《操作系统》

主讲教师：**翟高寿**

联系电话：**010-51684177** (办)

电子邮件：**gszhai@bjtu.edu.cn**

制作人：**翟高寿**

制作单位：**北京交通大学计算机学院**

第四章 内存管理

4.1 内存管理概述

4.2 连续分配存储管理方式

4.3 基本分页存储管理方式

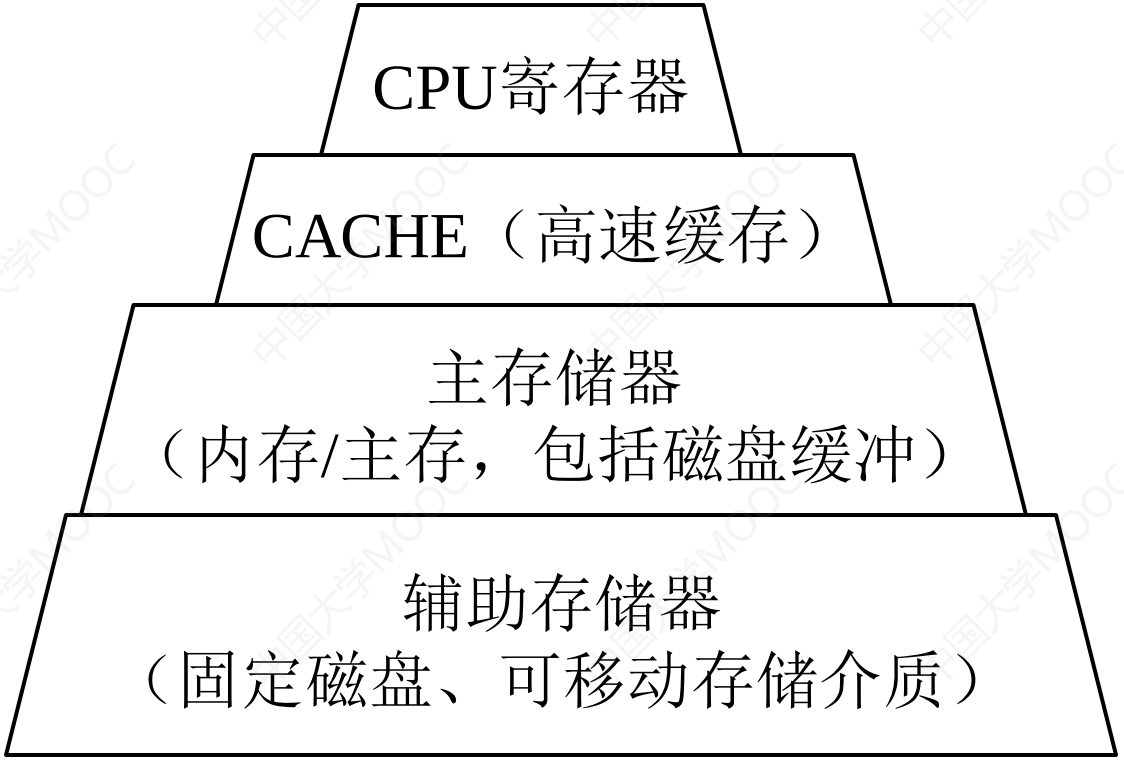
4.4 基本分段存储管理方式

4.5 虚拟存储器概念及关键技术

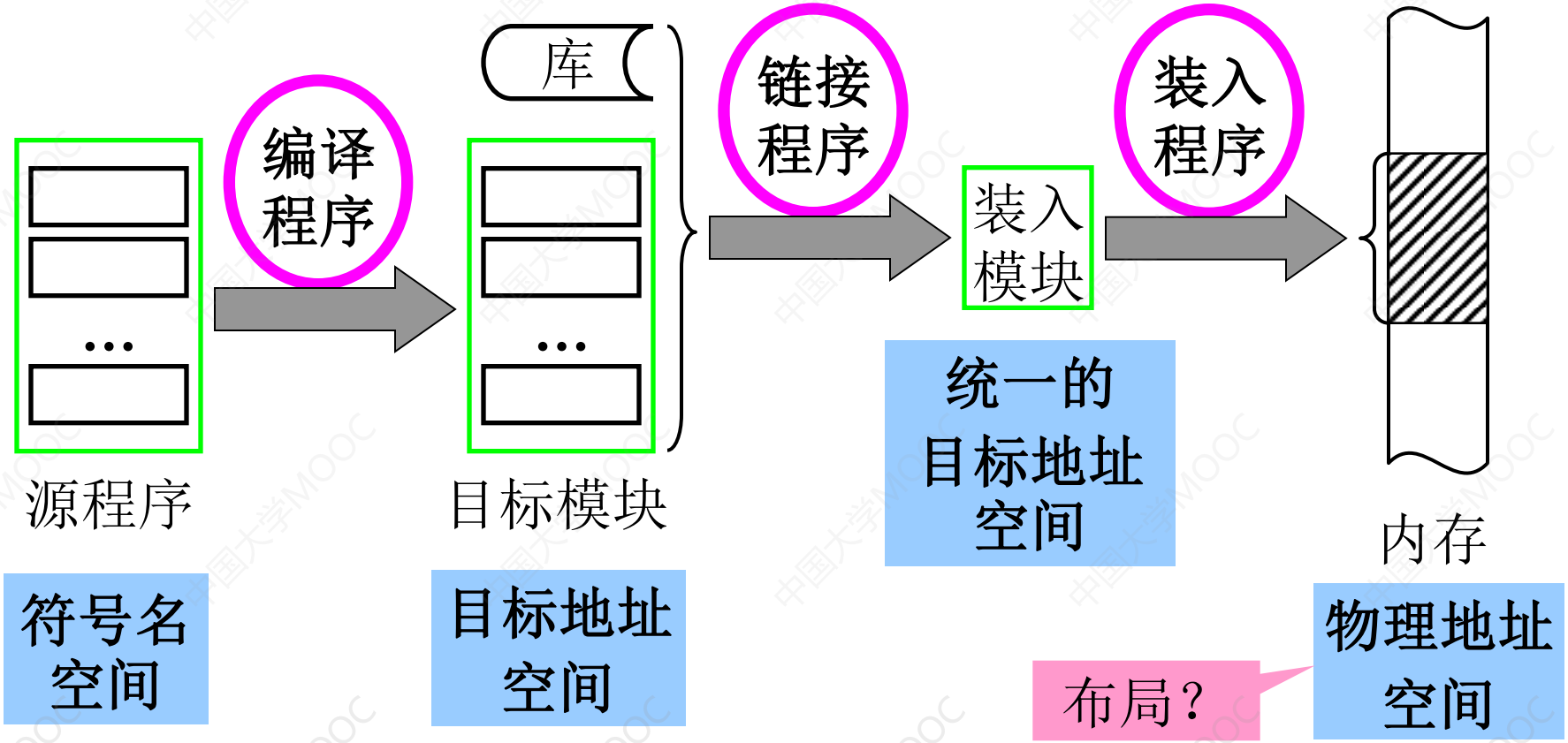
4.6 请求分页存储管理方式

4.7 请求分段存储管理方式

存储器层次结构



用户程序处理过程……？


编译程序
库
链接程序
装入模块
装入程序
源程序
目标模块
统一的目标地址空间
内存
符号名空间
目标地址空间
布局？
物理地址空间

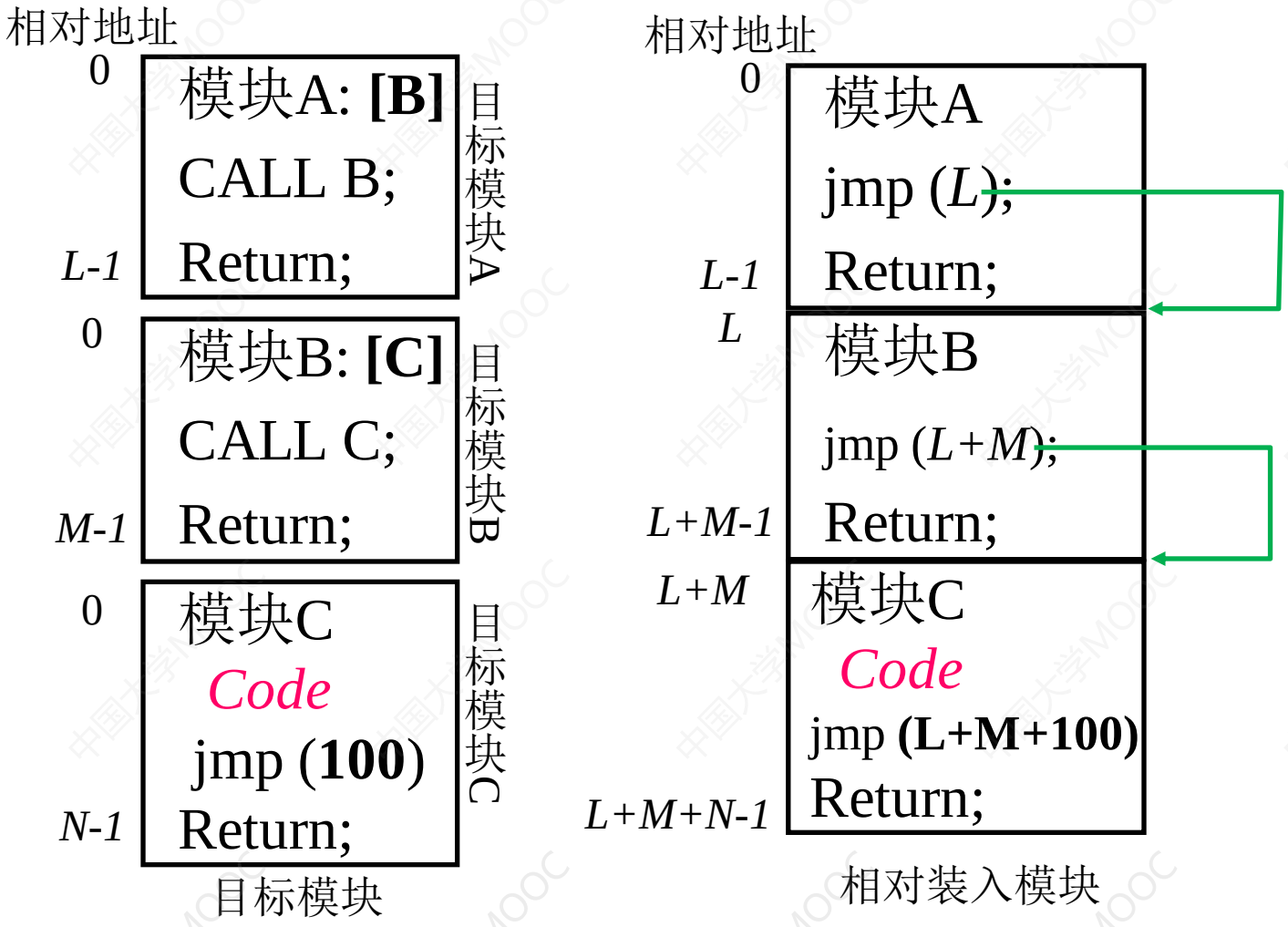
- 给定地址 DS:SI = 0x0200:0x0230
- 实模式下地址转换
  - 0x0200 << 4 + 0x0230 = 0x02230
- 保护模式下地址转换
  - 由0x0200 = （0000 0010 0000 0000）$_2$

    查找全局描述符表第64项描述符给出的分段基址，加上0x0230才是线性地址（若未采用分页机制，就是物理地址）
- 内存管理方式
  - 实模式和保护模式

程序的链接

- 链接过程
  - 根据外部访问符号名表，将经过编译或汇编得到的一组目标模块以及它们所需要的库函数，装配成一个完整的装入模块

程序的链接过程


相对地址
0
模块A: [B]
CALL B;
L-1
Return;
目标模块A
0
模块B: [C]
CALL C;
M-1
Return;
目标模块B
0
模块C
Code
jmp (100)
N-1
Return;
目标模块C
目标模块
相对地址
0
模块A
jmp (L);
L-1
Return;
L
模块B
jmp (L+M);
L+M-1
Return;
L+M
模块C
Code
jmp (L+M+100)
L+M+N-1
Return;
相对装入模块

程序的链接

- 链接过程
  - 根据外部访问符号名表，将经过编译或汇编得到的一组目标模块以及它们所需要的库函数，装配成一个完整的装入模块
- 关键问题
  - 修改相对地址
  - 变换外部调用符号
- 链接方式
  - 静态链接方式、装入/运行时动态链接方式

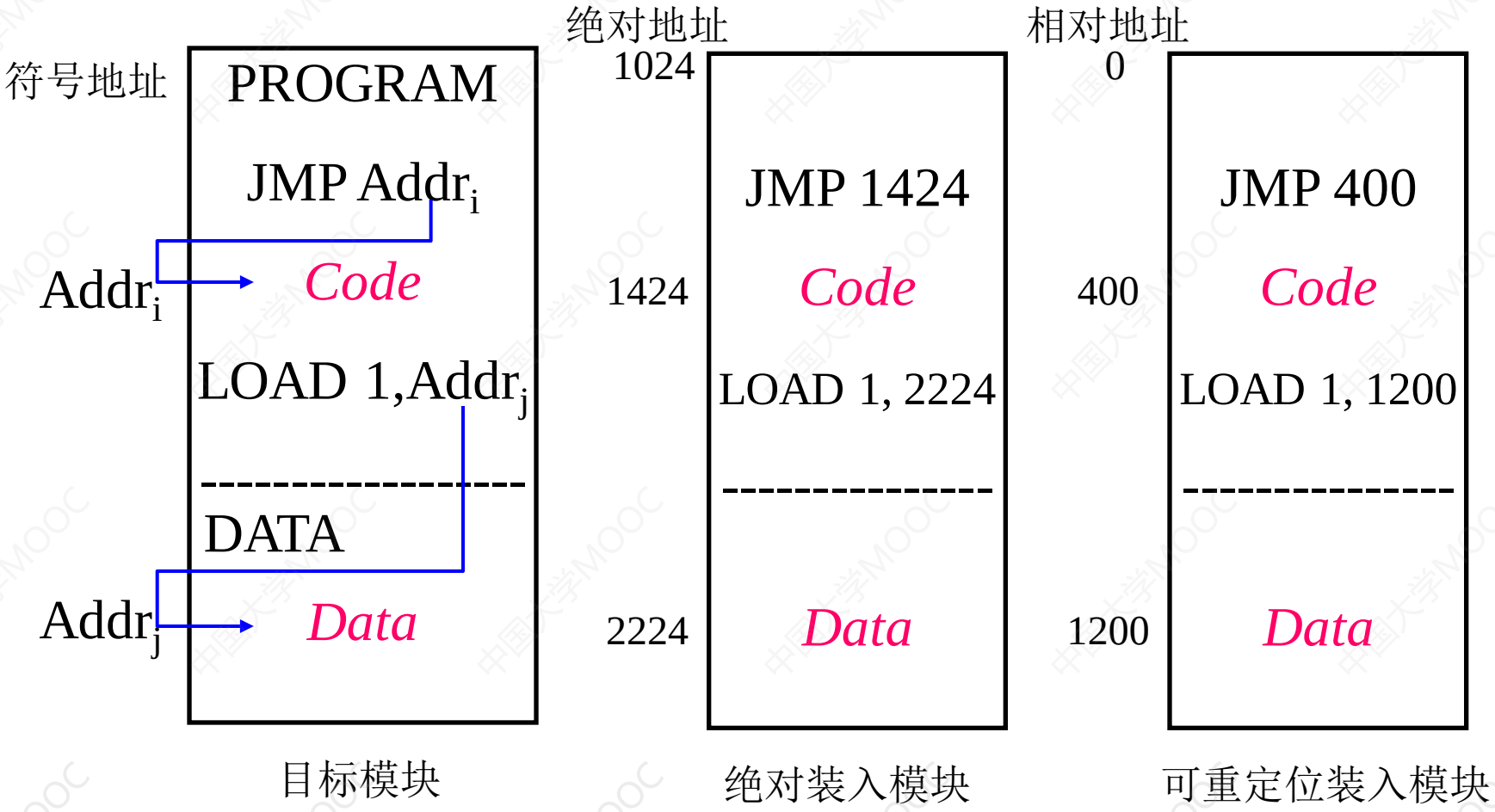
链接方式比较

- 静态链接方式
  - 可执行文件、难以实现“内存” 模块共享
- 装入时动态链接
  - 便于软件版本的修改和更新
  - 便于实现目标模块为多个应用程序共享
- 运行时动态链接
  - 将某些目标模块的链接推迟到执行时根据是否需要再完成，更加有利于内存的有效利用

程序的装入

- 基本目标及相关问题
  - 由装入程序将装入模块载入到内存
  - 装入位置、地址变换（重定位）及时机
- 关键概念
  - 相对地址、绝对地址、重定位及其寄存器
- 装入方式
  - 绝对装入方式（单道程序环境）
  - 静态可重定位装入方式（多道程序环境）
  - 动态运行时装入方式（运行中移动位置）

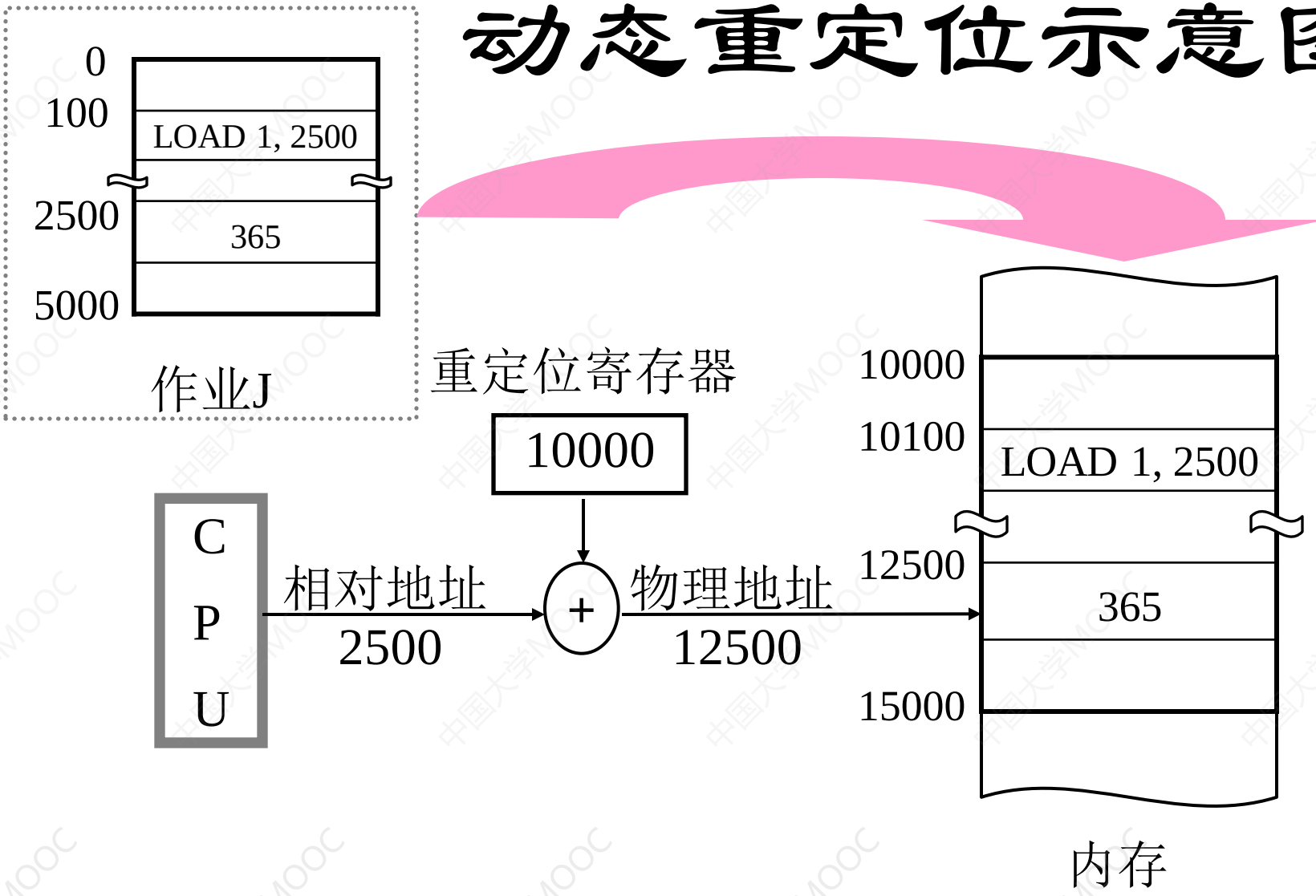
绝对装入模块和可重定位装入模块



（静态/动态）可重定位

- 重定位
  - 程序装入或执行时对装入模块或目标程序中的指令及数据地址的修改过程
- 静态重定位
  - 由重定位装入程序在将装入模块装入内存时一次性完成重定位
  - [需要连续存储空间X]，装入后不能移动
- 动态重定位
  - 需要特殊硬件（地址变换机构）支持，以保证地址转换不会影响指令的执行速度
  - 便于动态链接和代码共享

动态重定位示意图


0
100
LOAD 1, 2500
2500
365
5000
作业J
重定位寄存器
10000
C
P
U
相对地址
2500
+
物理地址
12500
10000
10100
LOAD 1, 2500
12500
365
15000
内存

操作系统内存管理功能要求

- 内存分配
  - 使各得其所、提高利用率及适应动态增长要求
  - 连续分配/离散分配方式
- 地址映射
  - 逻辑地址转换为物理地址，与分配方式相关
- 内存保护
  - 基于地址的保护、存取访问控制保护
- 内存扩充
  - 覆盖技术、对换技术、虚拟存储技术

作业题

- **4.1** 谈谈你对程序处理过程及内存管理相关概念与关键问题的认识与理解。同时并就各种可能的程序链接方式和程序装入方式进行比较和展开讨论。

第四章 内存管理

4.1 内存管理概述

4.2 连续分配存储管理方式

4.3 基本分页存储管理方式

4.4 基本分段存储管理方式

4.5 虚拟存储器概念及关键技术

4.6 请求分页存储管理方式

4.7 请求分段存储管理方式

4.2 连续分配存储管理方式

4.2.1 单一连续分配存储管理

4.2.2 固定分区分配存储管理

4.2.3 动态分区分配存储管理

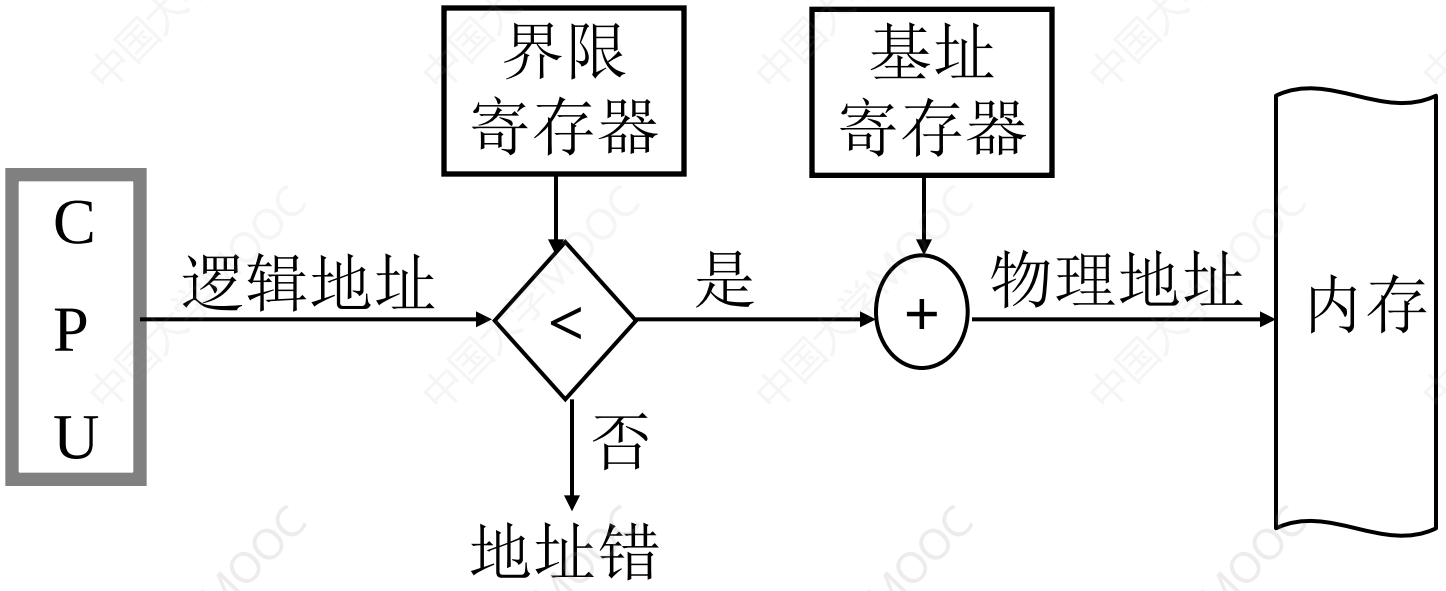
4.2.4 动态可重定位分区分配

4.2.5 对换技术与覆盖技术

4.2.6 伙伴系统

单一连续分配方式

- 内存划分为系统区和用户区
- 整个用户区为一个用户独占，仅驻留一道程序
- 静态链接和动态**重定位**技术、存储器保护措施
- 仅适用于单用户、单任务操作系统中



