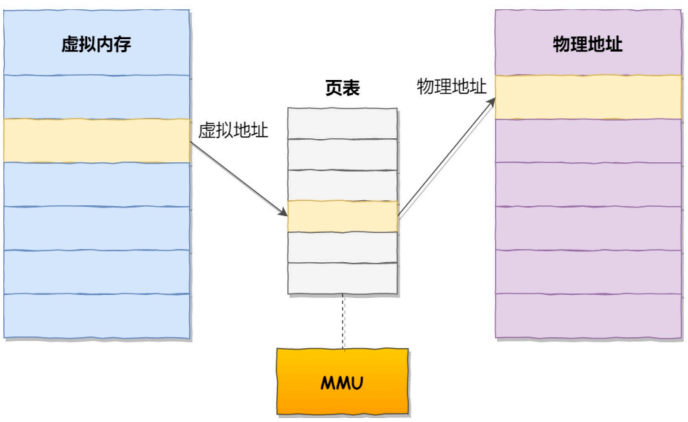

CPU
虚拟地址
段选择因子
段内偏移量
段号
特权等标志位
段表
段内描述符
段基地址
段界限
特权级 DPL
偏移量
起始地址
段
高地址
段界限
低地址

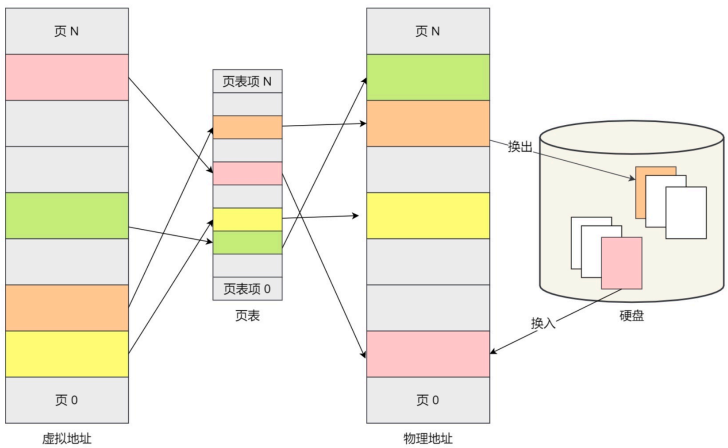

栈
(段 3)
堆
(段 2)
数据
(段 1)
代码
(段 0)
虚拟地址


段基地址
段界限
段号 0
1000
1000
段号 1
6000
500
段号 2
3000
3000
段号 3
7000
1000

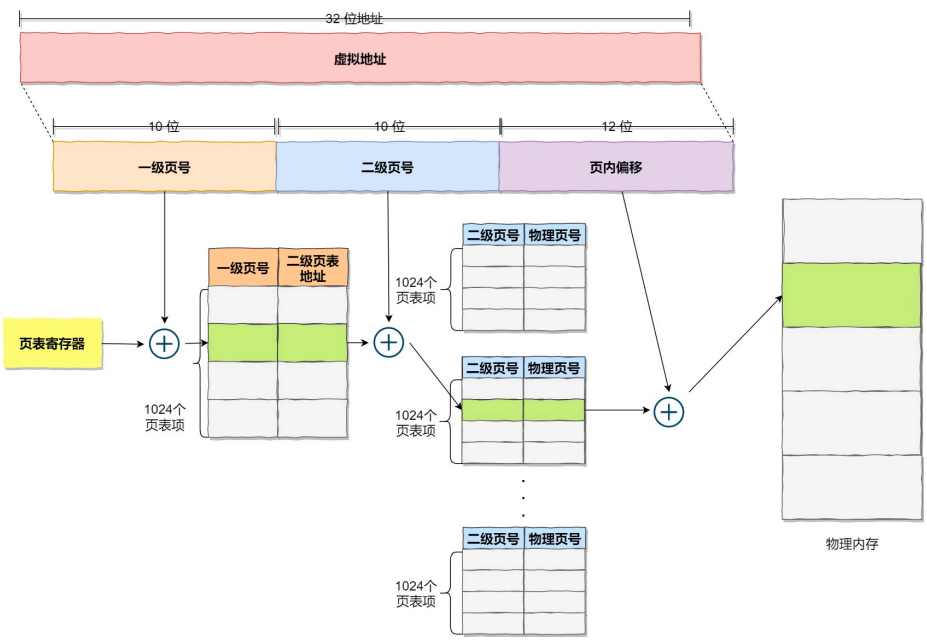

8000
7000
6500
