

第4章 存储管理

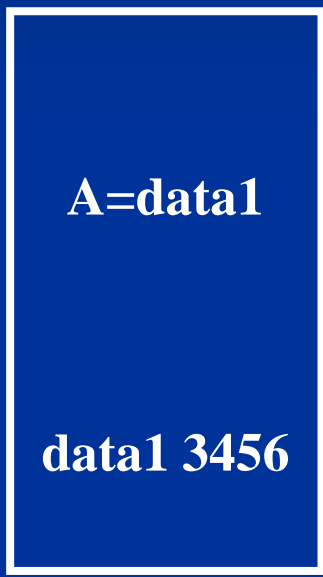
4.3 基本分页存储管理方式

- 分区式分配存在的问题：
碎片（内外“零头”）问题难以解决。

页框(page frame)

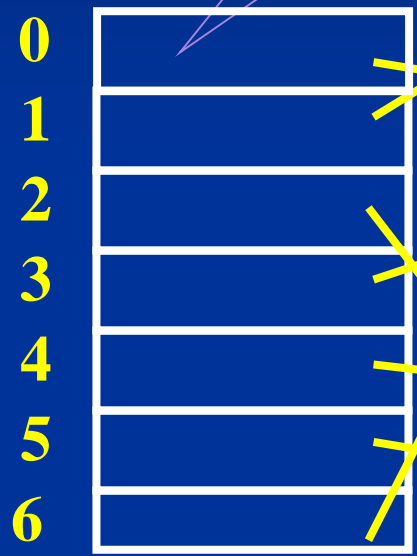
4.3.1 基本设计思想

(1) 设计思想概述



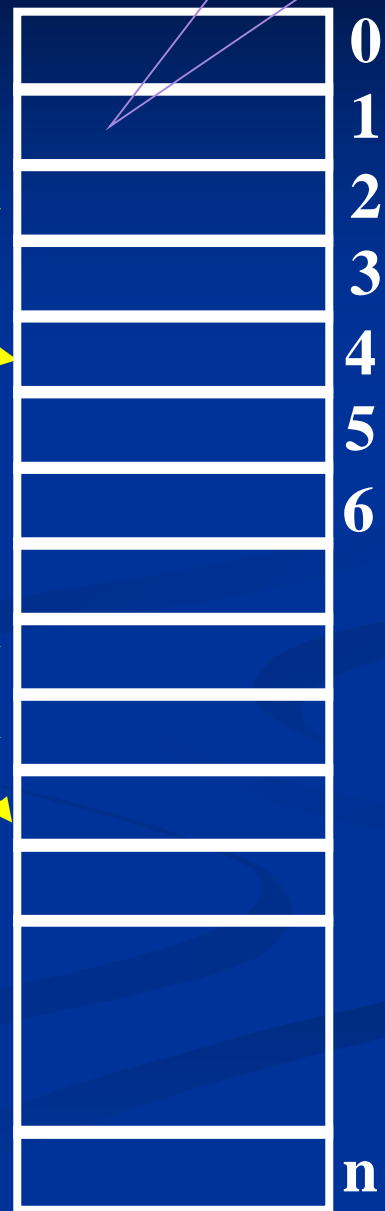
源程序

编译
连接



目标程序

内存/主存

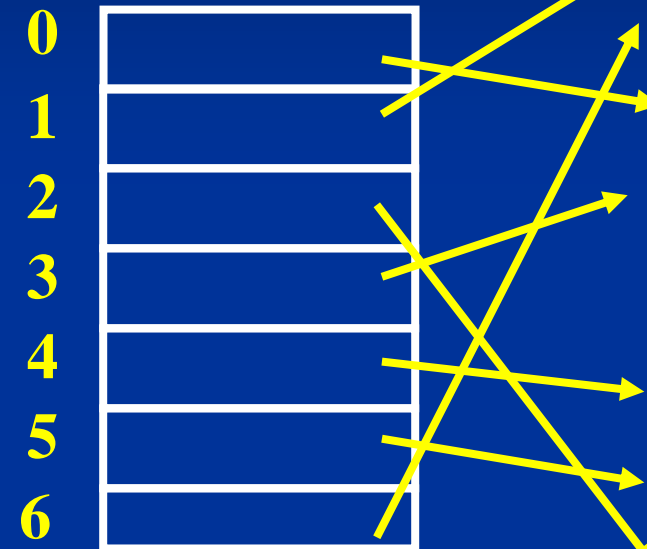


■ 要解决的问题:

- 页面/页框尺寸;
- 逻辑地址结构;
- 记录每个页面储存在哪个页框;
- 程序如何正确运行（地址重定位）？
- 指令跨页时的执行问题;
- 存储（内存）保护问题

内存/主存

(2) 页表



目标程序

页表:

页号	页框号	存取控制
0	4	X
1	2	X
2	10	X
3	5	X
4	8	RW
5	9	X
6	3	RW

- 页表存储在内存;
- 页表起始地址保存在PCB;

(3) 页面尺寸

页面太小:

内存利用率（内零头）高，页面数量增加，占用内存；

页面太大:

内存利用率（内零头）低，页面数量减少，节省内存；

正常尺寸:

2的整数次方，512-8K

(4) 地址结构

包括：页号和页内地址2部分。

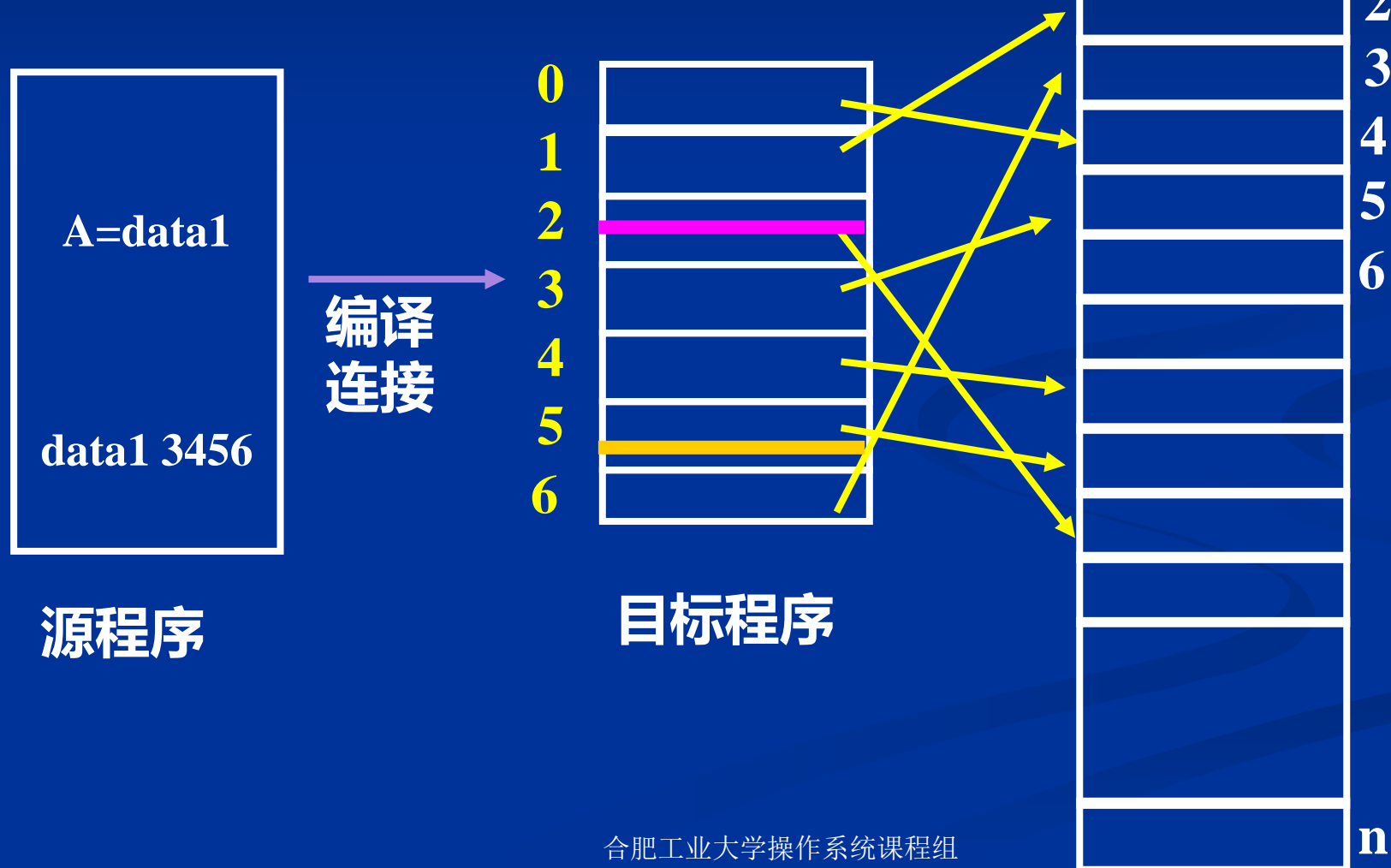


编号 $0 \sim 2^{20} - 1$

相对地址 $0 \sim 2^{12} - 1$