4.2 连续分配存储管理方式

4.2.1 单一连续分配存储管理

4.2.2 固定分区分配存储管理

4.2.3 动态分区分配存储管理

4.2.4 动态可重定位分区分配

4.2.5 对换技术与覆盖技术

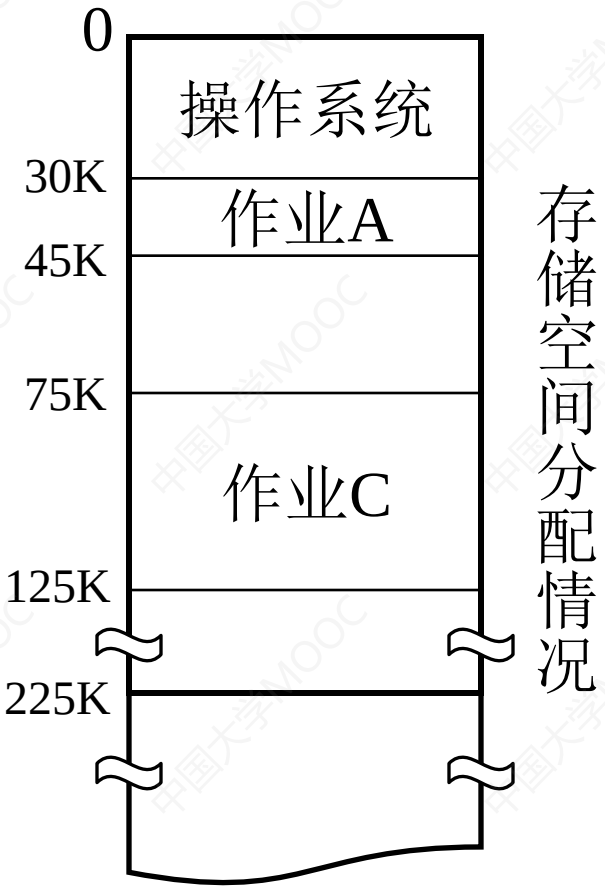
4.2.6 伙伴系统

固定分区分配方式

作业B: 26KB

- 用户区分为若干固定区域
- 每个分区可装入一道作业
- 分区划分方法(等分/不等分)
- 分区说明表与内存分配算法
- 可用于多道程序存储管理

分区号	大小KB	始址K	状态
1	15	30	已分配
2	30	45	已分配
3	50	75	已分配
4	100	125	空闲
…	…	…	…



A:B:C = 15:30:50

4.2 连续分配存储管理方式

4.2.1 单一连续分配存储管理

4.2.2 固定分区分配存储管理

4.2.3 动态分区分配存储管理

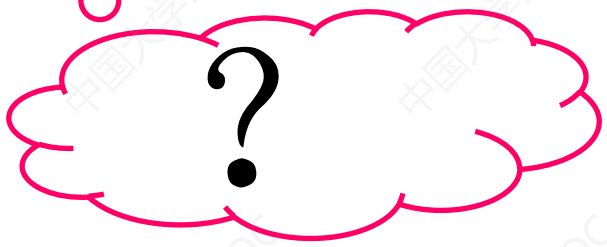
4.2.4 动态可重定位分区分配

4.2.5 对换技术与覆盖技术

4.2.6 伙伴系统

动态分区分配方式

- 基本思想
  - 根据进程的实际需求，动态地对内存空间进行分配、回收及划分
- 关键问题
  - 分区分配用数据结构
  - 分区分配算法
  - 分区分配与回收操作
  - 碎片（零头）处理

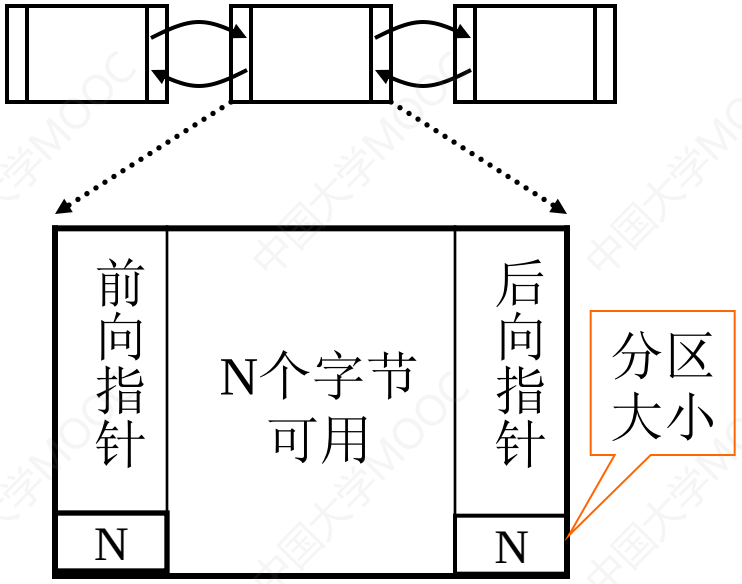


分区分配用数据结构

空闲分区表

分区号	大小KB	始址K
1	64	44
2	24	132
3	40	210
4	30	270
…	…	…

空闲分区链



分区分配算法

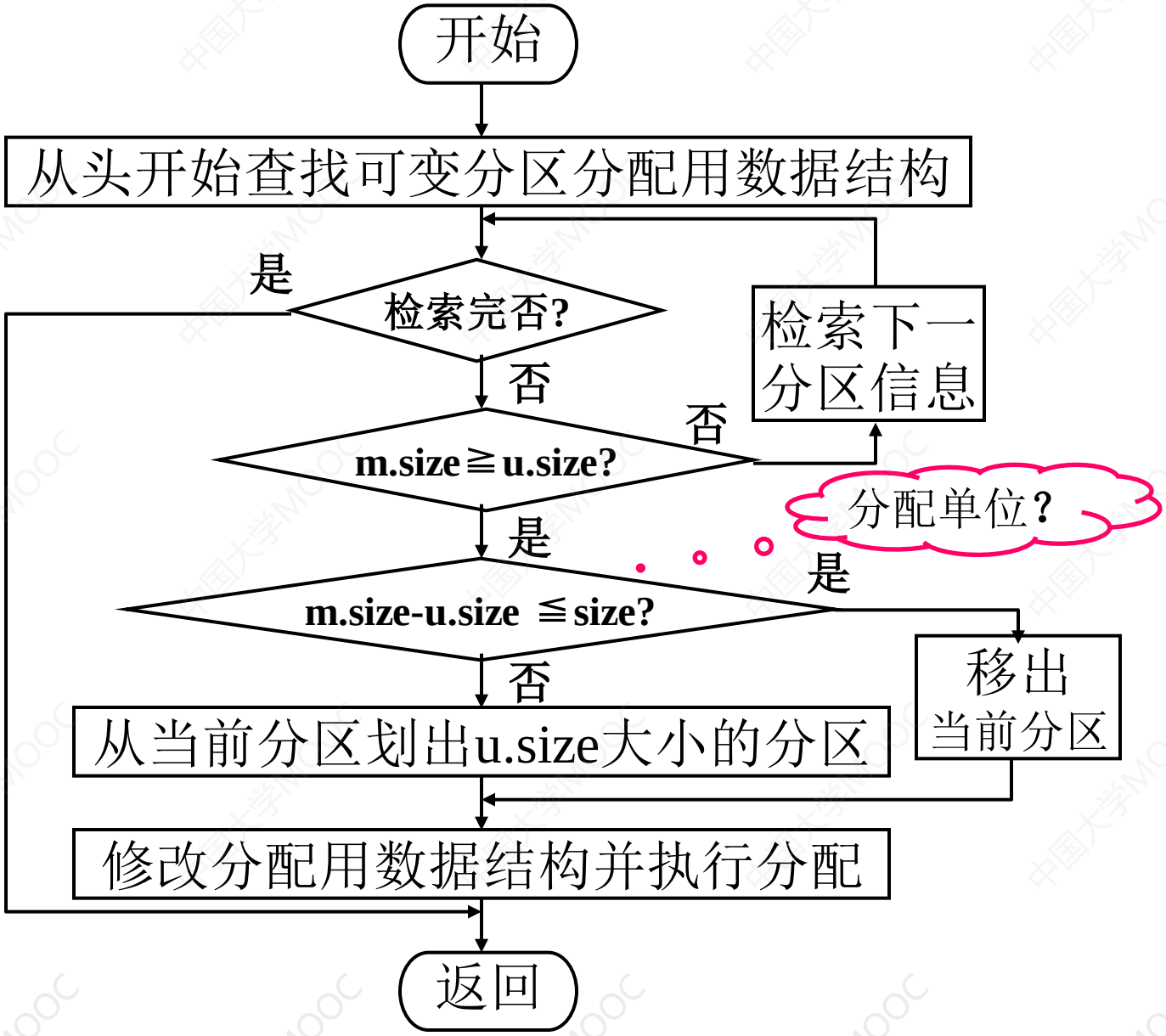
快速适应算法

- 首次适应算法FF
  - 要求空闲分区链以地址递增次序链接
  - 查找开销大，但有利于大作业分配
- 循环首次适应算法
  - 首次适应 + 起始查寻指针 + 循环查找
  - 减少查找开销，但不利于大作业分配
- 最佳适应算法

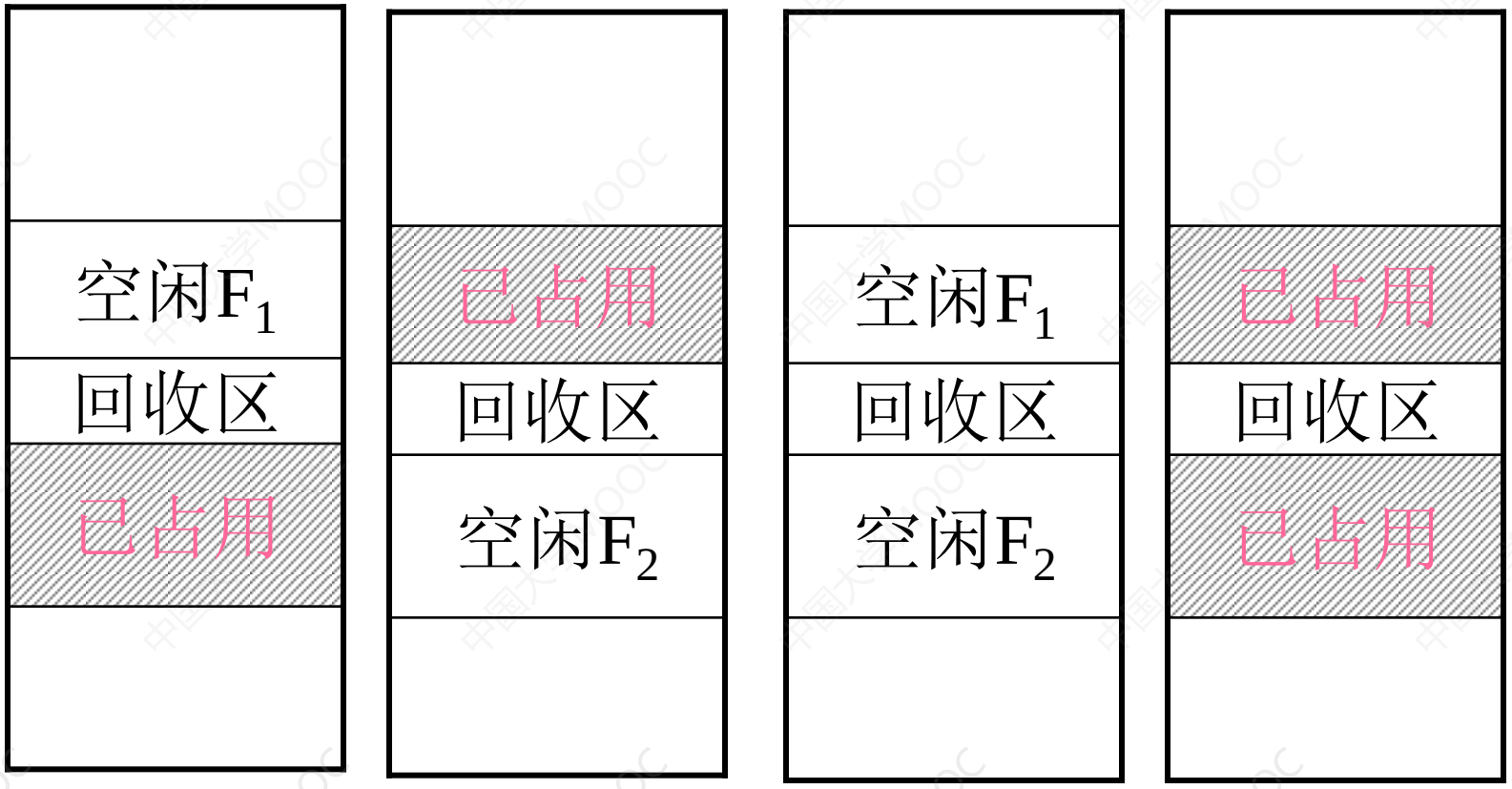
最坏适应算法？

  - 追求既能满足要求且又最小的空闲分区
  - 要求空闲分区按大小递增次序链接
  - 微观意义上的最佳与宏观上的零头问题

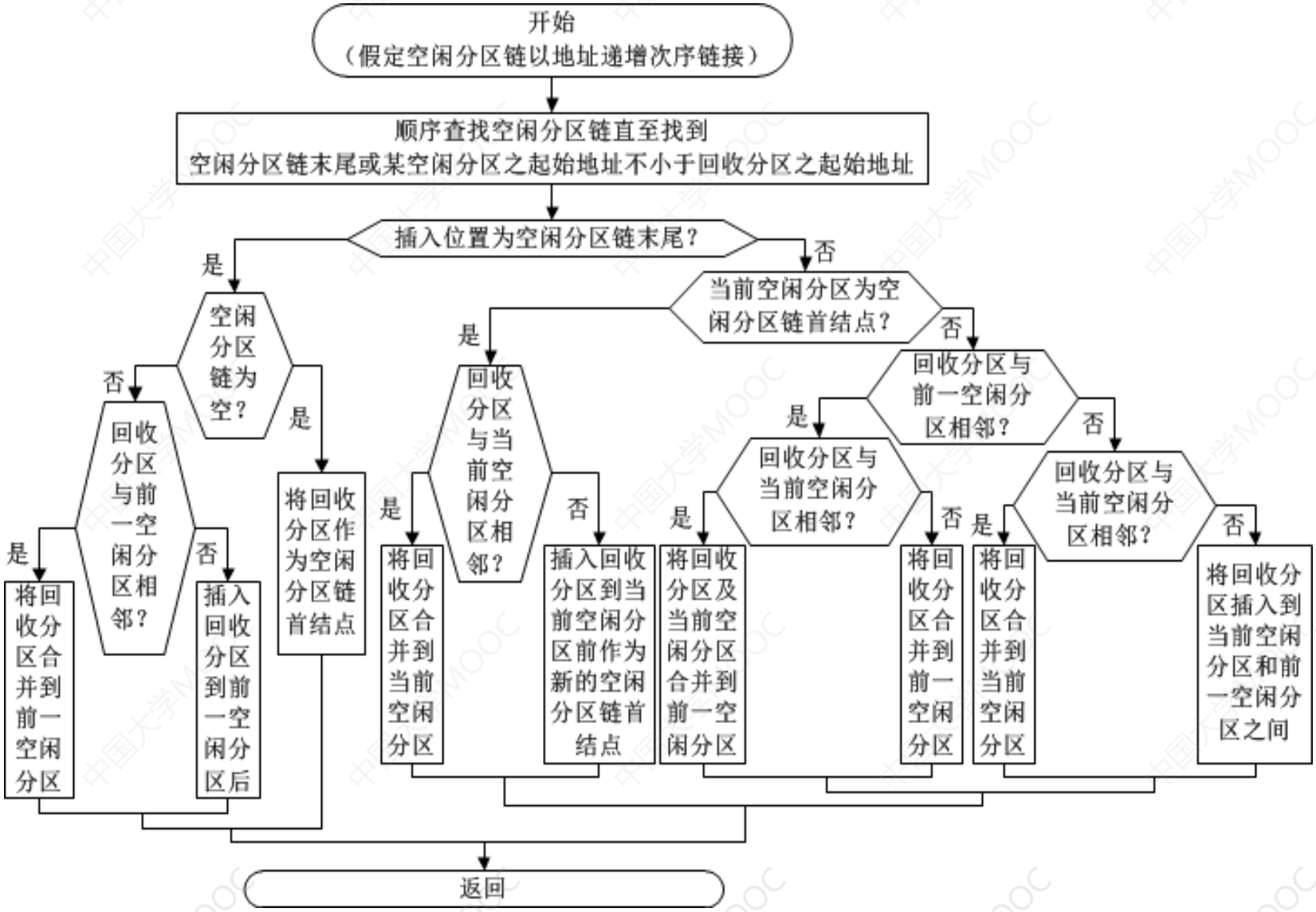
动态分区内存分配流程


开始
从头开始查找可变分区分配用数据结构
是
检索完否?
否
检索下一
分区信息
否
m.size≧u.size?
分配单位?
是
是
m.size-u.size ≦size?
否
移出
当前分区
从当前分区划出u.size大小的分区
修改分配用数据结构并执行分配
返回

动态分区内存回收情况



动态分区内存回收流程


开始
（假定空闲分区链以地址递增次序链接）
顺序查找空闲分区链直至找到
空闲分区链末尾或某空闲分区之起始地址不小于回收分区之起始地址
插入位置为空闲分区链末尾？
是
否
空闲分区链为空？
是
否
将回收分区作为空闲分区链首结点
回收分区与前一空闲分区相邻？
是
否
将回收分区合并到前一空闲分区
插入回收分区到前一空闲分区后
当前空闲分区为空闲分区链首结点？
是
否
回收分区与当前空闲分区相邻？
是
否
将回收分区合并到当前空闲分区
插入回收分区到当前空闲分区前作为新的空闲分区链首结点
回收分区与前一空闲分区相邻？
是
否
回收分区与当前空闲分区相邻？
是
否
将回收分区及当前空闲分区合并到前一空闲分区
将回收分区合并到前一空闲分区
回收分区与当前空闲分区相邻？
是
否
将回收分区合并到当前空闲分区
将回收分区插入到当前空闲分区和前一空闲分区之间
返回

4.2 连续分配存储管理方式

4.2.1 单一连续分配存储管理

4.2.2 固定分区分配存储管理

4.2.3 动态分区分配存储管理

4.2.4 动态可重定位分区分配

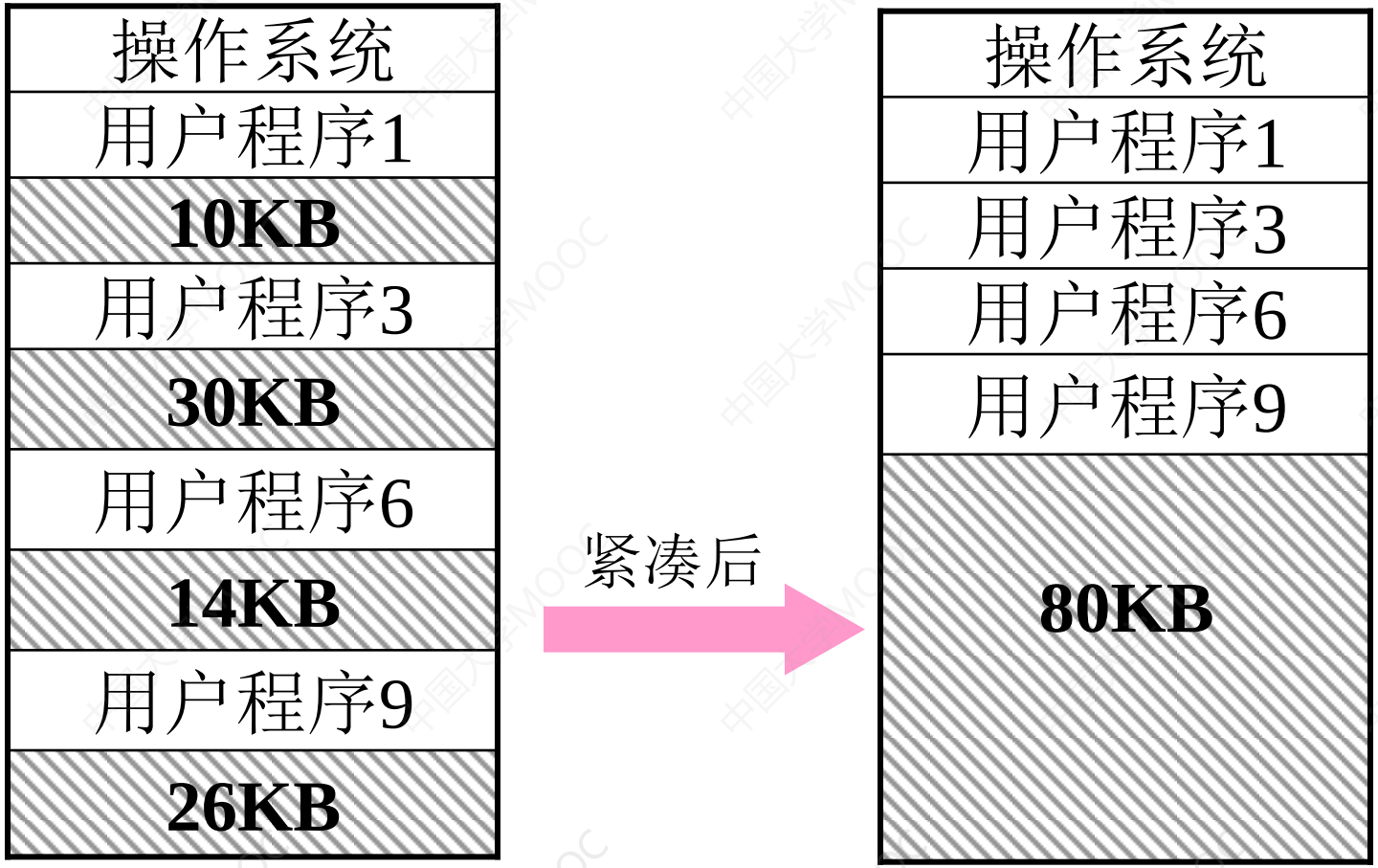
4.2.5 对换技术与覆盖技术

4.2.6 伙伴系统

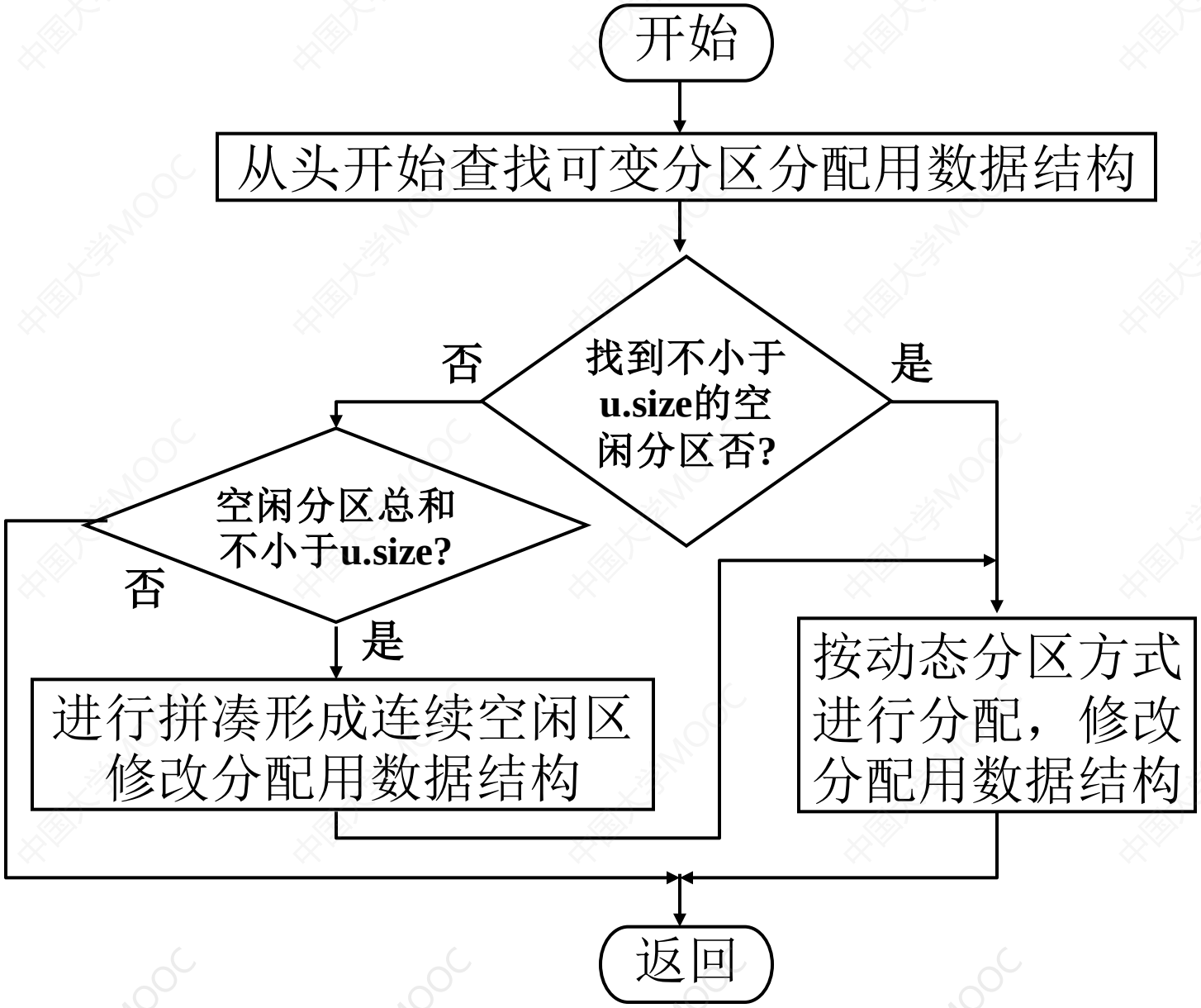
动态重定位分区分配方式

- 紧凑技术
  - 连续分配要求程序装入内存空间的连续性
  - 分区分配产生的零头/碎片问题
  - 通过移动把多个分散拼接成大分区
  - 用户程序内存地址变化及地址修正问题
- 动态重定位
  - 动态运行时装入方式及重定位寄存器
- 动态重定位分区分配算法
  - 动态分配分区算法 + 紧凑功能

紧凑（拼接）技术


操作系统
用户程序1
10KB
用户程序3
30KB
用户程序6
14KB
用户程序9
26KB
紧凑后
操作系统
用户程序1
用户程序3
用户程序6
用户程序9
80KB

动态重定位分区分配流程



4.2 连续分配存储管理方式

4.2.1 单一连续分配存储管理

4.2.2 固定分区分配存储管理

4.2.3 动态分区分配存储管理

4.2.4 动态可重定位分区分配

4.2.5 对换技术与覆盖技术

4.2.6 伙伴系统

多道程序环境下的对换

- 对换的概念及意义
  - 内存 ⇔（进程、程序、数据）⇔ 外存
  - 提高内存利用率
- 对换实现机制
  - UNIX：对换进程（中级调度）
- 对换实现方式
  - 进程对换：分时系统
  - 页面/分段对换：虚拟存储技术

对换空间的管理

- 文件区和对换区
  - 功能、管理目标及管理方式
- 对换区使用情况数据结构
  - 空闲盘区表/链（盘块组为基本单位）
- 对换区分配与回收操作
  - 分配算法
  - 分配操作
  - 回收操作