6000
3000
2000
1000
栈
(段 3)
数据
(段 1)
堆
(段 2)
代码
(段 0)
物理地址

分页

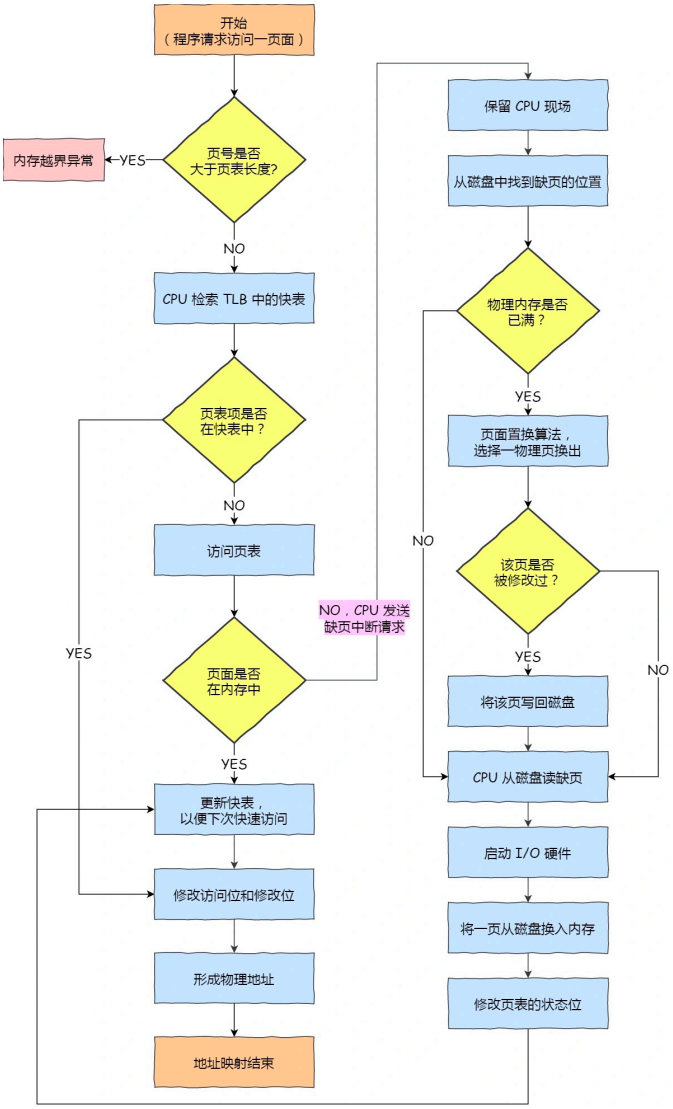

虚拟内存
虚拟地址
页表
物理地址
物理地址
MMU


页 N
页 0
虚拟地址
页表项 N
页表项 0
页表
页 N
页 0
物理地址
换出
换入
硬盘

二级分页


32 位地址
虚拟地址
10 位
10 位
12 位
一级页号
二级页号
页内偏移
页表寄存器
一级页号
二级页表地址
1024个页表项
1024个页表项
1024个页表项
二级页号
物理页号
二级页号
物理页号
二级页号
物理页号
物理内存

如何调入页面？



缺页中断的处理