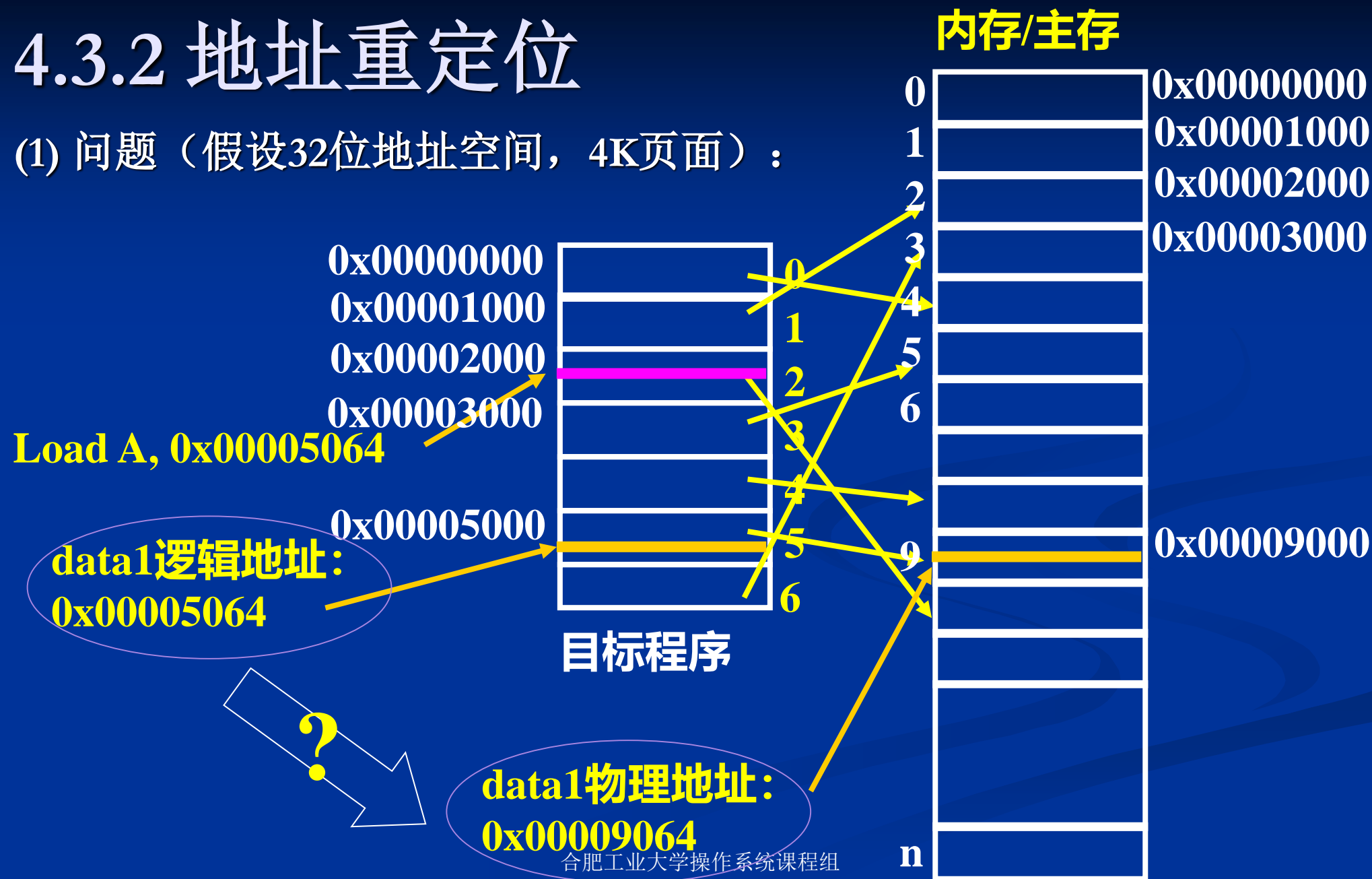
4.3.2 地址重定位

(1) 问题（假设4K页面）：

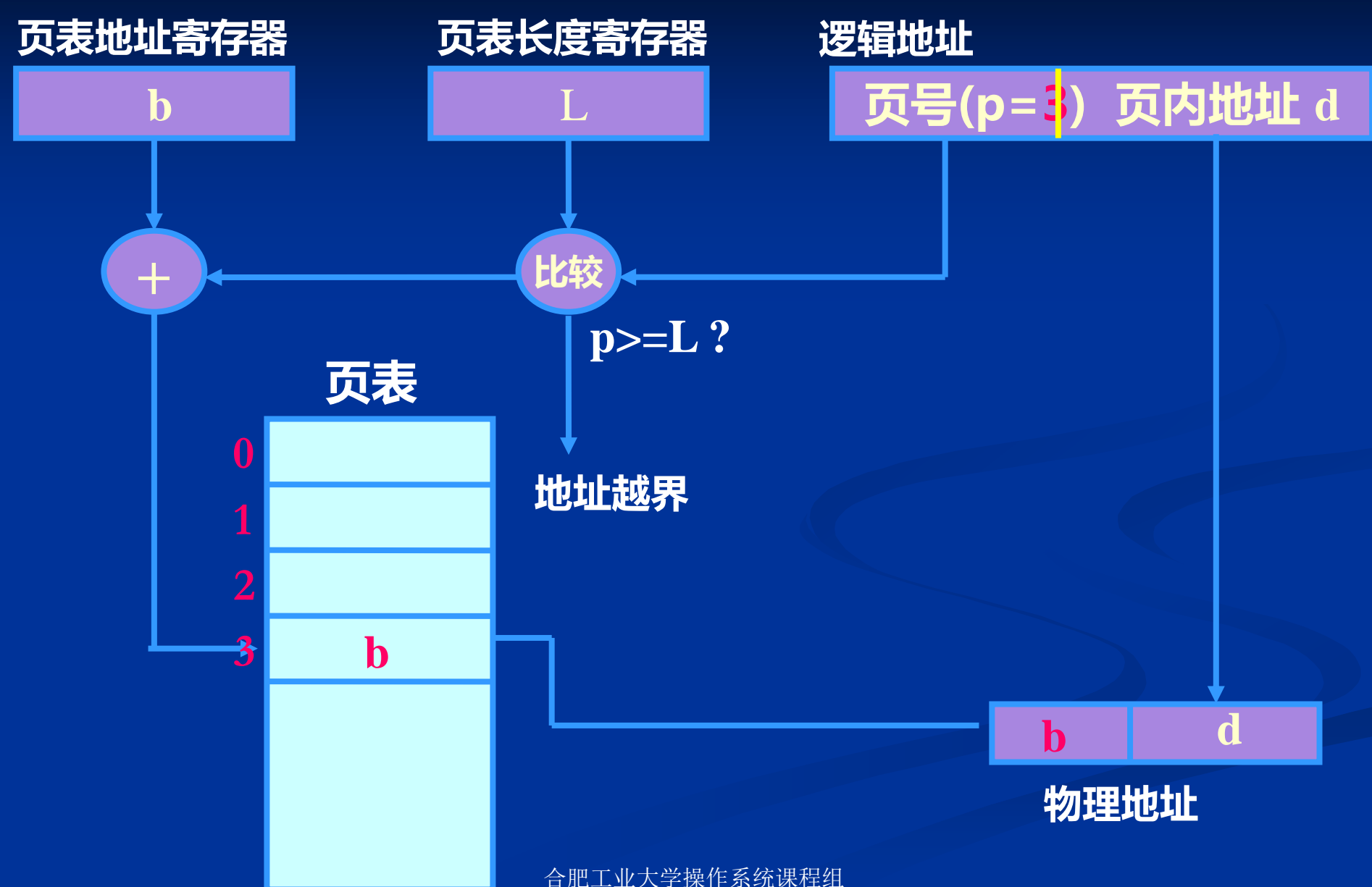


4.3.2 地址重定位

(1) 问题（假设32位地址空间，4K页面）：



(2) 基本地址变换机构/地址映射机制



(2) 基本地址变换机构/地址映射机制： 计算示例

假设32位地址空间， 4K页面



页表:

页号	页框号	存取控制
0	4	X
1	2	X
2	10	X
3	5	X
4	8	RW
5	9	X
6	3	RW

data1逻辑地址: 0x00005064



9



data1物理地址: 0x00009064

■ 页表的作用：分页系统的核心数据结构

- 记录程序各页面所在的页框位置；
- 支持进行地址重定位；
- 实现页面访问控制；
- 存储保护：限制程序在操作系统指定的内存区域内运行。

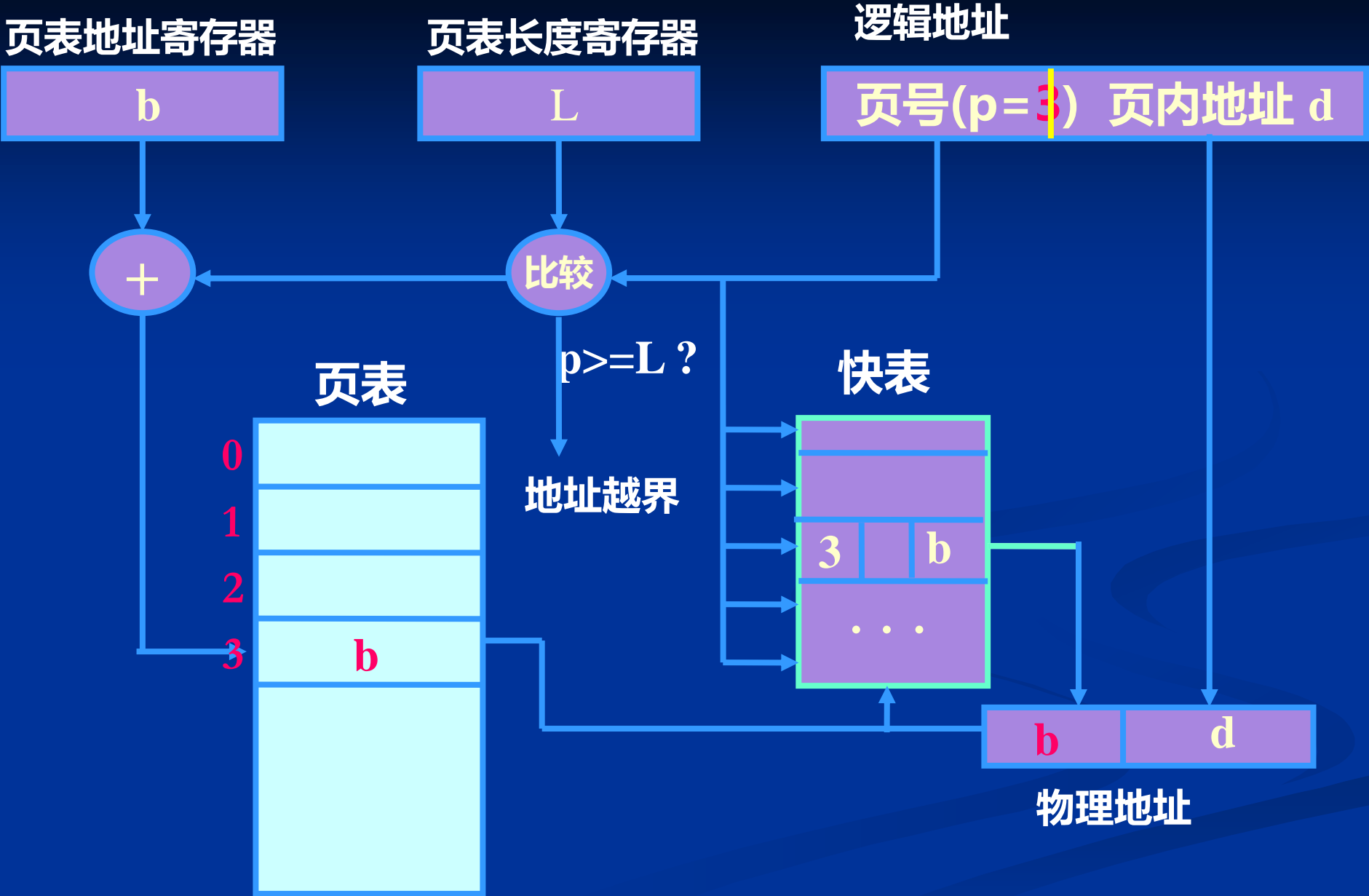
■ 性能问题:

访问1次内存变量，涉及2次地址访问：页表+变量

■ 解决办法：快表

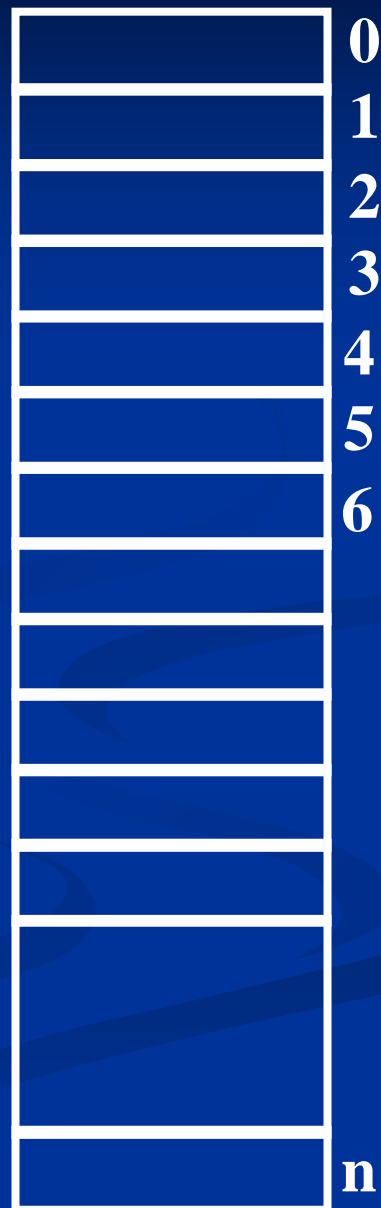
- 设置在CPU内部；
- 具有并行查找能力；
- 暂存当前正在使用的页表项；
- 尺寸：16-512 。可达到90%以上命中率
- 别名：
联想存储器（相联存储器）（associative memory）
Intel术语：TLB(Translation lookaside buffers)