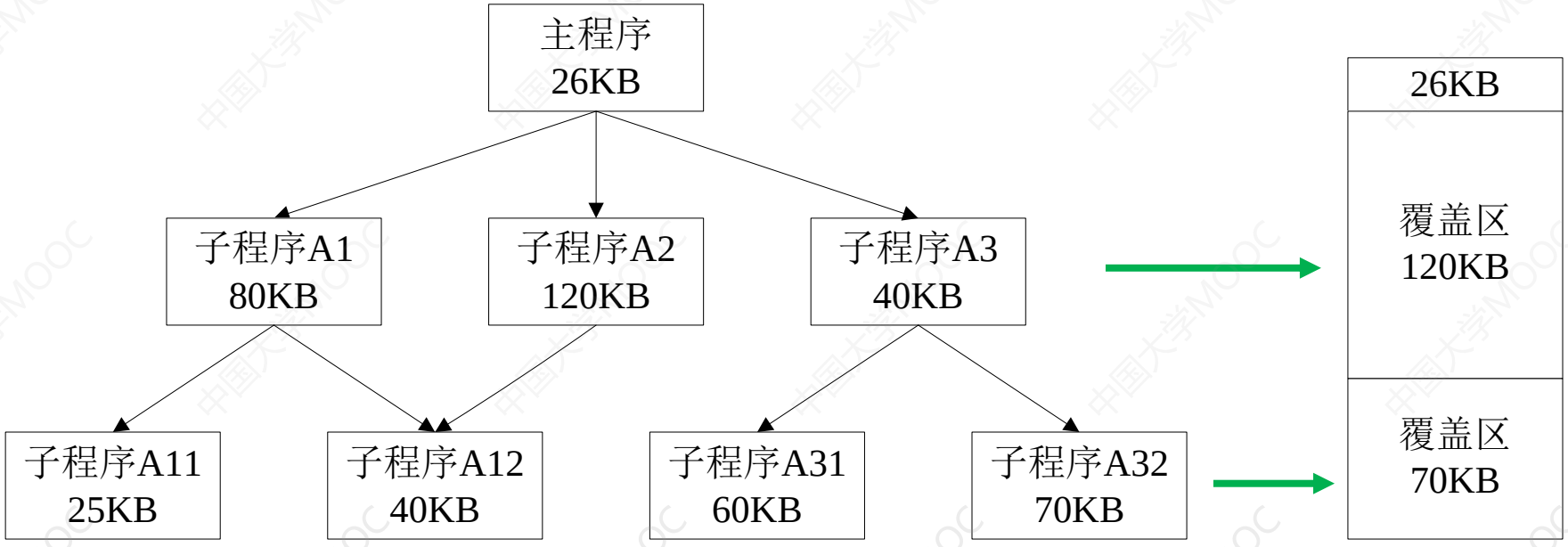
序号	第一空闲盘块号	空闲盘块数
0	3	3
1	8	5
2	16	2
3	…	…

进程的换出和换入

- 进程的换出
  - 被换出进程的选择（进程状态+优先级+内存驻留时间）
  - 换出过程（换出非共享或不再共享的程序及数据段⇒对换空间申请⇒换出⇒内存释放⇒内存分配数据结构及PCB修改）
- 进程的换入
  - 被换入进程的选择（进程状态+换出时间）
  - 换入过程（内存申请⇒换入⇒ PCB修改）

覆盖技术

- 基本思想及概念
  - 让那些不会同时执行的程序段共享一块内存
  - 可以互相覆盖的程序段称为覆盖段
  - 被共用的内存空间称为覆盖区



4.2 连续分配存储管理方式

4.2.1 单一连续分配存储管理

4.2.2 固定分区分配存储管理

4.2.3 动态分区分配存储管理

4.2.4 动态可重定位分区分配

4.2.5 对换技术与覆盖技术

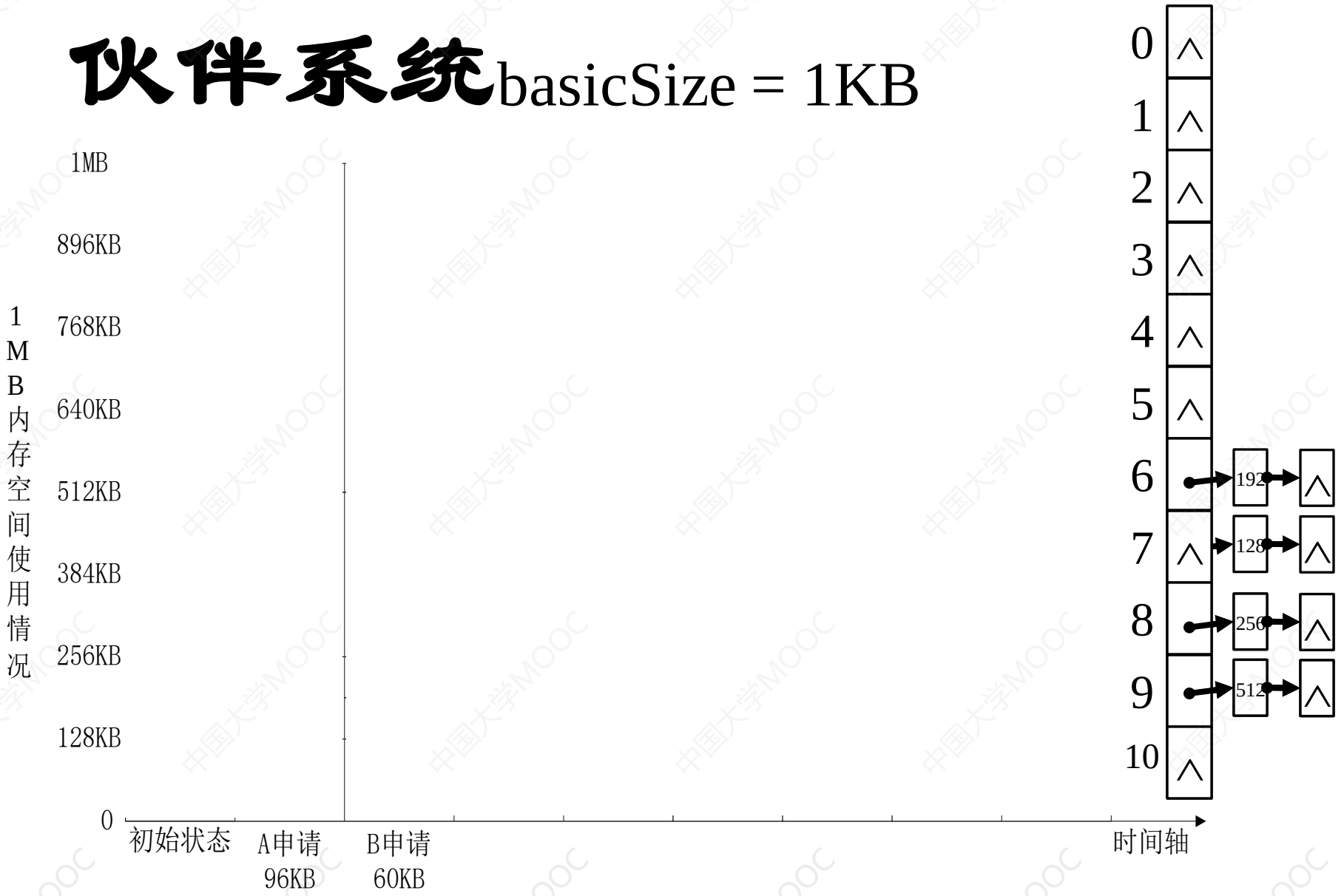
4.2.6 伙伴系统

伙伴系统

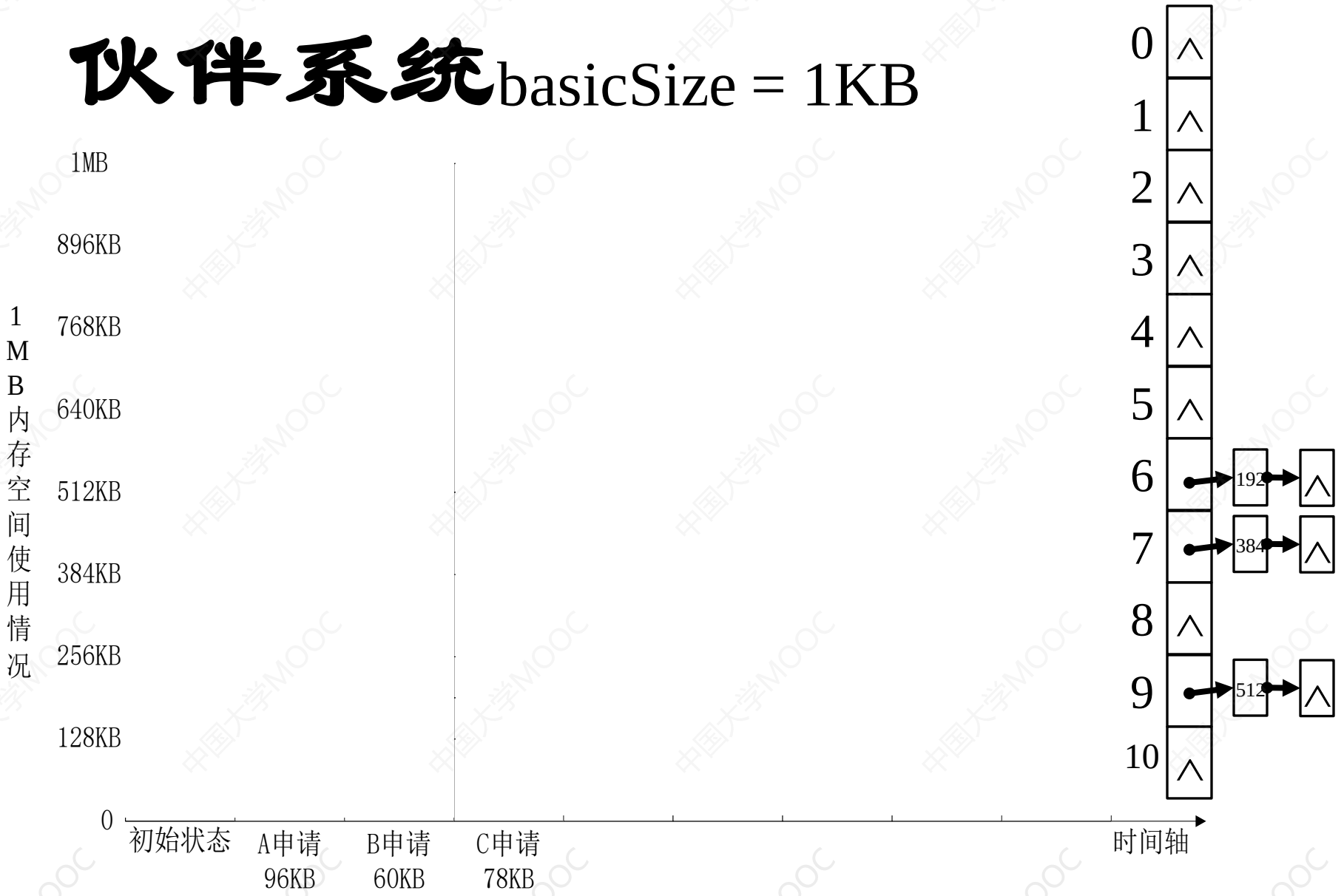
- 引入理由
  - 对固定分区和动态分区方式的折衷
- 分区大小
  - 要求分别设立和保留大小为基本大小basicSize的1倍、2倍、4倍、8倍、16倍等直到可分配内存空间大小所对应的空闲分区链，即要求内存空闲分区链均对应2的整次幂大小的空间。
- 空闲分区查找方法
  - 哈希算法（$\lceil \log_2(request/basicSize) \rceil$）

伙伴系统basicSize = 1KB

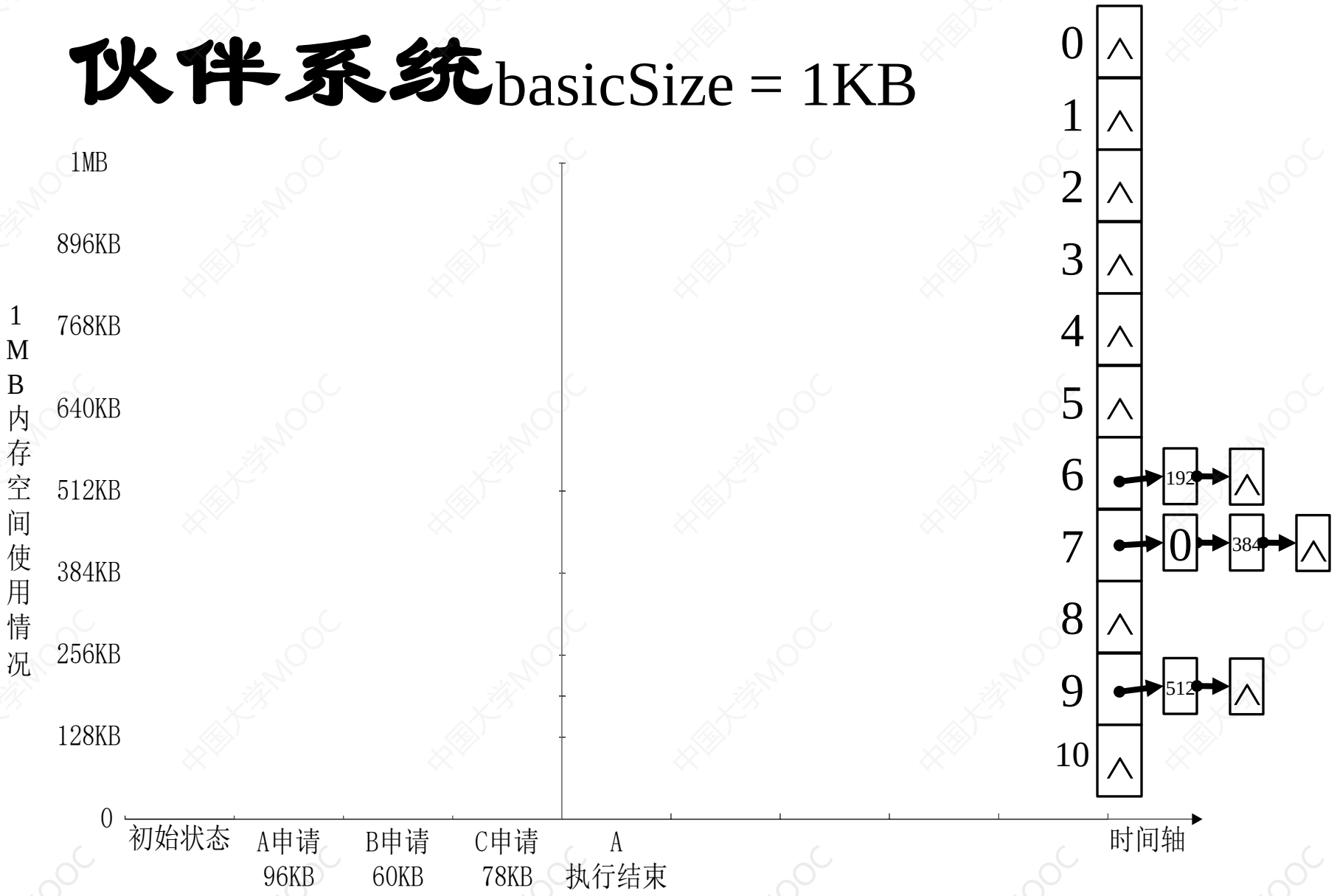
伙伴系统basicSize = 1KB



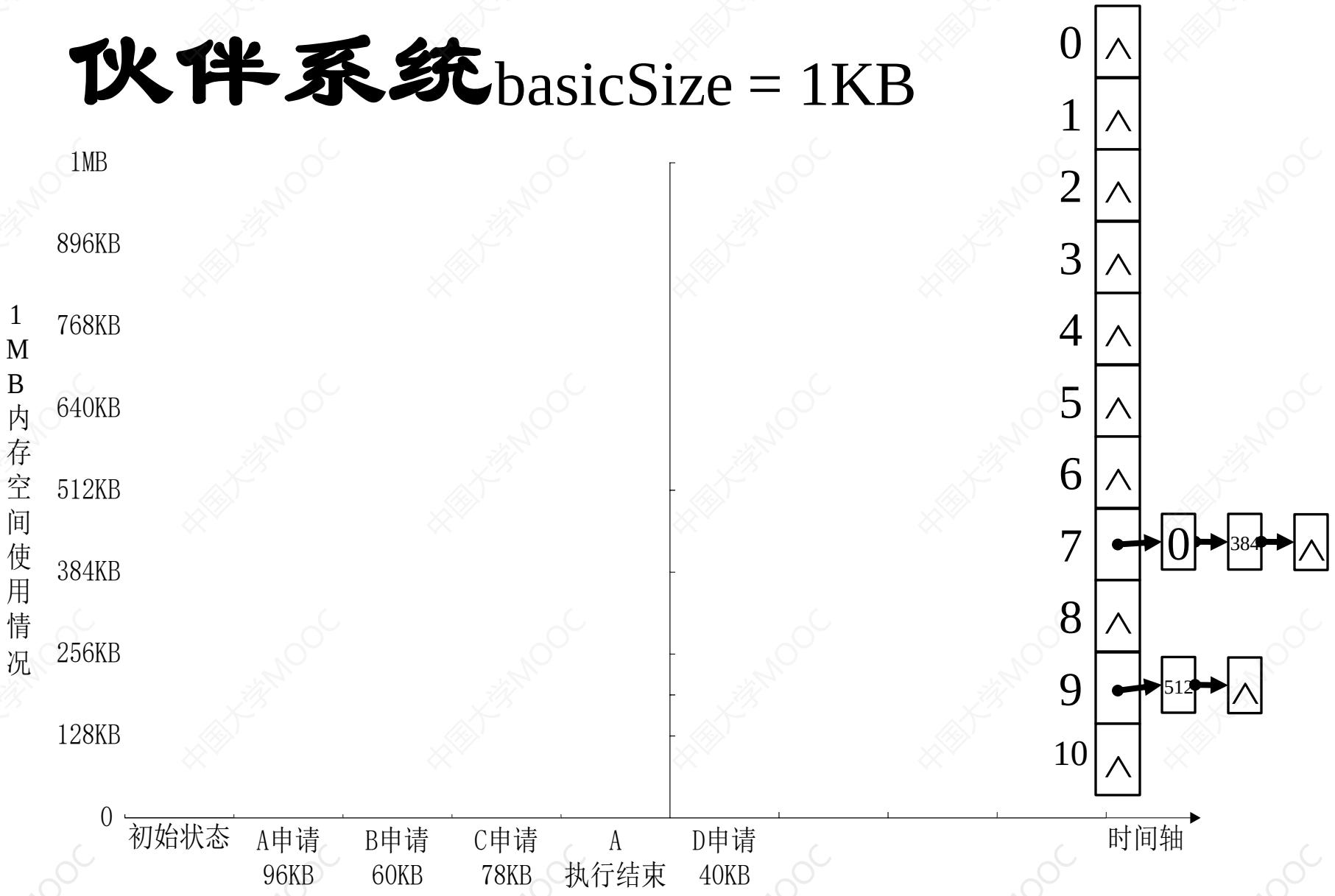
伙伴系统basicSize = 1KB



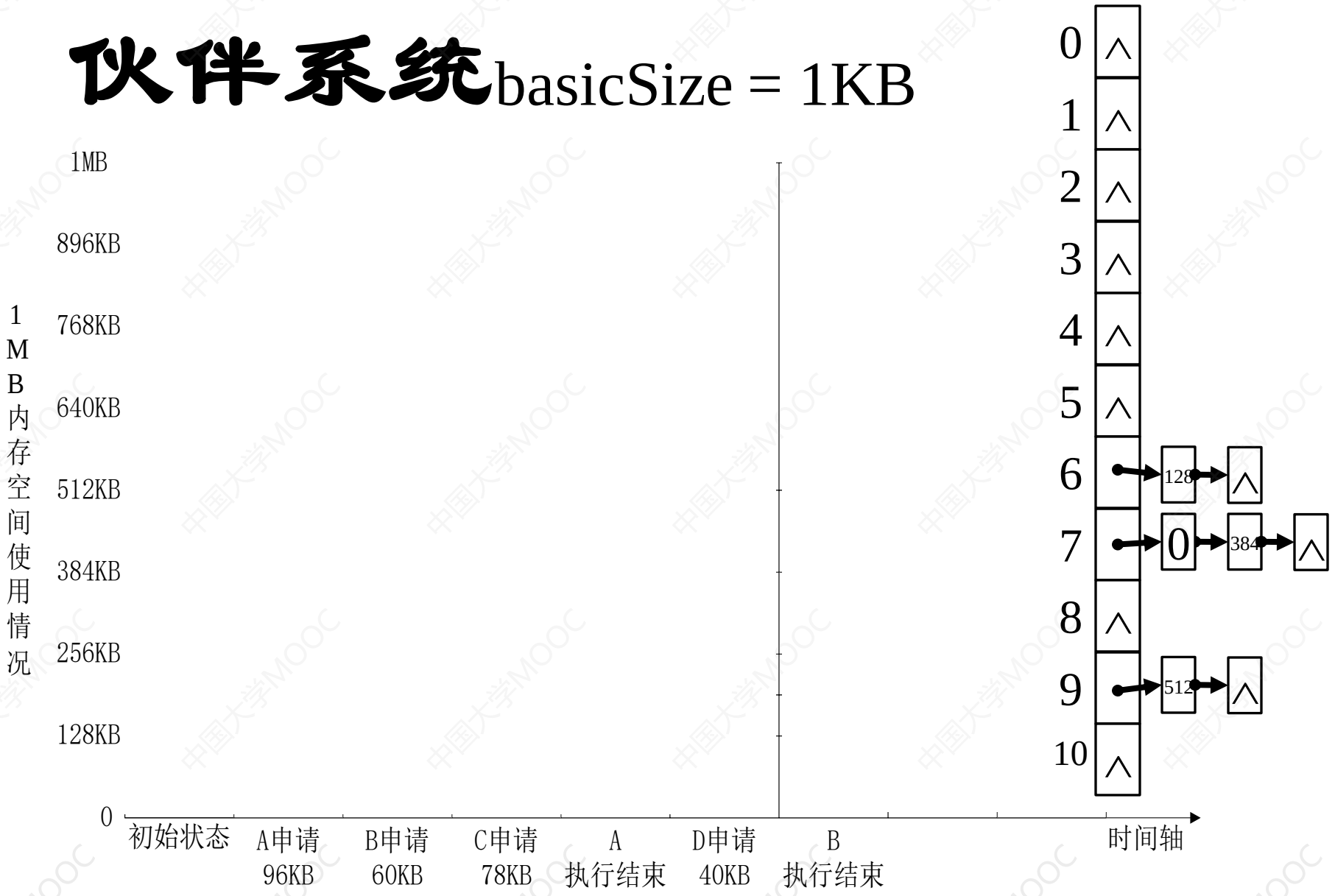
伙伴系统basicSize = 1KB



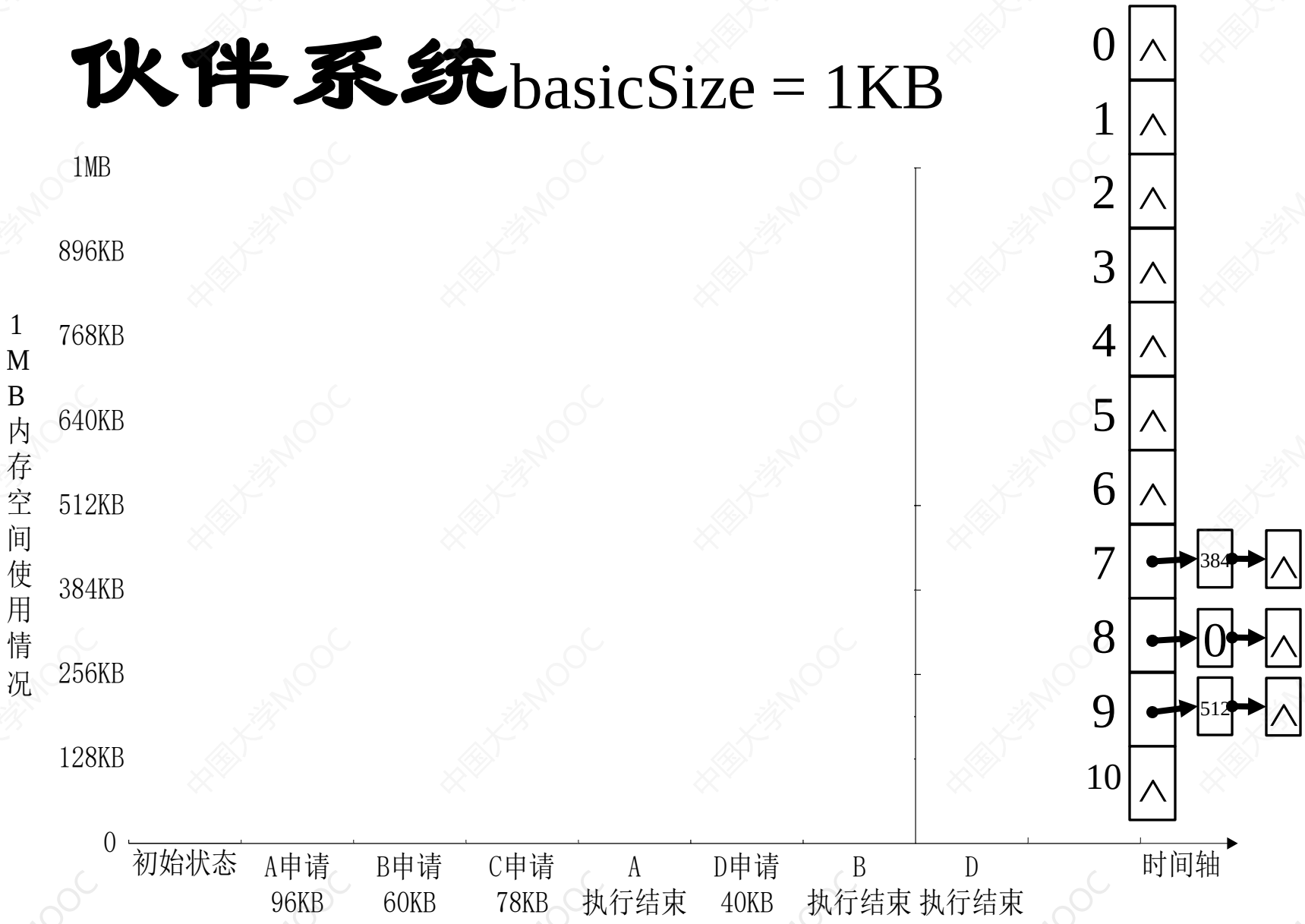
伙伴系统basicSize = 1KB



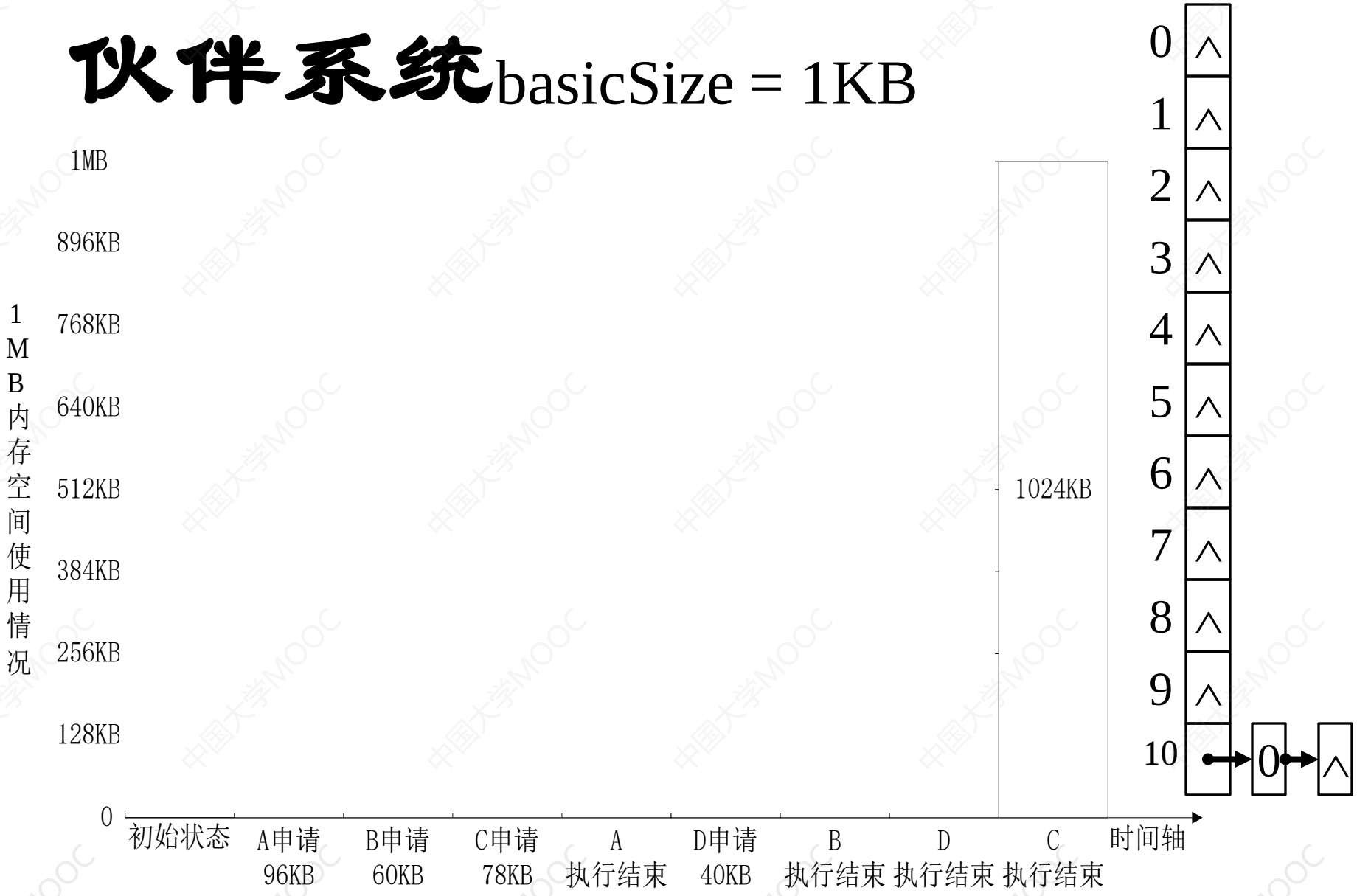
伙伴系统basicSize = 1KB



伙伴系统basicSize = 1KB



伙伴系统basicSize = 1KB



4.2 连续分配存储管理方式

4.2.1 单一连续分配存储管理

4.2.2 固定分区分配存储管理

4.2.3 动态分区分配存储管理

4.2.4 动态可重定位分区分配

4.2.5 对换技术

4.2.6 伙伴系统

作业题

- **4.2** 从内存管理的各个方面比较单一连续分配、固定分区分配、动态分区分配以及动态可重定位分区等存储管理方式的异同，特别注意相关数据结构和算法的理解。
- **4.3** 谈谈你对覆盖技术与对换技术的认识与理解。

实验课题

- **实验课题12**

动态可重定位分区内存管理模拟设计与实现

第四章 内存管理

4.1 内存管理概述

4.2 连续分配存储管理方式

4.3 基本分页存储管理方式

4.4 基本分段存储管理方式

4.5 虚拟存储器概念及关键技术

4.6 请求分页存储管理方式

4.7 请求分段存储管理方式

4.3 基本分页存储管理方式

4.3.1 离散存储管理方式

4.3.2 页面与页表

4.3.3 地址变换机构

4.3.4 两级和多级页表

4.3.5 反置页表

离散存储管理方式

- 连续存储管理方式弊端
  - 碎片问题及紧凑开销
- 离散分配方式
  - 分页存储管理
  - 分段存储管理
  - 段页式存储管理

4.3 基本分页存储管理方式

4.3.1 离散存储管理方式

4.3.2 页面与页表

4.3.3 地址变换机构

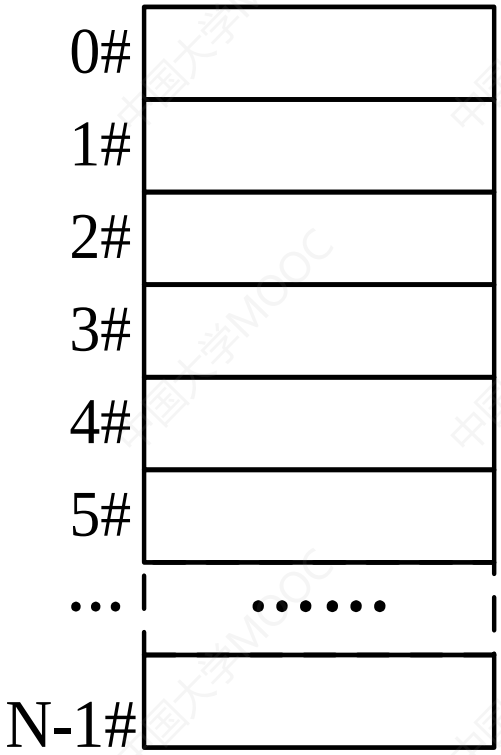
4.3.4 两级和多级页表

4.3.5 反置页表

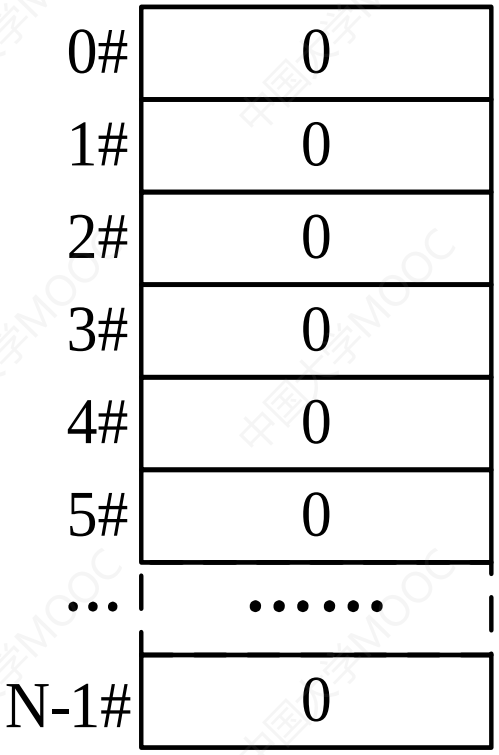
物理块、页面与页表

- 物理块和页面
  - 内存空间及进程逻辑地址空间的划分
  - 物理块编号与页面编号
  - 内存分配与页内碎片
- 内存页表（物理块表）
- 进程页表
  - 页面映射表及其表项(物理块号、存取控制字段)
- 页面大小选择
  - 由机器的地址结构所决定
  - 页面大/小评析（2^9B～8KB之间）

物理块表（内存页表）



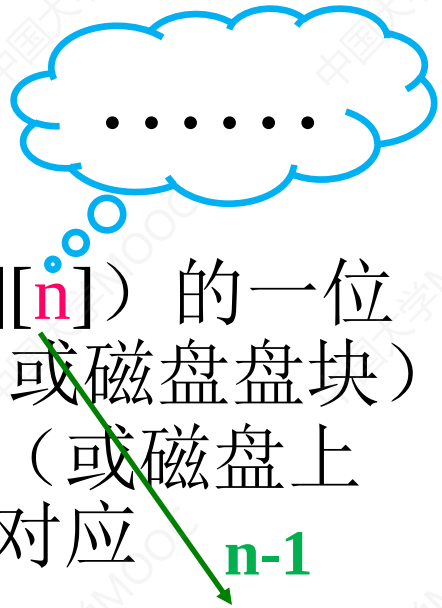
(a) 内存页表基本结构



(b) 内存页表初始化

位示图

- 利用位示图（即二维数组Map[m][n]）的一位（0/1）来表示一个内存物理块（或磁盘盘块）的使用情况，使内存所有物理块（或磁盘上所有盘块）都与一个二进制位相对应



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	0	0	0	1	0	0	1	0	1	0	0	1	1	0
1	1	0	0	1	0	0	0	1	0	1	0	0	1	1	1	1
2	1	1	0	1	1	0	0	1	0	1	0	0	1	0	1	0
…								……								
m-1	1	1	0	1	0	0	1	0	1	0	1	0	0	0	1	0

物理块（盘块）的分配

Var Map: array [m][n] of bit;

- 根据需求计算确定所需物理块数zKs
- 顺序扫描位示图，找出zKs个其值均为空闲“0”的二进制位
- 将所找到的“0”值二进制位Map[i][j]的行/列号转换为与之对应的物理块（盘块）号b：$b = n \times i + j$
- 按物理块（或盘块）号分配物理块（或盘块），同时修改位示图和进程页表

物理块（盘块）的回收

I. 将回收物理块（或盘块）的物理块（或盘块）号b转换为位示图中的行号i和列号j：

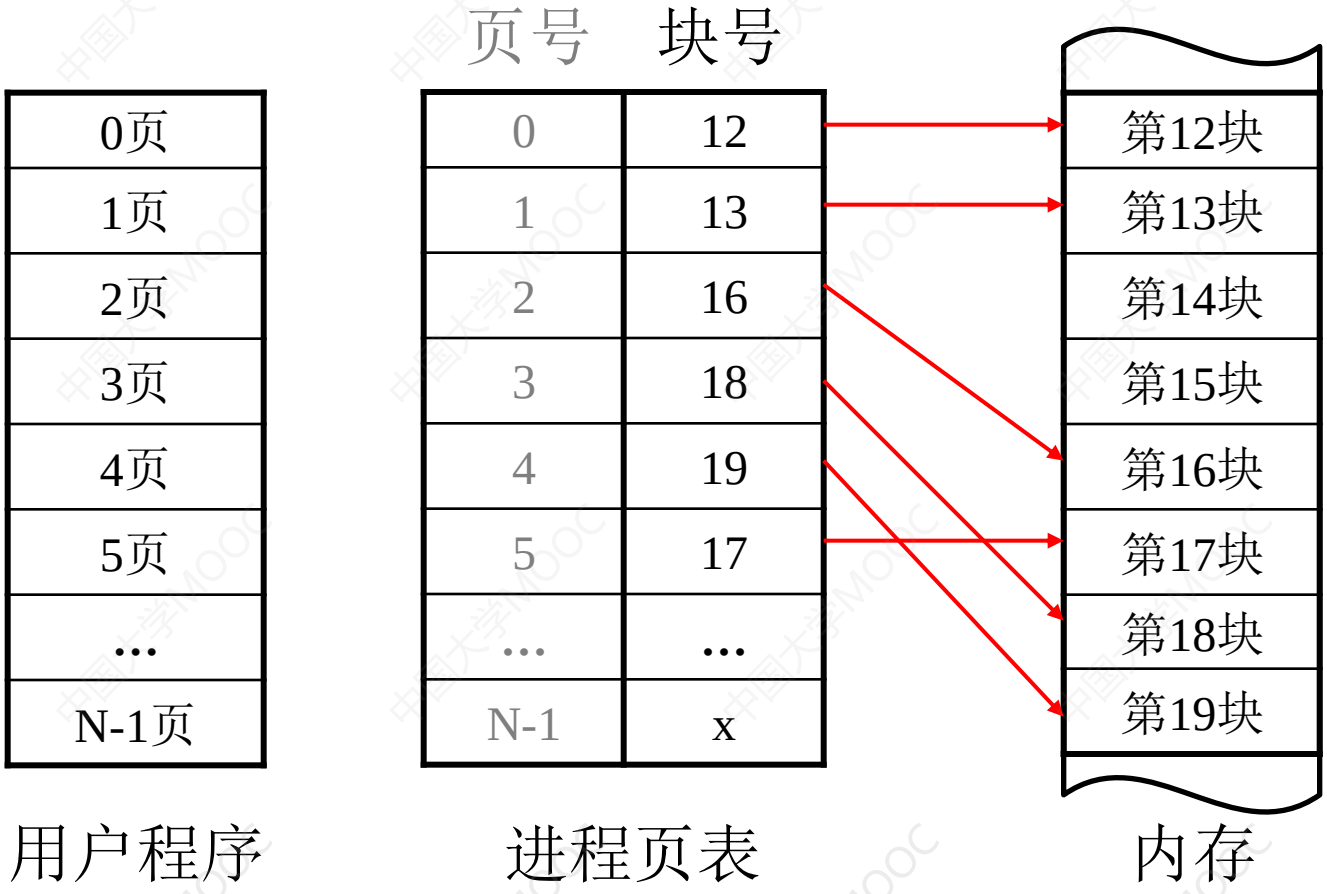
   A. i = b DIV n;

   B. j = b MOD n;

II. 按物理块（或盘块）号回收物理块（或盘块）

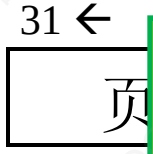
III. 根据回收物理块（或盘块）对应二进制位的行/列号修改位示图和进程页表

（进程）页表


页号
块号
0页
1页
2页
3页
4页
5页
…
N-1页
0
1
2
3
4
5
…
N-1
12
13
16
18
19
17
…
x
第12块
第13块
第14块
第15块
第16块
第17块
第18块
第19块
用户程序
进程页表
内存

分页存储地址结构及地址变换

- 分页存储管理地址结构（32位机、4KB页大小）



- 逻辑地
  - ***Pag***
  - ***PageOffs*** ***Addr*** **MOD** ***PageLength***
  - 举例：对于4KB页面，若给定逻辑地址第5292字节，则 ***PageNo*** **= 1，** ***PageOffset*** **= 1196**

方法二：由页面大小为4KB可以推算出页内偏移地址占低12位；而逻辑地址5292即0x14AC。所以可知页号为1，页内偏移地址为0x04AC

- 地址变换任务
  - 关键在于页号到物理块号的转变

4.3 基本分页存储管理方式

4.3.1 离散存储管理方式

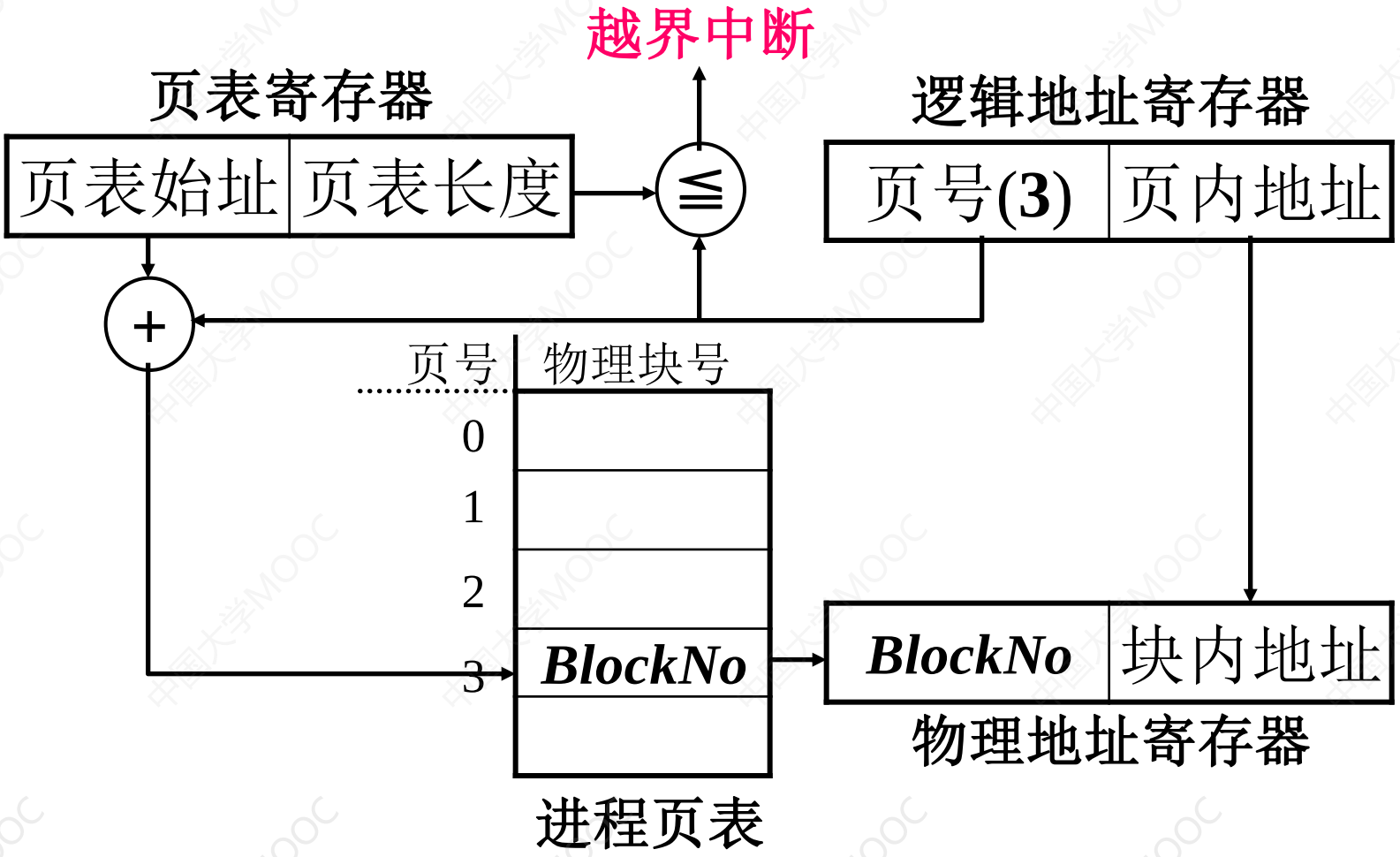
4.3.2 页面与页表

4.3.3 地址变换机构

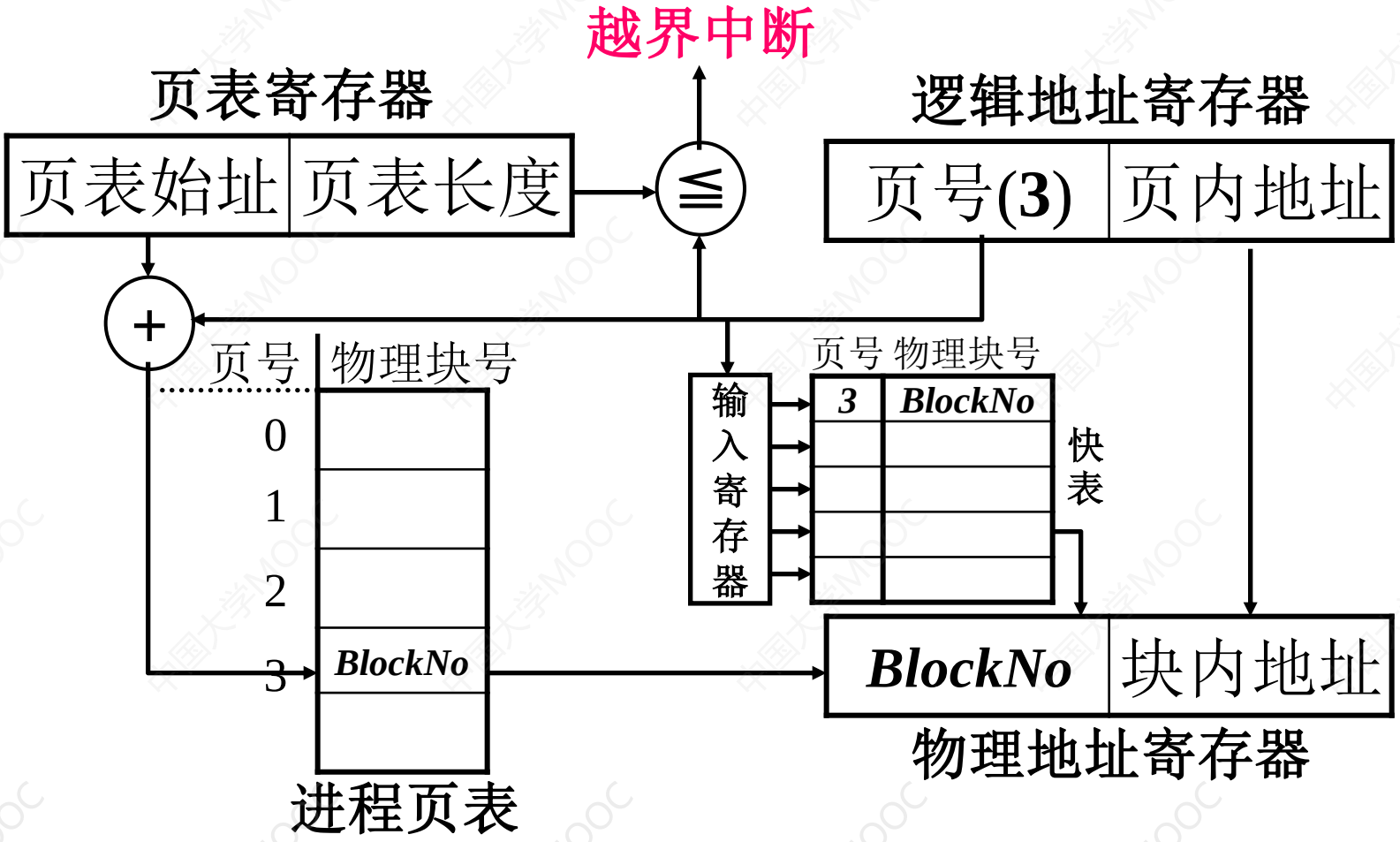
4.3.4 两级和多级页表

4.3.5 反置页表

基本的地址变换机构


越界中断
页表寄存器
页表始址
页表长度
≦
逻辑地址寄存器
页号(3)
页内地址
+
页号
物理块号
0
1
2
3
BlockNo
BlockNo
块内地址
物理地址寄存器
进程页表

具有快表的地址变换机构



快表引入前后数据存取速度比较

- 假定快表检索时间为20ns，内存访问时间为100ns，

4.3 基本分页存储管理方式

4.3.1 离散存储管理方式

4.3.2 页面与页表

4.3.3 地址变换机构

4.3.4 两级和多级页表

4.3.5 反置页表

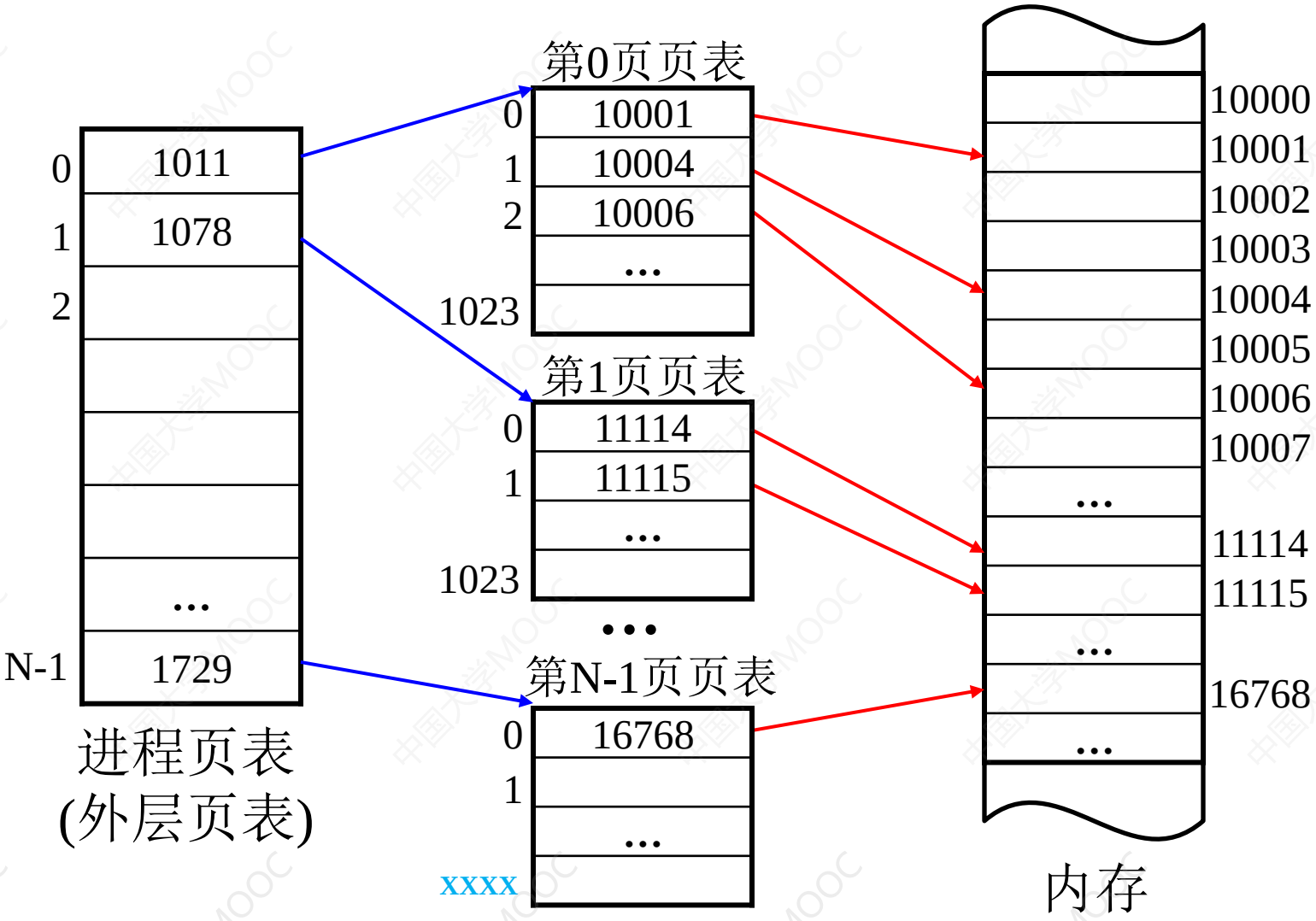
页表空间问题及对策

- 页表空间问题
  - 现代计算机系统支持非常大的逻辑地址空间(2^{32}~2^{64})，页表变得庞大。例如，对于具有32位逻辑地址空间的分页系统，规定页面大小为4KB即2^{12}B，则每个进程页表的页表项可达？ 个；若同时设定页表项大小为4B，则每个进程仅页表便需占用 ？ 内存空间，且要求是连续的
- 解决方案
  - 多级页表(页表分页及对换)或反置页表

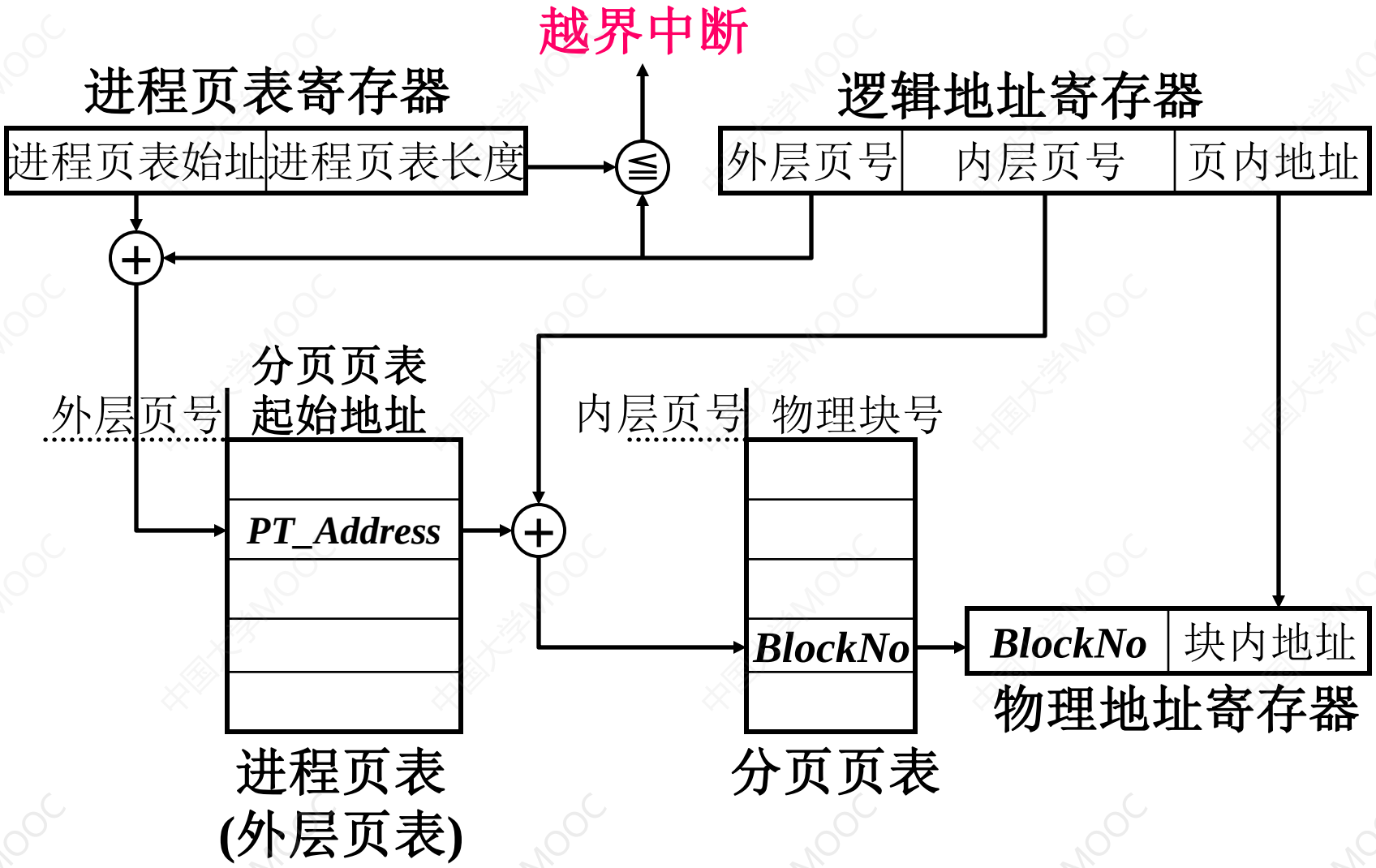
两级页表结构基本方法

- 基本思想
  - 对**页表**按内存物理块大小进行分页，对它们进行编号并离散地存放于不同的物理块中；同时为离散分配的**页表分页**再建立一张页表，称之为外层页表，以记录各页表分页对应的物理块号
  - 以前述32位逻辑地址空间、页面大小为4KB的系统为例，若页表项大小为4B，则每个进程页表的页表项可达**1M**个；若采用两级页表结构，由于各页表分页包含 ？ 个页表项，故需？个页表分页即可，因此外层页表的外层页号及内层页号 ？ 位。此时逻辑地址结构为：

两级页表结构示意图


0
1
2
N-1
1011
1078
...
1729
进程页表
(外层页表)
第0页页表
0
1
2
1023
10001
10004
10006
...
第1页页表
0
1
1023
11114
11115
...
第N-1页页表
0
1
xxxx
16768
...
10000
10001
10002
10003
10004
10005
10006
10007
...
11114
11115
...
16768
...
内存

具有两级页表的地址变换结构



多级页表结构

- 对于64位计算机，如规定页面大小仍为4KB，

物理块大小由4KB扩大到4MB，于是64位系统的进程逻辑地址空间可划分为 ？ 个页面，进程页表拥有？个页表项；页表项大小保持不变仍为4B，故每个物理块可含 ？ 个页表项；因此采用两级页表机制的话，外层页表应含 ？ 个页表项，需占用 ？ 的连续内存空间。

论[illegible]其结果都是不能接受的。因此，必须[illegible]采用多级页表，将**16GB的外层页表再进行分页**，并将各个分页离散的分配到不相邻接的物理块中，在利用第二级的外层页表来映射它们之间的关系。事实上，**对于64位的机器，用三级页表结构都很难适应**

4.3 基本分页存储管理方式

4.3.1 离散存储管理方式

4.3.2 页面与页表

4.3.3 地址变换机构

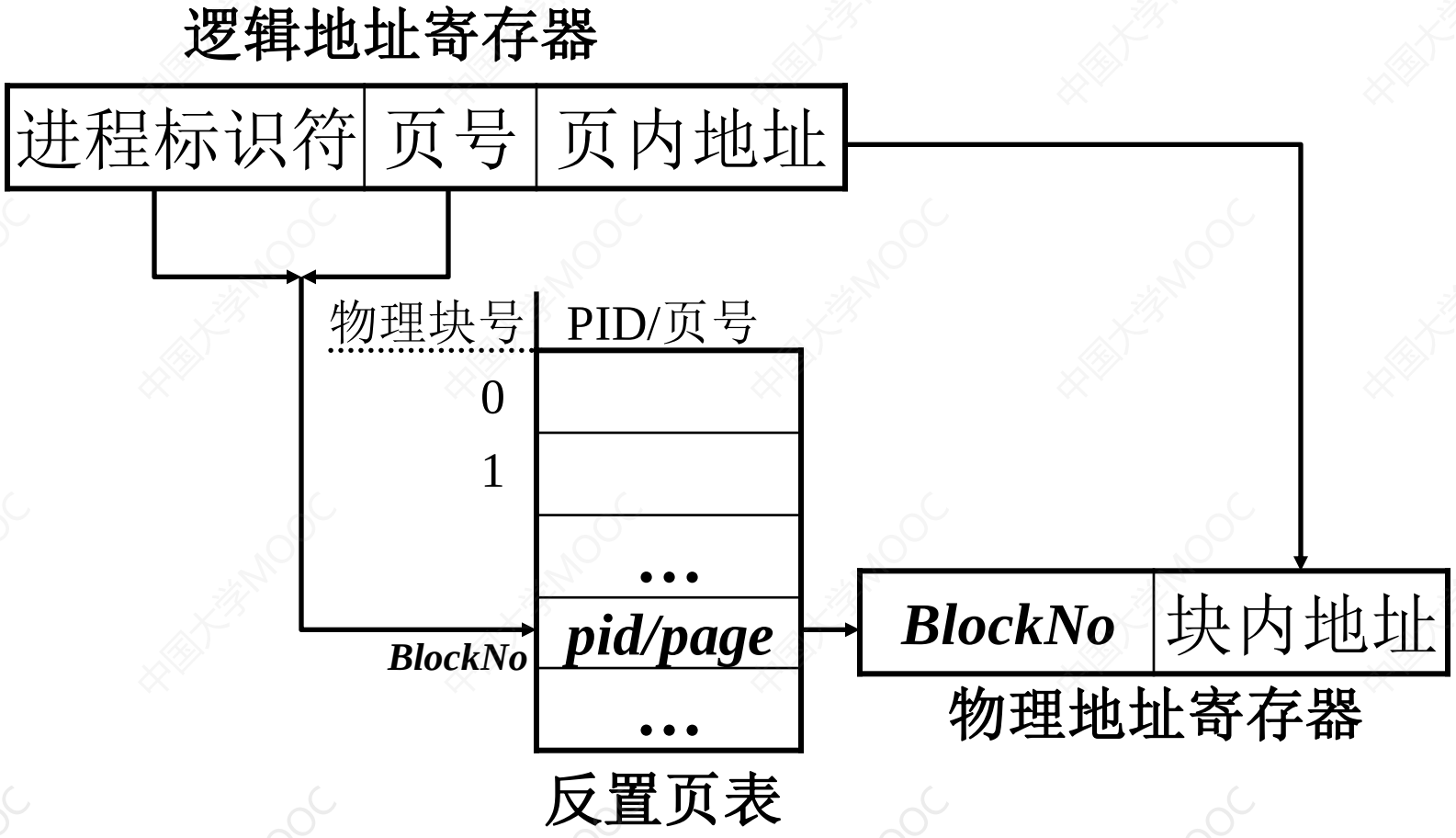
4.3.4 两级和多级页表

4.3.5 反置页表

反置页表

- 一般页表的表项是按页号进行排序，而页表项内容包括物理块号的；反置页表则是为每一个物理块设置一个页表项并将它们按物理块的号数排序，页表项内容包括对应页号及其隶属进程的标识符
- 在利用反置页表进行地址变换时，是用进程标识符和页号去检索反置页表。若找到与之匹配的表项，则以该表项序号即该页所在物理块号与页内地址构成物理地址；否则地址出错或产生调页中断请求
- 进程外部页表的设立及反置页表Hash检索

反置页表地址变换机构


逻辑地址寄存器
进程标识符
页号
页内地址
物理块号
PID/页号
0
1
…
pid/page
BlockNo
…
BlockNo
块内地址
物理地址寄存器
反置页表

4.3 基本分页存储管理方式

4.3.1 离散存储管理方式

4.3.2 页面与页表

4.3.3 地址变换机构

4.3.4 两级和多级页表

4.3.5 反置页表

作业题

- **4.4** 从内存管理的各个方面谈谈你对基本分页存储管理方式的认识与理解，包括相关数据结构和算法以及页面大小选择、多级页表和反置页表等。

第四章 内存管理

4.1 内存管理概述

4.2 连续分配存储管理方式

4.3 基本分页存储管理方式

4.4 基本分段存储管理方式

4.5 虚拟存储器概念及关键技术

4.6 请求分页存储管理方式

4.7 请求分段存储管理方式

4.4 基本分段存储管理方式

4.4.1 分段存储管理方式的引入

4.4.2 分段系统的基本原理

4.4.3 信息共享

4.4.4 段页式存储管理方式

分段存储管理方式的引入

- 满足用户在编程和使用上的多方面要求
  - 方便编程（作业基于逻辑关系自然分段）

    LOAD 1, [A]|<D>;

    STORE 1, [B]|<C>;
  - 信息共享
  - 信息保护
  - 动态链接

    （分段作为信息的逻辑单位）
  - 动态增长（特别是数据段地动态增长）

4.4 基本分段存储管理方式

4.4.1 分段存储管理方式的引入

4.4.2 分段系统的基本原理

4.4.3 信息共享

4.4.4 段页式存储管理方式

分段

- 作业地
  组逻辑
  据段D
- 每个段
  空间，
  度，因
- 整个作
  号(名)

31 ←

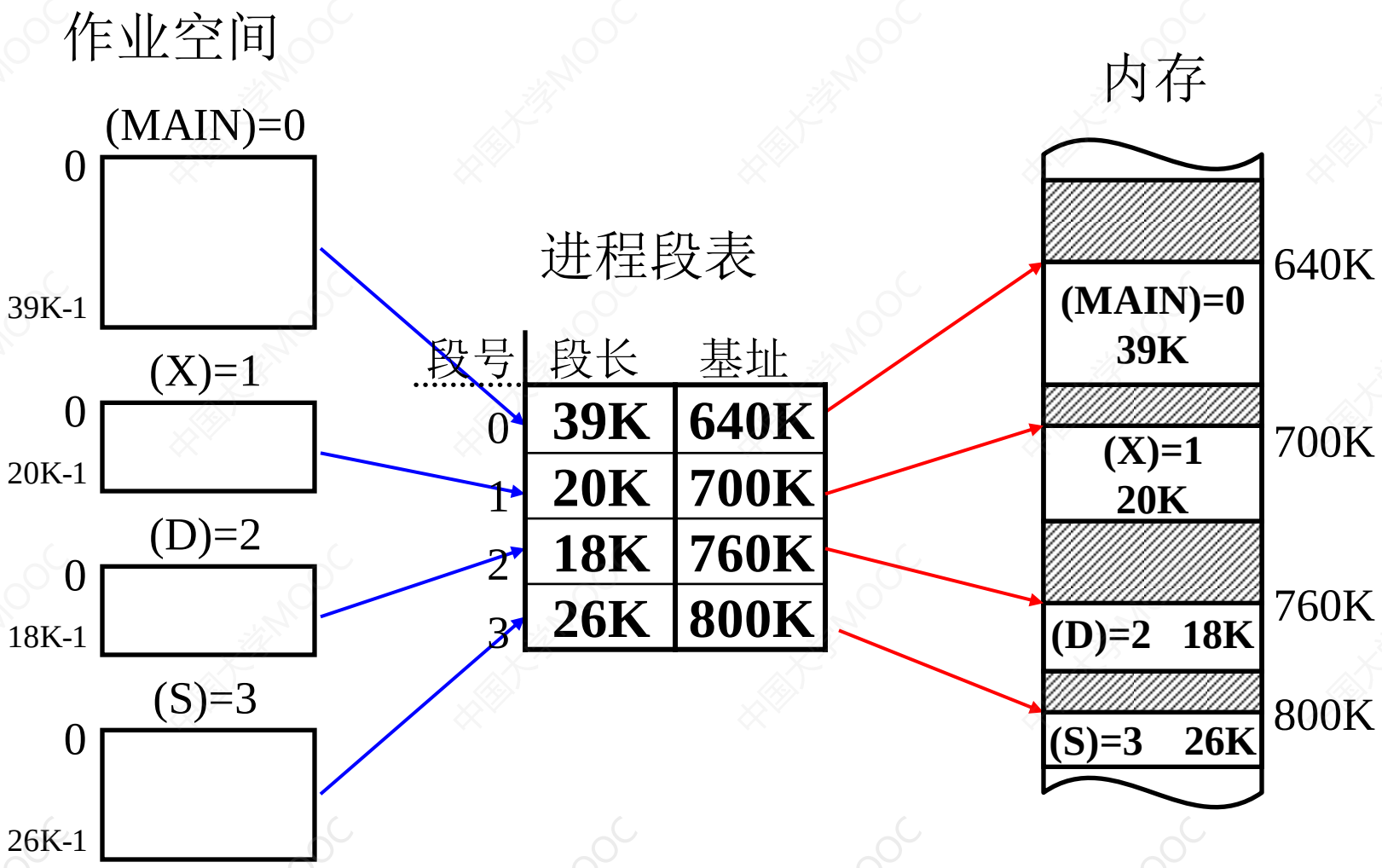
段

对于1KB物理块的分页：

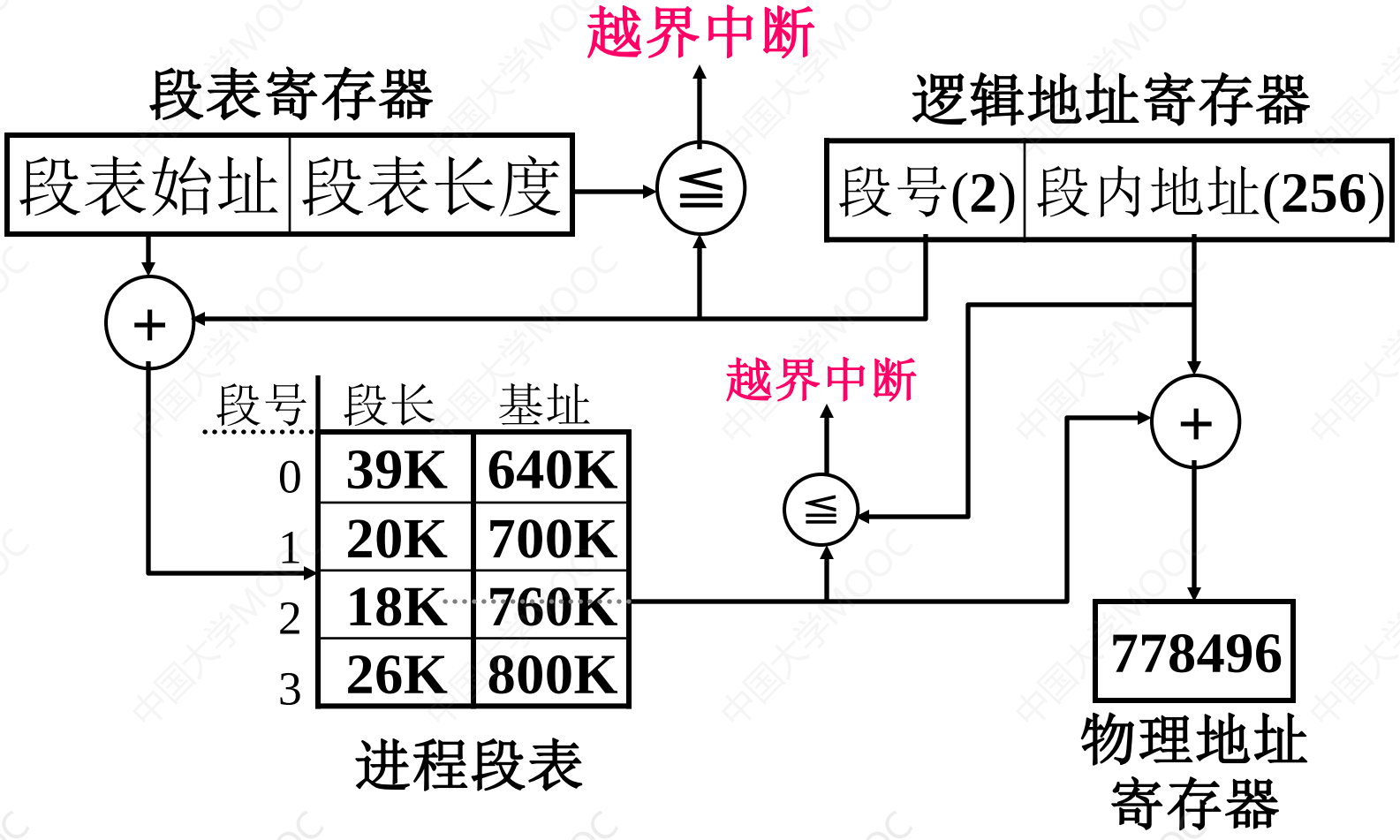
==========

页号	页内地址[十进制]		逻辑地址[二进制]		
0	0000	⇔	0000	0000	0000
0	0001	⇔	0000	0000	0001
0	0002	⇔	0000	0000	0010
	……				
0	1022	⇔	0011	1111	1110
0	1023	⇔	0011	1111	1111
1	0000	⇔	0100	0000	0000
1	0001	⇔	0100	0000	0001
1	0002	⇔	0100	0000	0010
	……				
1	1023	⇔	0111	1111	1111
2	0000	⇔	1000	0000	0000
2	0001	⇔	1000	0000	0001
	……				

利用段表实现地址映射示意图



分段系统地址变换机构


越界中断
段表寄存器
段表始址
段表长度
≦
逻辑地址寄存器
段号(2)
段内地址(256)
+
越界中断
段号
段长
基址
0
39K
640K
1
20K
700K
2
18K
760K
3
26K
800K
进程段表
≦
+
778496
物理地址
寄存器

分页与分段存储管理的比较

- 相似之处
  - 实现机制（离散分配、地址映射机构）
- 目的及内涵
  - 系统管理需要/用户需要、物理/逻辑单位
- 分页/段长度
  - 固定与否、决定因素(系统硬件/信息性质)
- 作业地址空间维数
  - 一维/二维、程序员编程

4.4 基本分段存储管理方式

4.4.1 分段存储管理方式的引入

4.4.2 分段系统的基本原理

4.4.3 信息共享

4.4.4 段页式存储管理方式

可重入代码（纯代码）

- 一种允许多个进程同时访问的代码
  - 为使各个进程所执行的代码完全相同，绝对不允许可重入代码在执行中有任何改变，所以它是一种不允许任何进程对其进行修改的代码
  - 但事实上，大多数代码在执行时都有可能发生改变，例如其中用于控制程序执行次数的变量及指针、信号量及数组等。为此，在每个进程中都必须配备局部数据区，并把在执行中可能改变的部分都拷贝到该数据区。这样，在程序执行时，只去对属于特定进程私有的数据区中的内容进行修改，而不去改变共享的代码，这时的可共享代码即成为可重入代码

信息共享比较举例说明

- 多个用户对文本编辑程序的共享
  - 某多用户系统，可同时接纳40个用户，假设均在执行Editor进行文本编辑。若该文本编辑程序含有160KB的代码区和40KB的数据区，则总共需有 ？ 的内存空间来支持40个用户。如果该文本编辑程序代码是**可重入的**，则无论分页系统还是分段系统该程序代码都能被共享，即内存中只需保留一份文本编辑程序的副本，因而所需内存空间仅为 ？

基于分页的文本编辑器共享

基于分段的文本编辑器共享

4.4 基本分段存储管理方式

4.4.1 分段存储管理方式的引入

4.4.2 分段系统的基本原理

4.4.3 信息共享

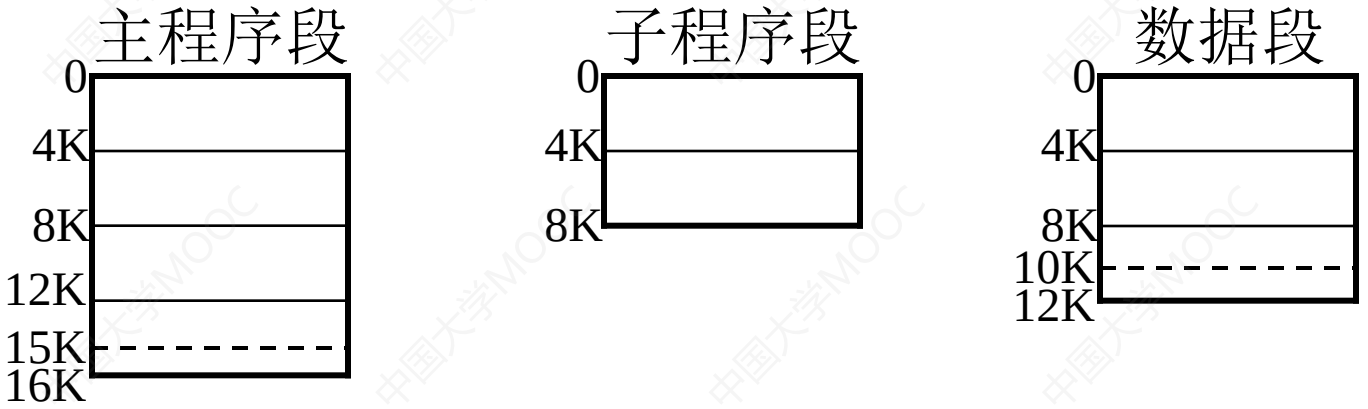
4.4.4 段页式存储管理方式

段页式存储管理方式的引入

- 分页与分段存储管理各有优缺点
  - 分页系统能有效地提高内存利用率
  - 分段系统则能很好地满足用户需要
- 分页/段存储管理各取所长、有机结合
  - 既具有分段系统便于实现、分段可共享、易于保护、可动态链接等一系列优点；又能像分页系统那样很好地解决内存的外部碎片问题以及为各个分段离散地分配内存等问题

段页式存储管理方法

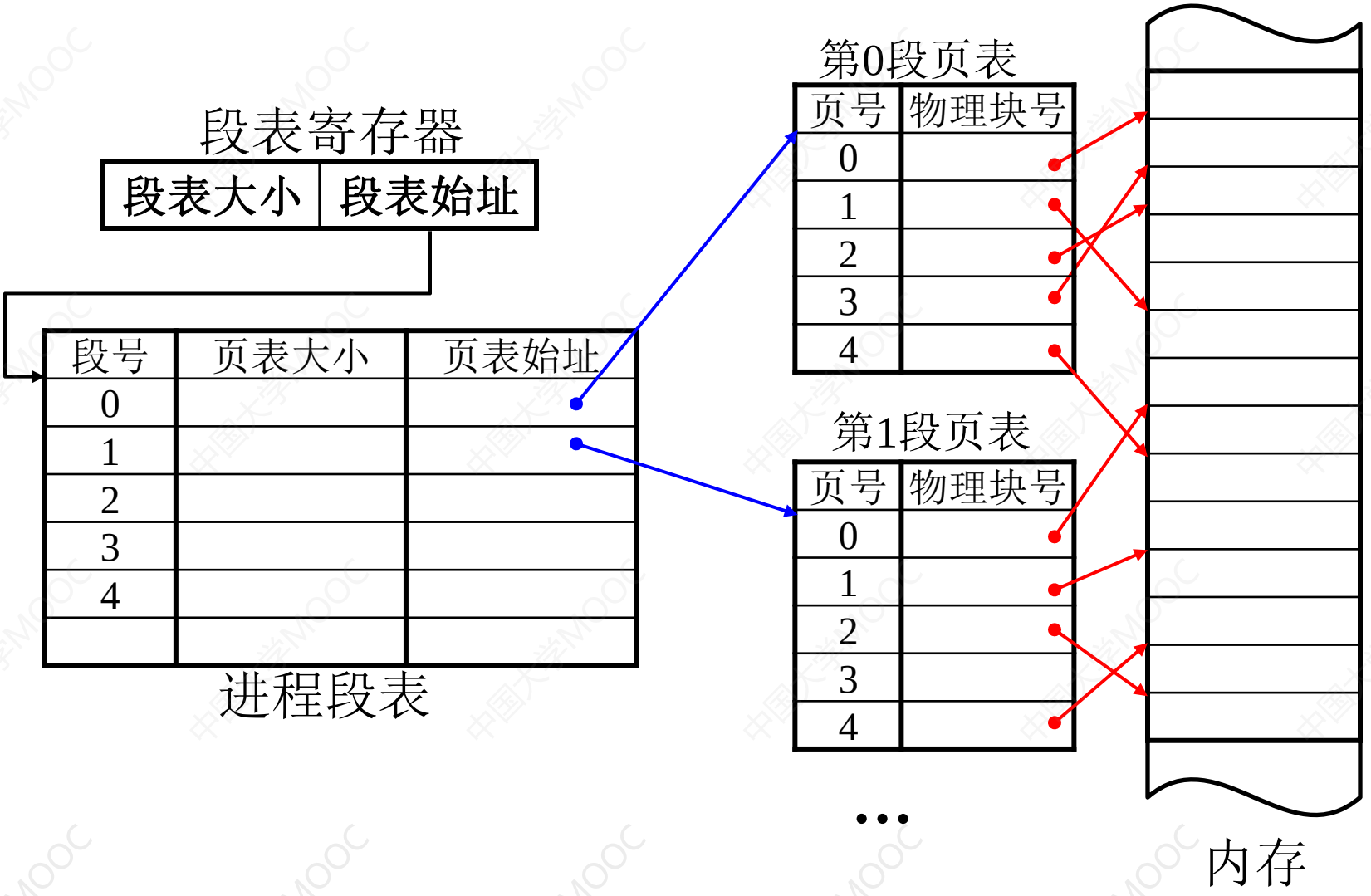
- 分段与分页原理的结合
  - 先将用户程序按信息性质分为若干段（赋予一个段名），再把每个段划分为若干页



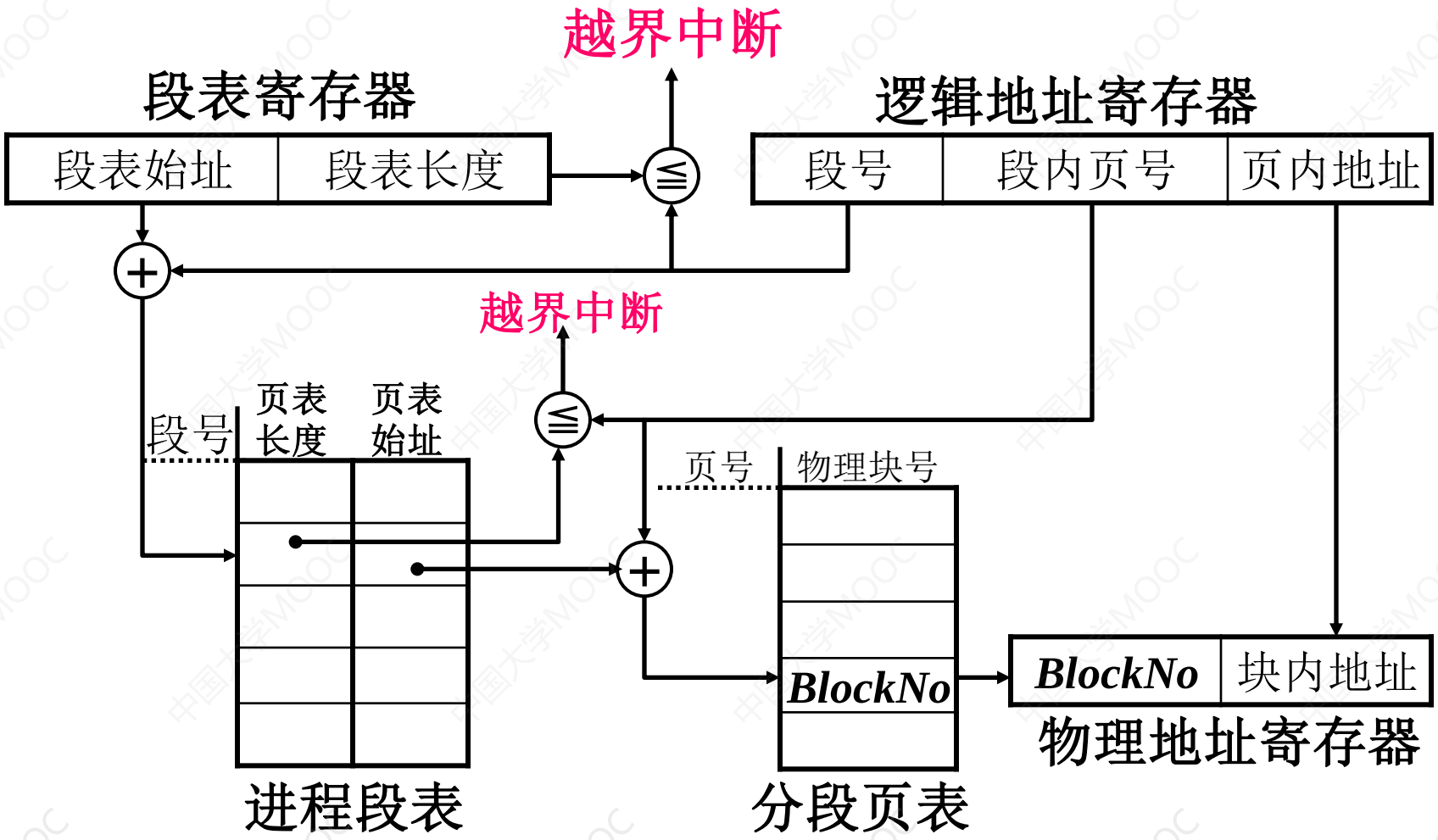
段页式存储管理地址结构

段号	段内偏移地址

利用段表和页表实现地址映射



段页式系统的地址变换结构



4.4 基本分段存储管理方式

4.4.1 分段存储管理方式的引入

4.4.2 分段系统的基本原理

4.4.3 信息共享

4.4.4 段页式存储管理方式

作业题

- **4.5** 从内存管理的各个方面谈谈你对基本分段存储管理方式的认识与理解，包括相关数据结构和算法等。
- **4.6** 比较分页存储管理与分段存储管理的异同。
- **4.7** 简要阐述段页式存储管理方式的基本原理。

第四章 内存管理

4.1 内存管理概述

4.2 连续分配存储管理方式

4.3 基本分页存储管理方式

4.4 基本分段存储管理方式

4.5 虚拟存储器概念及关键技术

4.6 请求分页存储管理方式

4.7 请求分段存储管理方式

常规存储管理问题与对策

- 要求将一个作业全部装入内存方能运行
  - 一些对内存空间要求超过内存容量的大作业因不能全部装入内存而无法运行
  - 同时有大量作业要求运行，但由于内存容量不足以容纳所有这些作业，则只能将少数作业装入内存首先运行，而将其它大量作业留在外存上等待
- 解决方法
  - 增加物理内存容量（系统成本和机器条件）
  - 从逻辑上扩充内存容量=>虚拟存储技术

一次性全部装入及驻留性问题

- 作业“一次性”全部装入内存并不必要
  - 许多作业在每次运行时并非用到其全部程序和数据
- 作业常驻内存存在不合理性
  - 因输入输出操作尚未完成而处于长期等待状态的运行进程或某些一次性运行程序对宝贵内存资源的占据
- 问题后果
  - 使一些需要运行的作业无法装入运行，从而严重降低内存利用率和减少系统吞吐量

局部性原理

- 程序在执行时将呈现局部性规律，即在一较短时间内，程序的执行仅限于某个部分；相应地，它所访问的内存空间也仅局限于某个区域
  - 程序在大多数情况下的顺序执行特点
  - 过程调用深度及执行轨迹
  - 程序循环结构执行及数据结构操作特点
- 局部性表现形式
  - 时间局部性（指令执行与数据结构访问）
  - 空间局部性（存储单元临近访问）

虚拟存储器技术要点

- 作业部分装入内存即可启动运行
  - 其余部分暂留磁盘
- 程序执行过程页段访问机制
  - 已调入内存则直接访问
  - 尚未调入内存则缺页/段中断及请求调入
  - 页段置换功能
- 技术效果
  - 大用户程序在小内存空间的运行
  - 多道程序度的提高

虚拟存储器的定义

- 所谓虚拟存储器，是指仅把作业的一部分装入内存便可运行作业的存储器系统。具体地说，所谓虚拟存储器是指具有请求调入功能和置换功能，能从逻辑上对内存容量进行扩充的一种存储器系统。实际上，用户看到的大容量只是一种感觉，是虚的，故而得名虚拟存储器。其逻辑容量由内存和外存容量之和所决定，其运行速度接近于内存速度，而每位的成本却又接近于外存。

请求分页虚拟存储系统

- 技术构成
  - 分页 + 请求调页 + 页面置换
- 硬件支持
  - 请求分页的页表机制
  - 缺页中断机构
  - 地址变换机构
- 软件支持
  - 请求调页
  - 页面置换

请求分段虚拟存储系统

- 技术构成
  - 分段 + 请求调段 + 分段置换
- 硬件支持
  - 请求分段的段表机制
  - 缺段中断机构
  - 地址变换机构
- 软件支持
  - 请求调段
  - 分段置换

虚拟存储器的特征

- 离散性
  - 离散分配方式
- 多次性
  - 作业被分成多次调入内存和运行
- 对换性
  - 程序和数据在作业运行过程中的换进/出
- 虚拟性
  - 内存容量的逻辑扩充

作业题

- **4.8** 谈谈你对虚拟存储器的概念、特征及其软硬件技术支持的认识与理解。

第四章 内存管理

4.1 内存管理概述

4.2 连续分配存储管理方式

4.3 基本分页存储管理方式

4.4 基本分段存储管理方式

4.5 虚拟存储器概念及关键技术

4.6 请求分页存储管理方式

4.7 请求分段存储管理方式

4.6 请求分页存储管理方式

4.6.1 请求分页中的硬件支持

4.6.2 内存分配策略和分配算法

4.6.3 调页策略

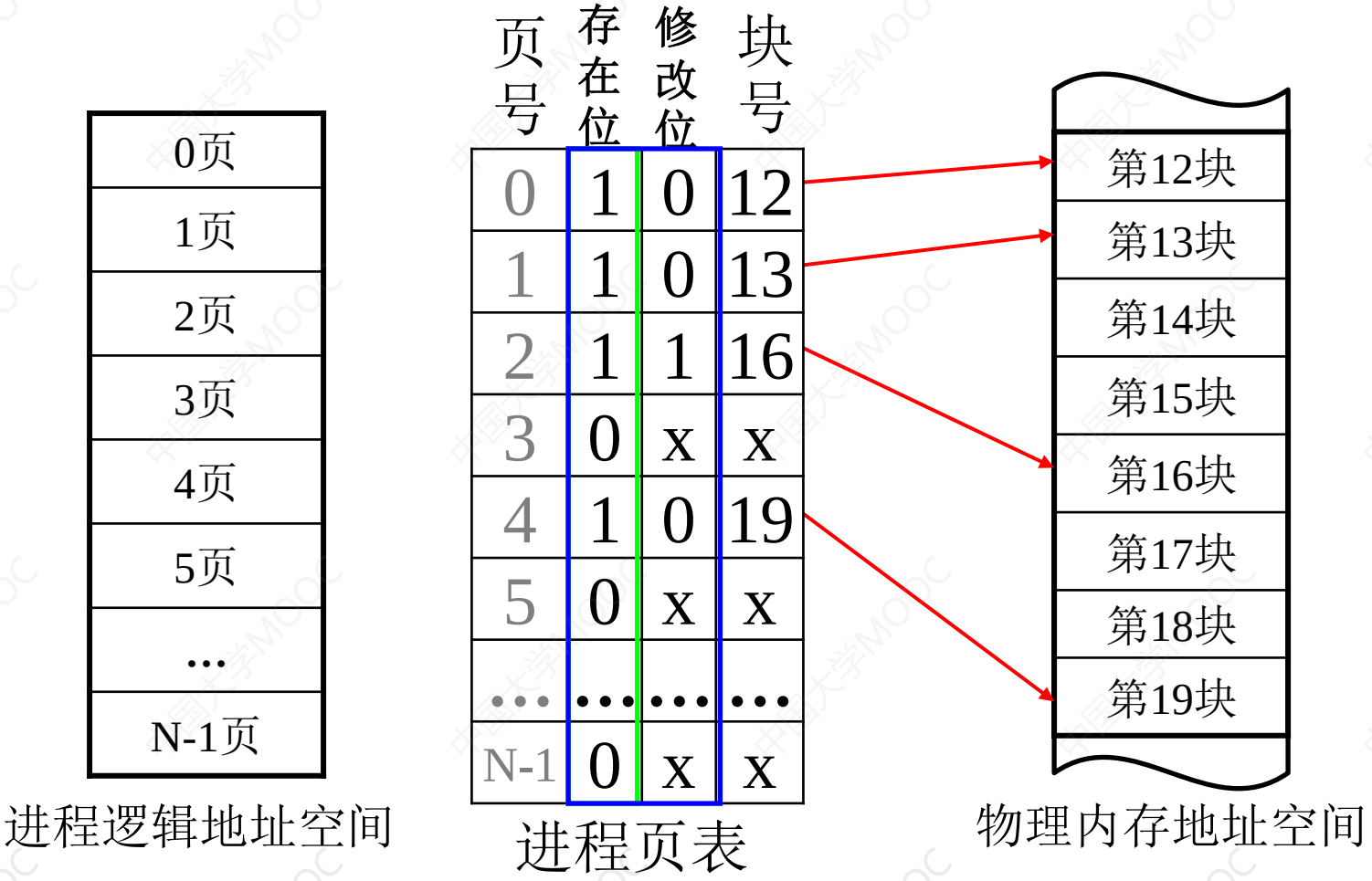
4.6.4 页面淘汰算法

页表机制

- 页表项的扩充

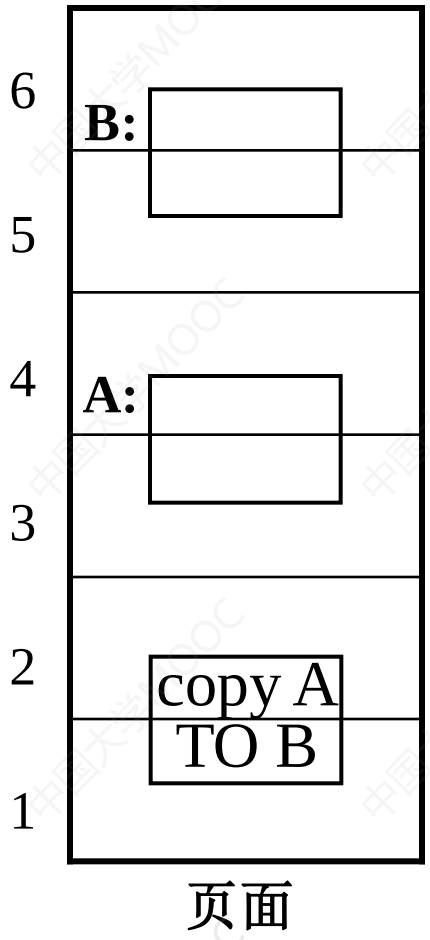
页号	物理块号

请求分页虚拟内存管理之页表举例



缺页中断机构

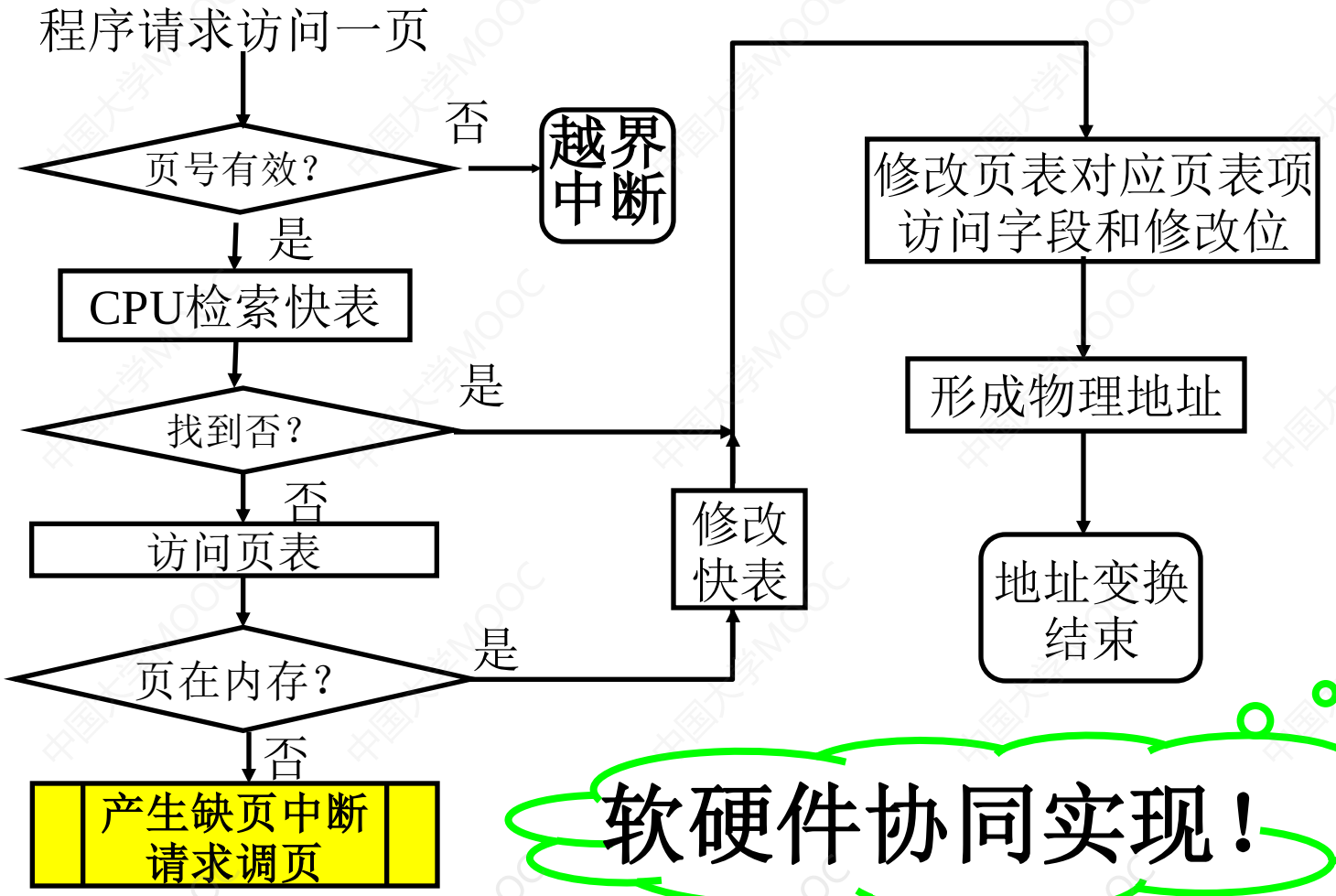
- 缺页中断之中断特征
  - 保护CPU现场
  - 分析中断原因
  - 转入缺页中断处理程序
  - 恢复CPU现场
- 缺页中断之特殊性
  - 在指令执行期间产生和处理中断信号
  - **一条指令执行期间，可能产生多次缺页中断**



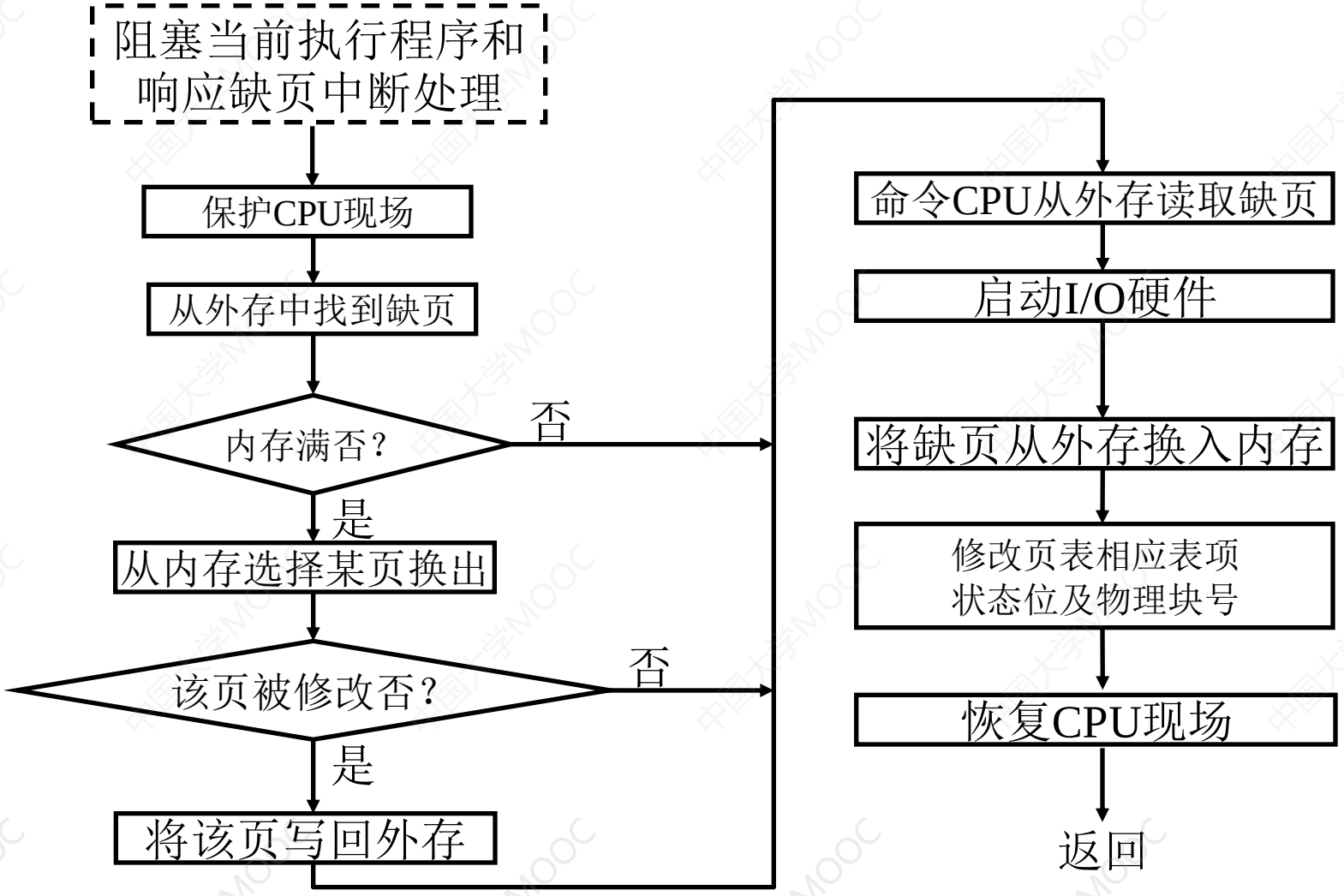
地址变换机构

- 在分页系统的地址变换机构的基础上，增加缺页中断产生和处理并页面置换功能而构成
- 地址变换过程要领
  - 从页表找到对应分页的页表项获悉该页尚未调入内存时，应产生缺页中断，请求操作系统从外存把该页调入内存
  - 关于快表和页表的检索及表项修改

请求分页系统地址变换过程



缺页中断处理算法流程


阻塞当前执行程序和
响应缺页中断处理
保护CPU现场
从外存中找到缺页
内存满否？
否
是
从内存选择某页换出
该页被修改否？
否
是
将该页写回外存
命令CPU从外存读取缺页
启动I/O硬件
将缺页从外存换入内存
修改页表相应表项
状态位及物理块号
恢复CPU现场
返回

4.6 请求分页存储管理方式

4.6.1 请求分页中的硬件支持

4.6.2 内存分配策略和分配算法

4.6.3 调页策略

4.6.4 页面淘汰算法

最小物理块数的确定

- 保证进程正常运行所需的最少物理块数
  - 若系统为某进程分配的物理块数少于此值，进程将无法正常运行
  - 不同于使进程有效工作所需的物理块数
- 与计算机的硬件结构有关，并取决于指令的格式（操作数个数）、功能和寻址方式（直接/间接）

物理块分配与置换策略

- 固定分配局部置换
  - 为每个进程分配一固定页数的内存空间，在整个运行期间都不再改变
- 可变分配全局置换
  - 系统设立一个空闲物理块队列，缺页即触发全局新增分配或全局置换
- 可变分配局部置换
  - 依据**缺页率**酌情增加或减少物理块

物理块分配算法

- 平均分配算法
  - 将系统可供分配的物理块平均分配
- 按比例分配算法
  - $\mathbf{BlockOfP_k = max\{minBlocks, Blocks \times PagesOfP_k/\Sigma PagesOfP_i\}}$
- 考虑优先权的分配算法
  - 照顾重要或紧迫的作业能尽快完成

4.6 请求分页存储管理方式

4.6.1 请求分页中的硬件支持

4.6.2 内存分配策略和分配算法

4.6.3 调页策略

4.6.4 页面淘汰算法

何时调入页面

- 预调页策略
  - 将那些预计在不久之后便会被访问的程序或数据所在的页面，预先调入内存
  - 以预测为基础，主要用于进程首次调入
- 请求调页策略
  - 当进程在运行中需要访问某部分程序和数据时，若发现其所在的页面不在内存，应立即提出请求，由系统将其所需页面调入内存；易于实现但系统开销大

何处调入页面

- 对换区空间充分
  - 进程运行前，便须将与该进程有关的文件，从文件区拷贝到对换区
- 对换区空间不足
  - 文件是否修改分别处理
- UNIX方式
  - 凡未运行过的页面都应从文件区调入，而对于曾运行过又被换出到对换区的页面则从对换区调入

页面调入过程

- 缺页中断发生
  - 程序所访问的页面不在内存时产生缺页中断并转入缺页中断处理程序
- 根据页表项给定外存地址调入所缺页面
  - 页表项外存地址（物理盘块号）
- 内存不足置换
  - 页面淘汰算法
  - 是否重写磁盘

4.6 请求分页存储管理方式

4.6.1 请求分页中的硬件支持

4.6.2 内存分配策略和分配算法

4.6.3 调页策略

4.6.4 页面淘汰算法

4.6.4 页面淘汰算法

4.6.4.1 抖动与缺页率

4.6.4.2 最佳淘汰算法

4.6.4.3 先进先出淘汰算法

4.6.4.4 最长时间未使用淘汰算法

4.6.4.5 时钟式淘汰算法

4.6.4.6 最少使用淘汰算法

4.6.4.7 页面缓冲算法

抖动与缺页率

- 抖动的定义
  - 如果所用淘汰算法不当，便可能导致这样一种情形：刚被换出的页面很快又被访问，需重新调入，为此，又需再选一页换出；而此刚被换出的页面，不久也被访问，故又需将它调入，如此频繁地更换页面，以致一个进程在运行中把大部分的时间耗费在页面置换的工作上，称该进程发生了抖动（或称之为颠簸）
- 缺页率
  - 缺页率 = 缺页中断次数/页面访问次数

4.6.4 页面淘汰算法

4.6.4.1 抖动与缺页率

4.6.4.2 最佳淘汰算法

4.6.4.3 先进先出淘汰算法

4.6.4.4 最长时间未使用淘汰算法

4.6.4.5 时钟式淘汰算法

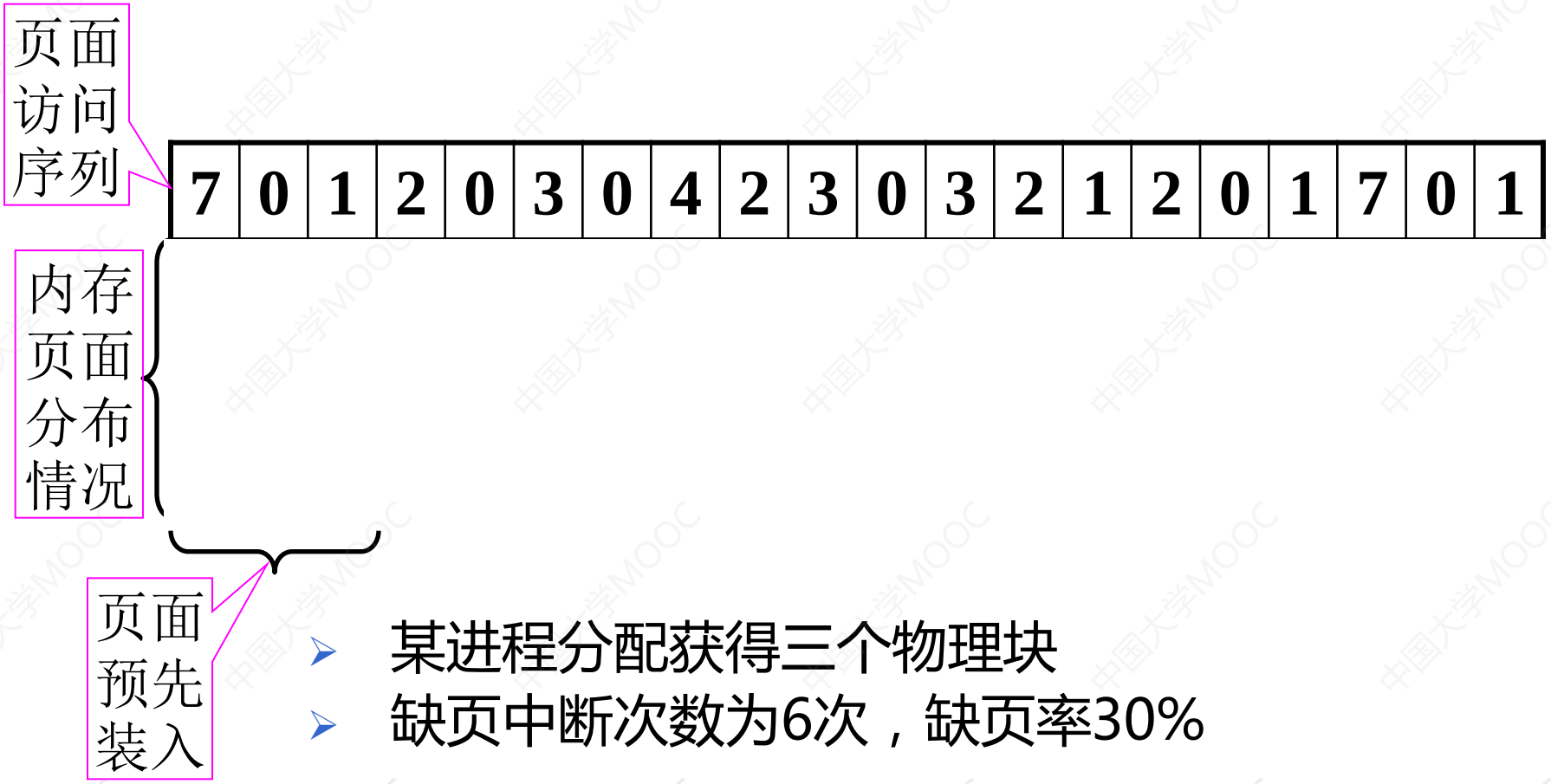
4.6.4.6 最少使用淘汰算法

4.6.4.7 页面缓冲算法

最佳淘汰算法

- 基本思想
  - 选择永不使用或是在最长时间内不再被访问（即距现在最长时间才会被访问）的页面淘汰出内存
- 评价
  - 理想化算法，具有最好性能（对于固定分配页面方式，本法可保证获得最低的缺页率），但实际上却难于实现，故主要用于算法评价参照

最佳淘汰算法举例说明



- 某进程分配获得三个物理块
- 缺页中断次数为6次，缺页率30%

4.6.4 页面淘汰算法

4.6.4.1 抖动与缺页率

4.6.4.2 最佳淘汰算法

4.6.4.3 先进先出淘汰算法

4.6.4.4 最长时间未使用淘汰算法

4.6.4.5 时钟式淘汰算法

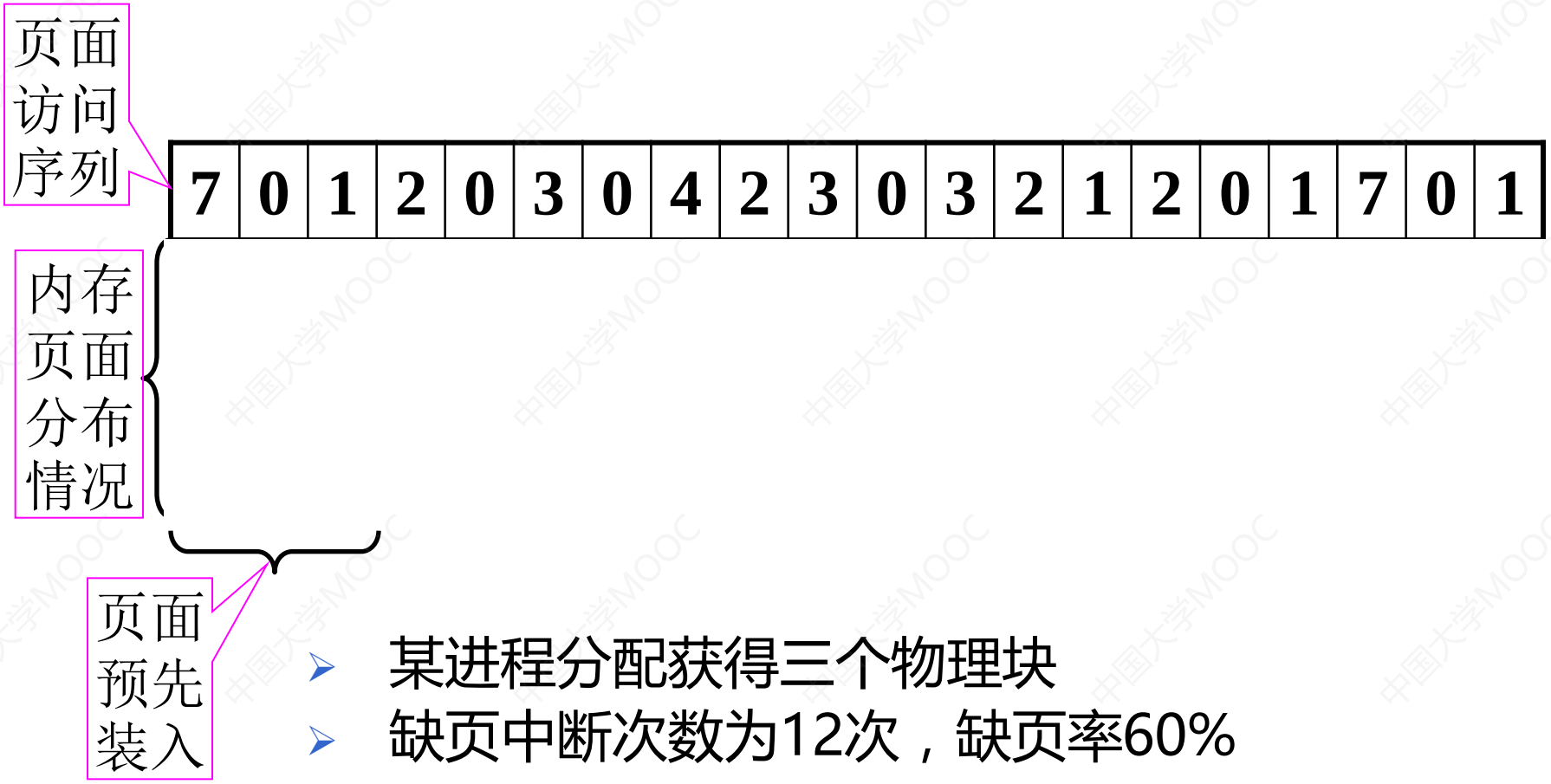
4.6.4.6 最少使用淘汰算法

4.6.4.7 页面缓冲算法

先进先出淘汰算法

- 基本思想
  - 选择最先进入内存即在内存驻留时间最久的页面换出到外存
  - 进程已调入内存的页面按进入先后次序链接成一个队列，并设置淘汰指针以指向最老页面
- 评价
  - 简单直观，但不符合进程实际运行规律，性能较差，故实际应用极少

先进先出淘汰算法举例说明



- 某进程分配获得三个物理块
- 缺页中断次数为12次，缺页率60%

4.6.4 页面淘汰算法

4.6.4.1 抖动与缺页率

4.6.4.2 最佳淘汰算法

4.6.4.3 先进先出淘汰算法

4.6.4.4 最长时间未使用淘汰算法

4.6.4.5 时钟式淘汰算法

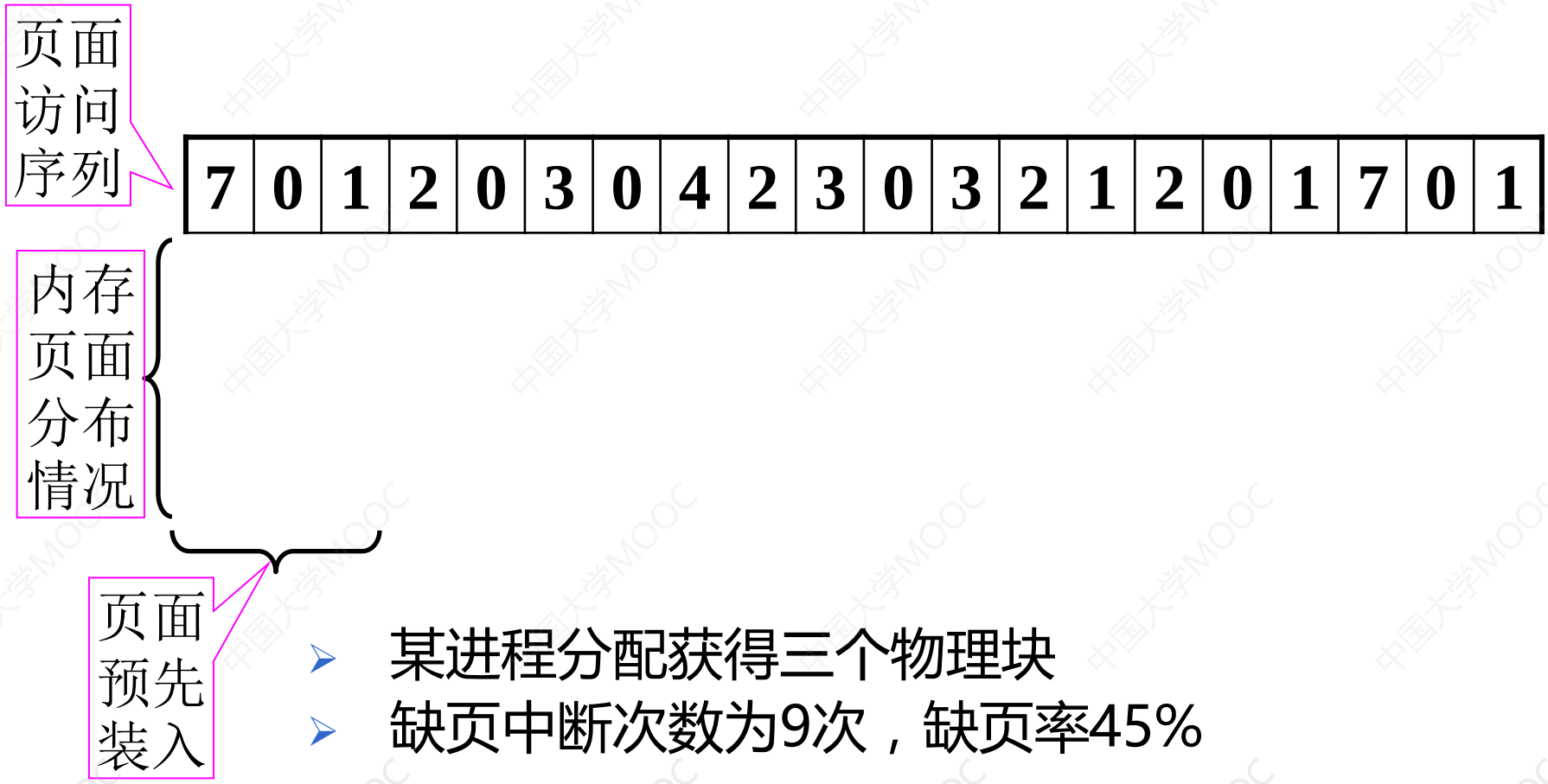
4.6.4.6 最少使用淘汰算法

4.6.4.7 页面缓冲算法

最长时间未使用淘汰算法LRU

- 基本思想
  - 以“最近的过去”作为“最近的将来”的近似，选择最近一段时间最长时间未被访问的页面淘汰出内存
- 评价
  - 适用于各种类型的程序，性能较好，但需要较多的硬件支持

最长时间未使用淘汰算法举例说明



- 某进程分配获得三个物理块
- 缺页中断次数为9次，缺页率45%

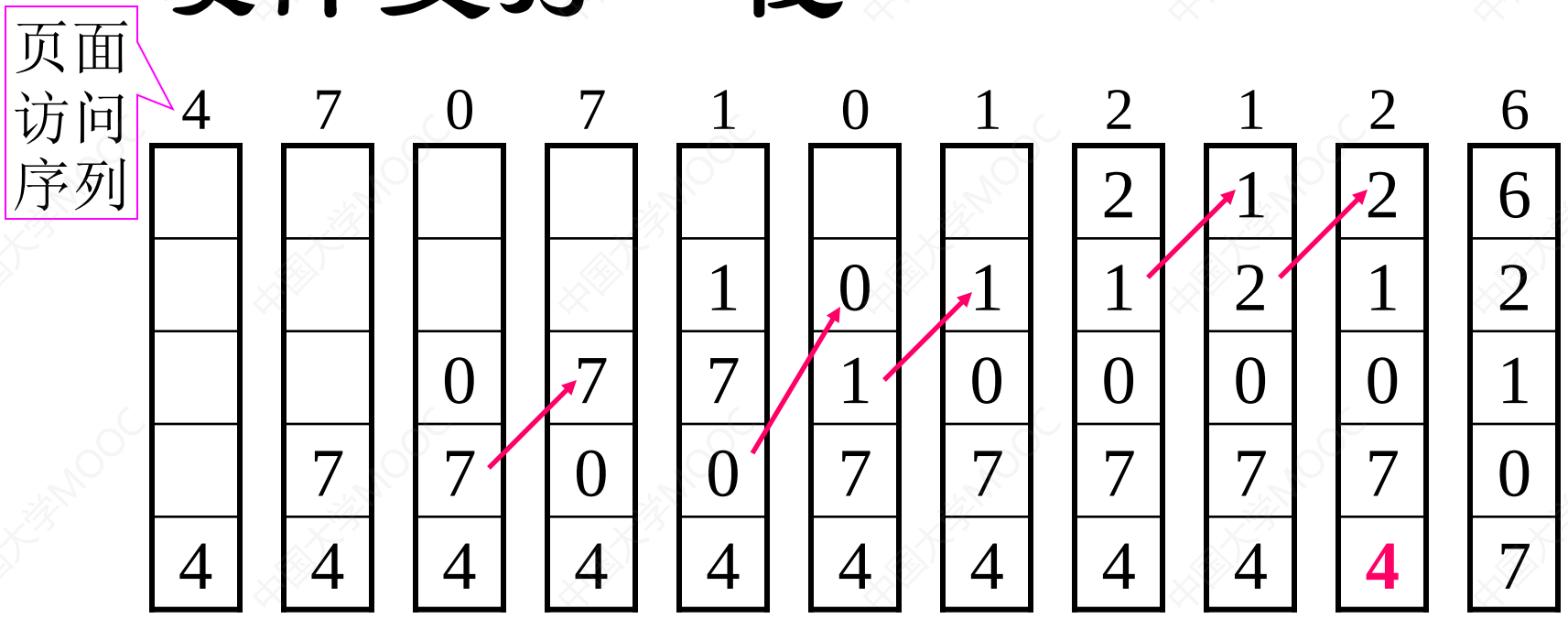
最长时间未使用淘汰算法

硬件支持——移位寄存器

B# \ R	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
1	0	1	0	1	0	0	1	0
2	1	0	1	0	1	1	0	0
3	0	0	0	0	1	1	0	0
4	0	1	1	0	1	0	1	1
5	1	1	0	1	0	1	1	0
6	0	0	1	0	1	0	1	1
7	0	0	0	0	0	1	1	1
8	0	1	1	0	1	1	0	1

最长时间未使用淘汰算法

硬件支持——栈



- 某进程分配获得五个物理块

4.6.4 页面淘汰算法

4.6.4.1 抖动与缺页率

4.6.4.2 最佳淘汰算法

4.6.4.3 先进先出淘汰算法

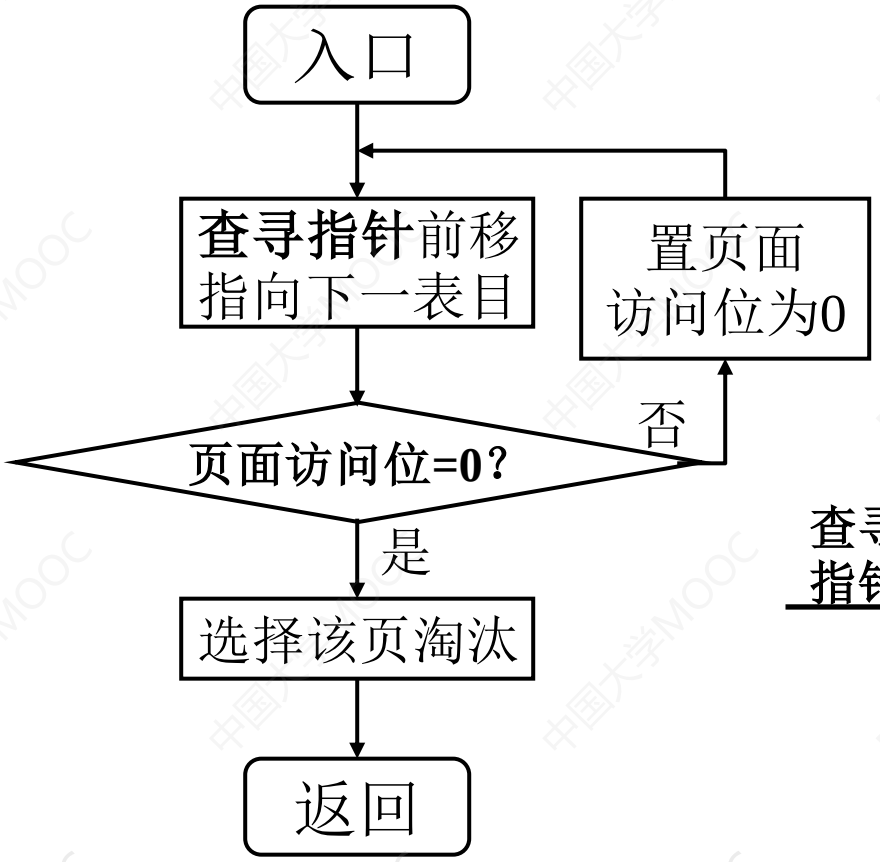
4.6.4.4 最长时间未使用淘汰算法

4.6.4.5 时钟式淘汰算法

4.6.4.6 最少使用淘汰算法

4.6.4.7 页面缓冲算法

简单时钟式淘汰算法 (Clock/NRU)



查寻指针

索引号	页号	页面访问位
0	12	1
1	10	1
2	4	0
3	6	1
4	2	1
5	3	0
…	…	0/1
W-1	1	1

改进型时钟式淘汰算法

- 基本思想

  ① 从查寻指针当前位置起扫描内存分页循环队列，选择A=0且M=0的第一个页面淘汰；若未找到，转②

  ② 开始第二轮扫描，选择A=0且M=1的第一个页面淘汰，同时将经过的所有页面访问位置0；若不能找到，转①

- 评价
  - 与简单时钟式淘汰算法相比，可减少磁盘的I/O操作次数，但淘汰页的选择可能经历多次扫描，故实现算法自身的开销增大

4.6.4 页面淘汰算法

4.6.4.1 抖动与缺页率

4.6.4.2 最佳淘汰算法

4.6.4.3 先进先出淘汰算法

4.6.4.4 最长时间未使用淘汰算法

4.6.4.5 时钟式淘汰算法

4.6.4.6 最少使用淘汰算法

4.6.4.7 页面缓冲算法

最少使用淘汰算法LFU

- 基本思想
  - 为内存各页设置一移位寄存器用于记录对应被访频率，并选择在最近时期使用次数最少的页面淘汰
- 评价
  - 鉴于仅用移位寄存器有限各位来记录页面使用会导致访问一次与访问多次的等效性，本算法并不能真实全面地反映页面使用情况

4.6.4 页面淘汰算法

4.6.4.1 抖动与缺页率

4.6.4.2 最佳淘汰算法

4.6.4.3 先进先出淘汰算法

4.6.4.4 最长时间未使用淘汰算法

4.6.4.5 时钟式淘汰算法

4.6.4.6 最少使用淘汰算法

4.6.4.7 页面缓冲算法

页面缓冲算法PBA

- 基本思想
  - 设立空闲页面链表和已修改页面链表
  - 采用可变分配和基于先进先出的局部置换策略，并规定被淘汰页先不做物理移动，而是依据是否修改分别挂到**空闲页面链表**或**已修改页面链表**的末尾
  - 空闲页面链表同时用于物理块分配
  - 当已修改页面链表达到一定长度如64个页面时，一起将所有已修改页面写回磁盘，故可显著减少磁盘I/O操作次数

4.6.4 页面淘汰算法

4.6.4.1 抖动与缺页率

4.6.4.2 最佳淘汰算法

4.6.4.3 先进先出淘汰算法

4.6.4.4 最长时间未使用淘汰算法

4.6.4.5 时钟式淘汰算法

4.6.4.6 最少使用淘汰算法

4.6.4.7 页面缓冲算法

4.6 请求分页存储管理方式

4.6.1 请求分页中的硬件支持

4.6.2 内存分配策略和分配算法

4.6.3 调页策略

4.6.4 页面淘汰算法

作业题

- **4.9** 试从内存分配、页表机制、地址变换、缺页中断以及调页过程等方面谈谈你对请求分页存储管理系统的认识与理解。
- **4.10** 什么是抖动？如何计算缺页率？阐述和比较各种页面淘汰算法的基本流程和实现要领。

作业题

- **4.11[必做]** 某虚拟存储器的用户空间共有32个页面，每页1KB，主存16KB。假定某时刻系统为用户的第0、1、2、3页分别分配的物理块号为5、10、4、7，试将虚拟地址0A5C和093C变换为物理地址。
- **4.12[必做]**某请求分页系统采用LRU页面淘汰算法，假定一个作业的页面走向为4、3、2、1、4、3、5、4、3、2、1、5，当分配给该作业的物理块数M分别为3和4时，试计算访问过程中所发生的缺页次数和缺页率，并比较所得结果。

实验课题

实验课题13

页面淘汰算法模拟实现与比较

第四章 内存管理

4.1 内存管理概述
4.2 连续分配存储管理方式
4.3 基本分页存储管理方式
4.4 基本分段存储管理方式
4.5 虚拟存储器概念及关键技术
4.6 请求分页存储管理方式
4.7 请求分段存储管理方式

第四章 内存管理

4.1 内存管理概述
4.2 连续分配存储管理方式
4.3 基本分页存储管理方式
4.4 基本分段存储管理方式
4.5 虚拟存储器概念及关键技术
4.6 请求分页存储管理方式
4.7 请求分段存储管理方式

4.7 请求分段存储管理方式

4.7.1 请求分段中的硬件支持

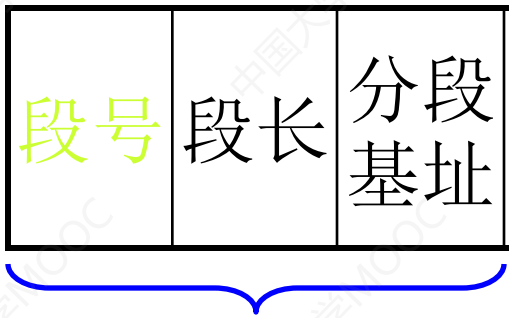
4.7.2 分段共享

4.7.3 分段保护

4.7.4 x86体系中请求段页式支撑机制

段表机制

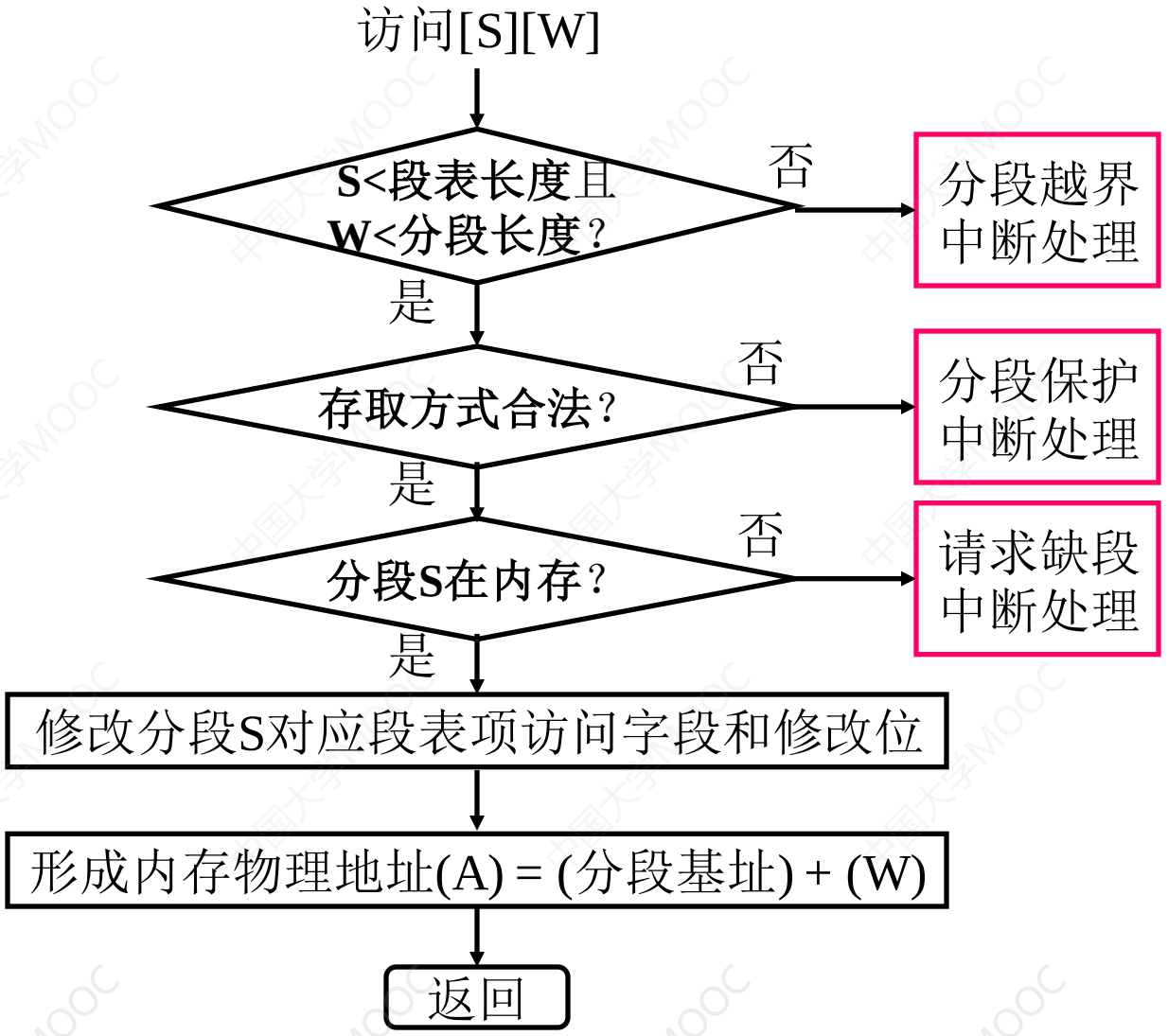
- 段表项的扩充



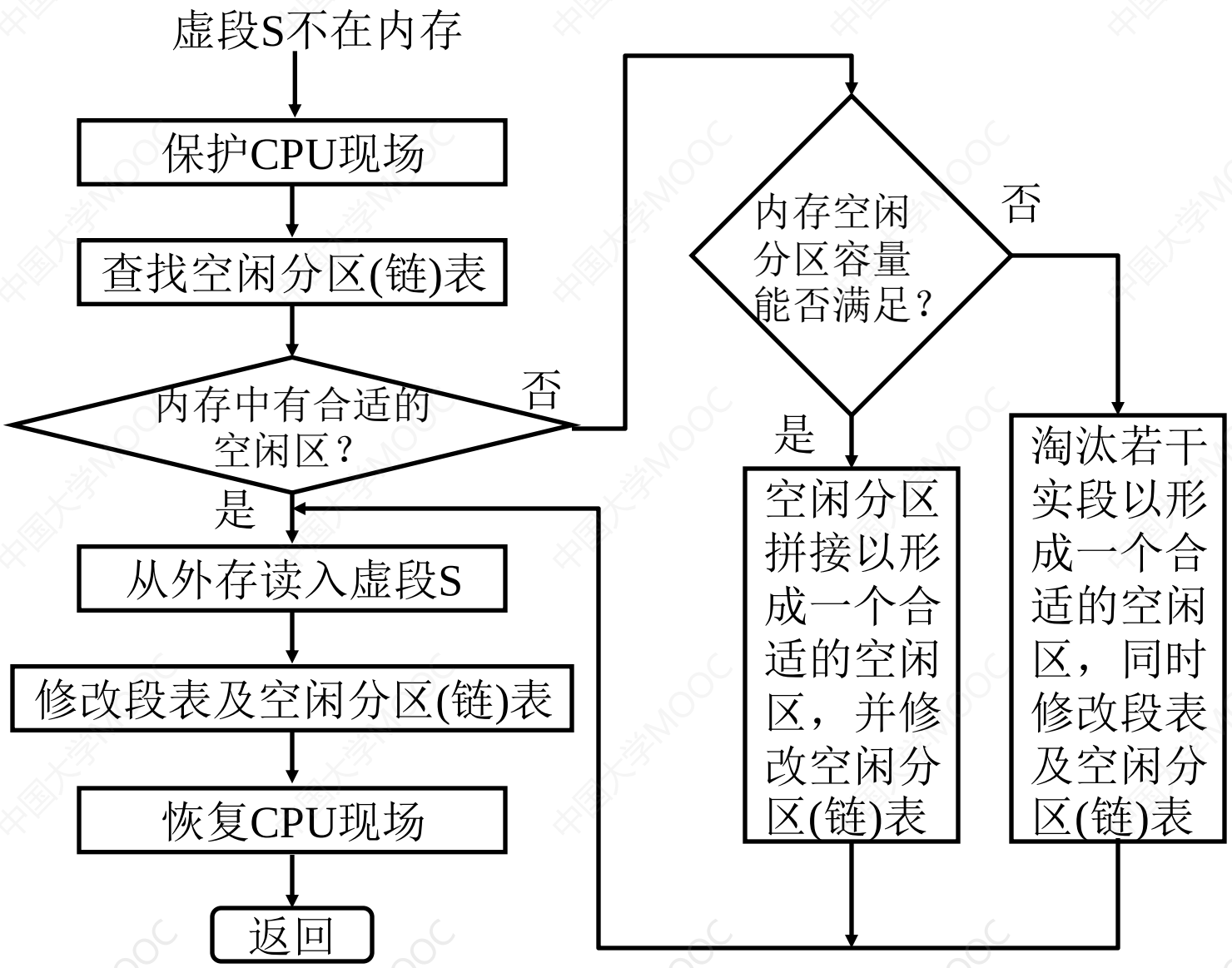
地址变换机构

- 在分段系统的地址变换机构的基础上，增加缺段中断产生和处理并分段置换功能而构成
- 地址变换过程要领
  - 从段表找到对应分段的段表项获悉该段尚未调入内存时，应产生缺段中断，请求操作系统从外存把该段调入内存
  - 关于快表和段表的检索及表项修改

请求分段系统地址变换


访问[S][W]
S<段表长度且
W<分段长度？
否
分段越界
中断处理
是
存取方式合法？
否
分段保护
中断处理
是
分段S在内存？
否
请求缺段
中断处理
是
修改分段S对应段表项访问字段和修改位
形成内存物理地址(A) = (分段基址) + (W)
返回

缺段中断机构及处理过程



4.7 请求分段存储管理方式

4.7.1 请求分段中的硬件支持

4.7.2 分段共享

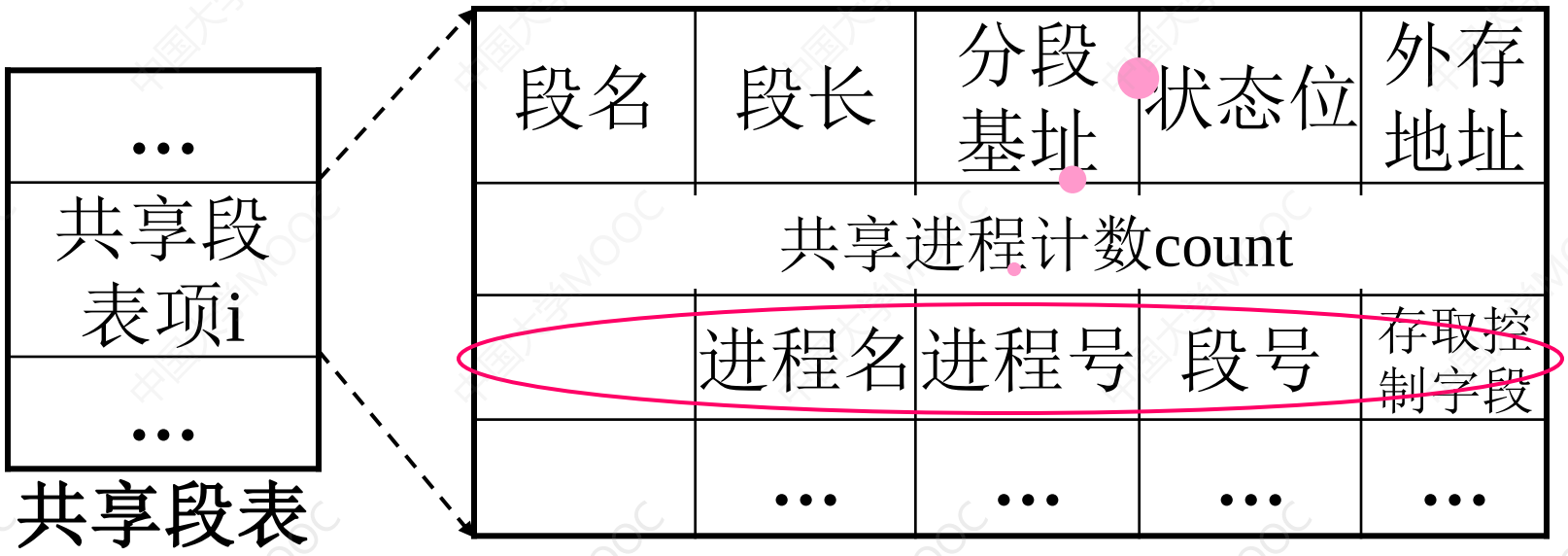
4.7.3 分段保护

4.7.4 x86体系中请求段页式支撑机制

共享段表

- 共享进程计数
- 存取控制字段
- 共享段不同段号

分段共享进程描述



共享段的分配与回收

- 共享段的分配
  - 对于第一个请求使用某共享段的进程，由系统为该共享段进行内存区的分配和装入，同时把共享段信息填入对应进程段表中，并在共享段表中为之增加一个表项和填写相关内容；对于以后其它进程提出共享该段要求，则仅需对对应的进程段表表项及共享段表表项修正即可
- 共享段的回收
  - 与共享段分配过程恰好相逆

4.7 请求分段存储管理方式

4.7.1 请求分段中的硬件支持

4.7.2 分段共享

4.7.3 分段保护

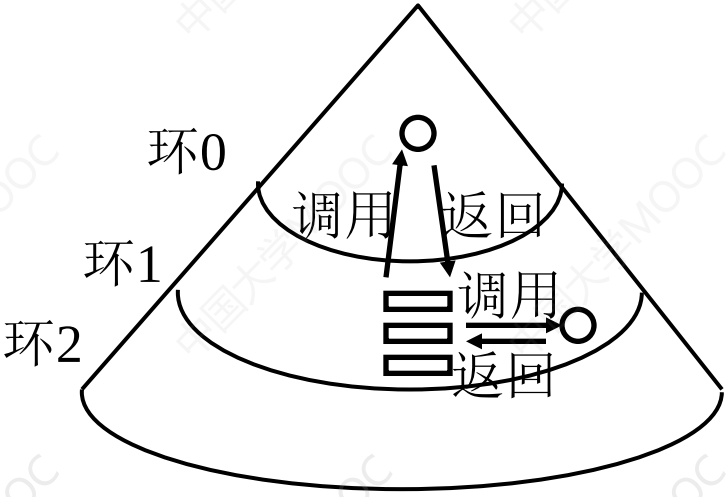
4.7.4 x86体系中请求段页式支撑机制

越界检查与存取控制检查

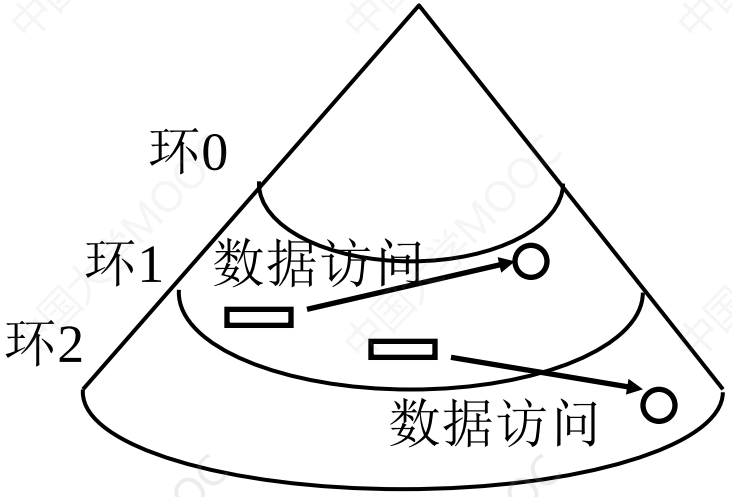
- 越界检查
  - 段号及段内地址合法性检查
- 存取控制检查
  - 段表中对应表项存取控制字段为依据
  - 通常访问方式包括只读、只执行（既不可读也不可写）、读/写三种
  - 共享段的存取控制应对不同进程赋予不同权限，既要保证信息安全，又要满足运行需要

环保护机构

- 程序间控制传输
  - 一个程序可调用驻留在相同环或较高特权环中的服务



- 数据访问
  - 一个程序可访问驻留在相同环或较低特权环中的数据



4.7 请求分段存储管理方式

4.7.1 请求分段中的硬件支持

4.7.2 分段共享

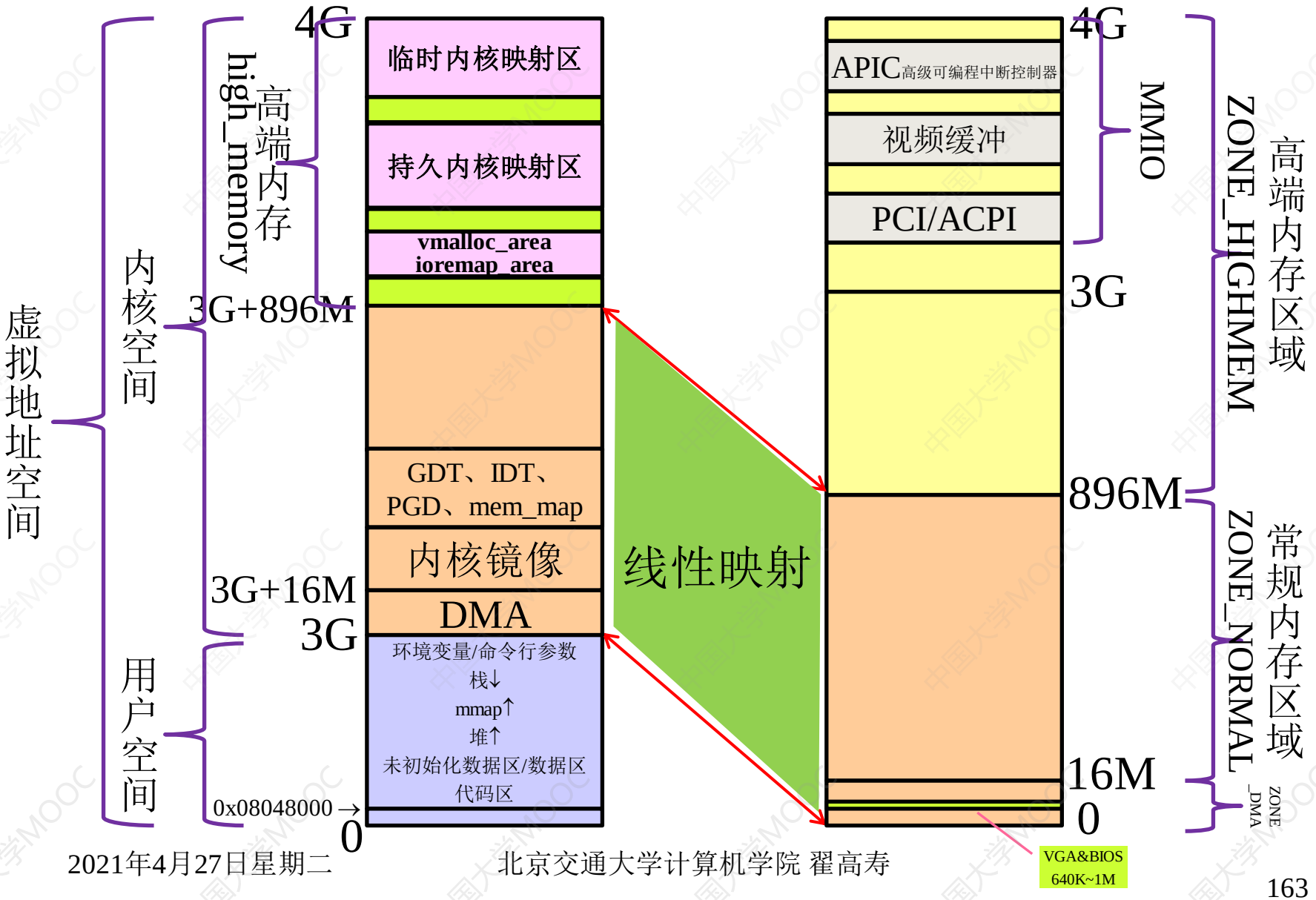
4.7.3 分段保护

4.7.4 x86体系中请求段页式支撑机制

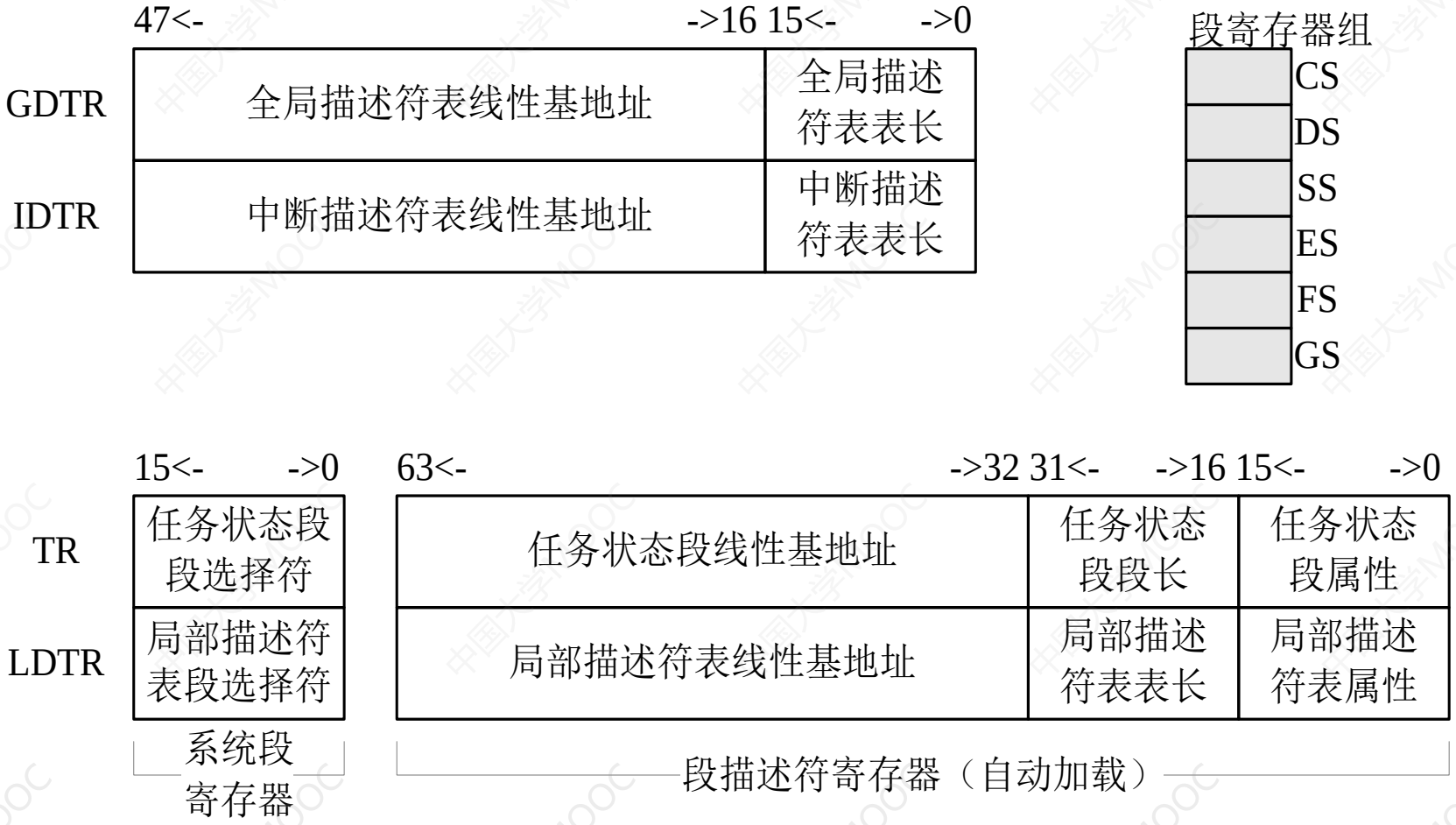
x86体系结构段页式存储管理

- 全局描述符表和局部描述符表
  - 代码段、数据段、堆栈段、系统段
  - 程序地址空间➔线性地址空间
  - lgdt/lldt
- 页目录表与页表
  - 线性地址空间➔物理地址空间
  - CR0/CR2/CR3（利用MOV指令操作）
- 系统内核空间与用户进程空间（Linux为例）
  - 线性地址空间划分（0~ [3GB] ~4GB）
  - 物理地址空间划分（0~16MB~896MB~4GB）

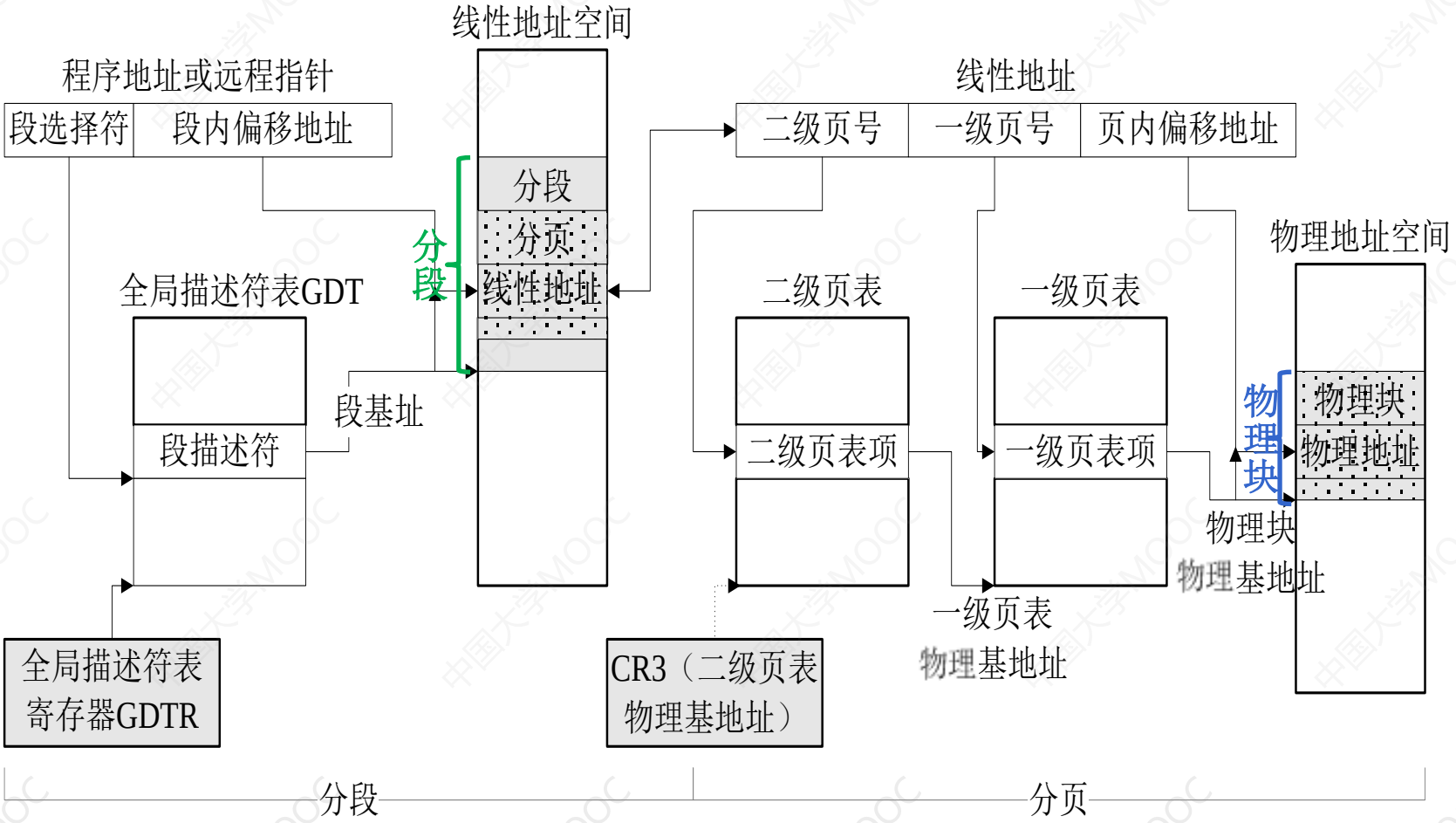
Linux虚拟/物理地址空间映射



x86体系结构存储相关寄存器



x86体系结构地址转换



4.7 请求分段存储管理方式

4.7.1 请求分段中的硬件支持

4.7.2 分段共享

4.7.3 分段保护

4.7.4 x86体系中请求段页式支撑机制

作业题

- **4.13** 试从内存分配、段表机制、地址变换、缺段中断以及调段过程等方面谈谈你对请求分段存储管理系统的认识与理解。
- **4.14** 分别阐述分段共享和分段保护的实现要领。
- **4.15** 计算机系统的虚拟存储器，其最大容量和实际容量分别取决于哪些因素？可通过哪些途径来提高内存利用率？

第四章 内存管理

4.1 内存管理概述
4.2 连续分配存储管理方式
4.3 基本分页存储管理方式
4.4 基本分段存储管理方式
4.5 虚拟存储器概念及关键技术
4.6 请求分页存储管理方式
4.7 请求分段存储管理方式

实验课题

实验课题14

Linux内存管理机制及页面淘汰算法探析

第四章
部分习题讲解

作业题

- **4.11[必做]** 某虚拟存储器的用户空间共有32个页面，每页1KB，主存16KB。假定某时刻系统为用户的第0、1、2、3页分别分配的物理块号为5、10、4、7，试将虚拟地址0A5C

每页1K

移地址

(0A5C)

物理地

每页1KB = 2^{10}字节，说明页内偏移地址占10位（二进制位）

$(093C)_{16} = (0000\ \mathbf{10}\ 01)_2(3C)_{16}$

页号2　页内地址

物理块号4　块内地址

物理地址 = $(0001\ \mathbf{00}\ 01)_2(3C)_{16}$

$= (113C)_{16}$

- **4.12[** 页面淘汰算 4、3、2、1、 配给该作业 计算访问过 ， 并比较所得结果。

同学们，再见！

2021年4月27日星期二