(3) 具有快表的地址变换机构



4.3.3 两级和多级页表

内存/主存



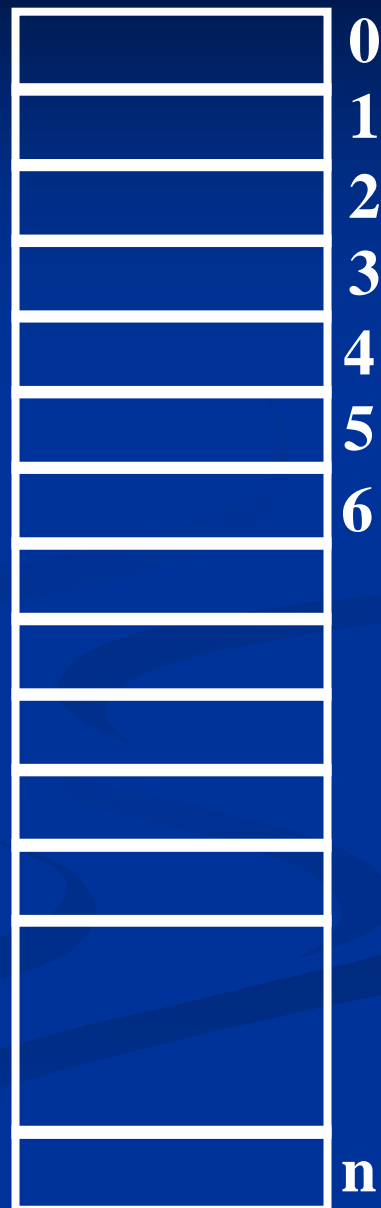
(1) 页表的存储问题（假设4K页面,4字节页表项）：

页表：

页号	页框号	存取控制
0	4	X
1	2	X
2	10	X
...
230	9879	RW
...
505	3	RW

4.3.3 两级和多级页表

内存/主存



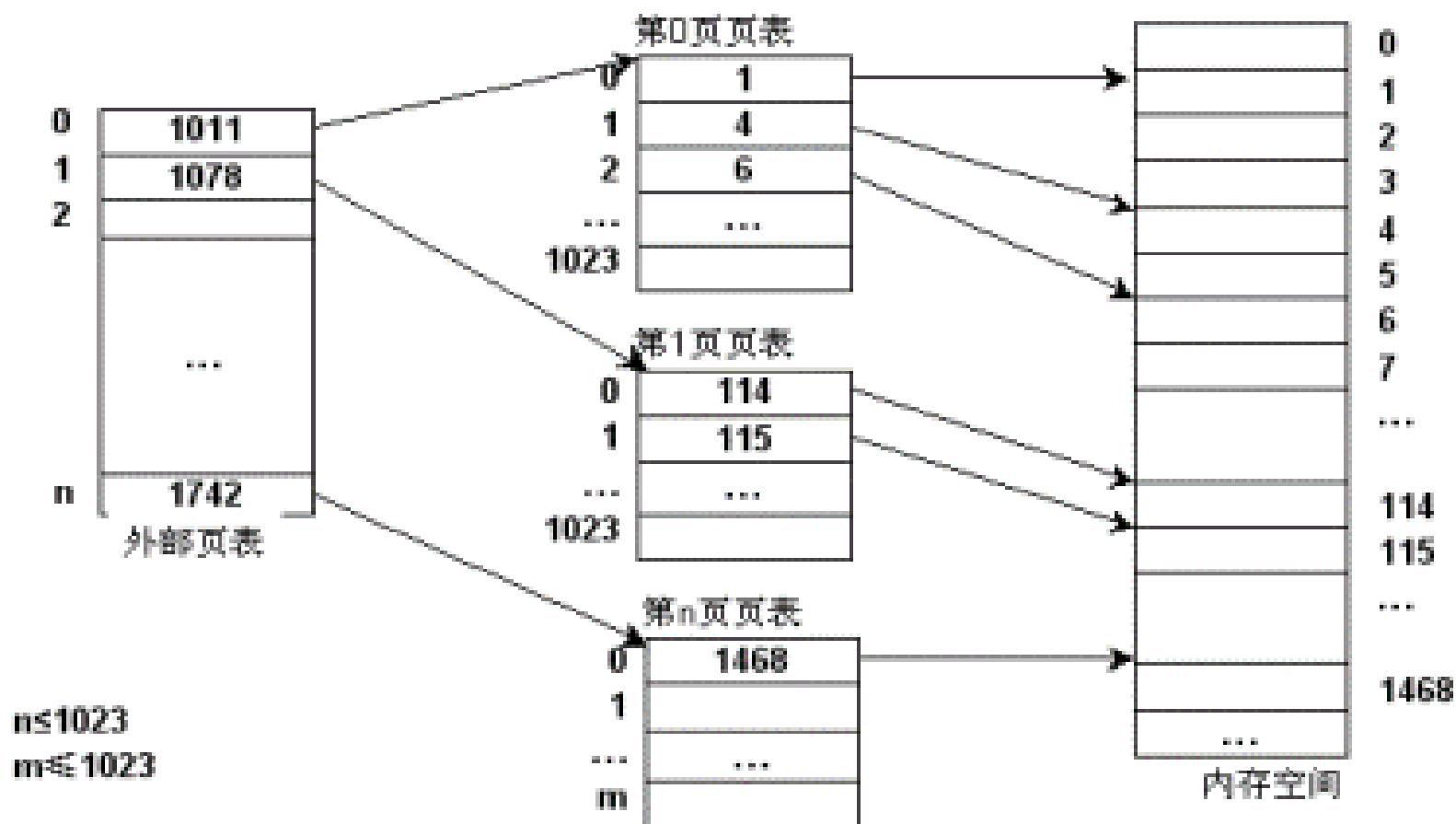
(1) 页表的存储问题（假设4K页面,4字节页表项）：

页表：

页号	页框号	存取控制
0	4	X
1	2	X
2	10	X
...
1023	9879	RW
...
1760	3	RW

4.3.3 两级和多级页表

(2) 两级页表



4.3.3 两级和多级页表

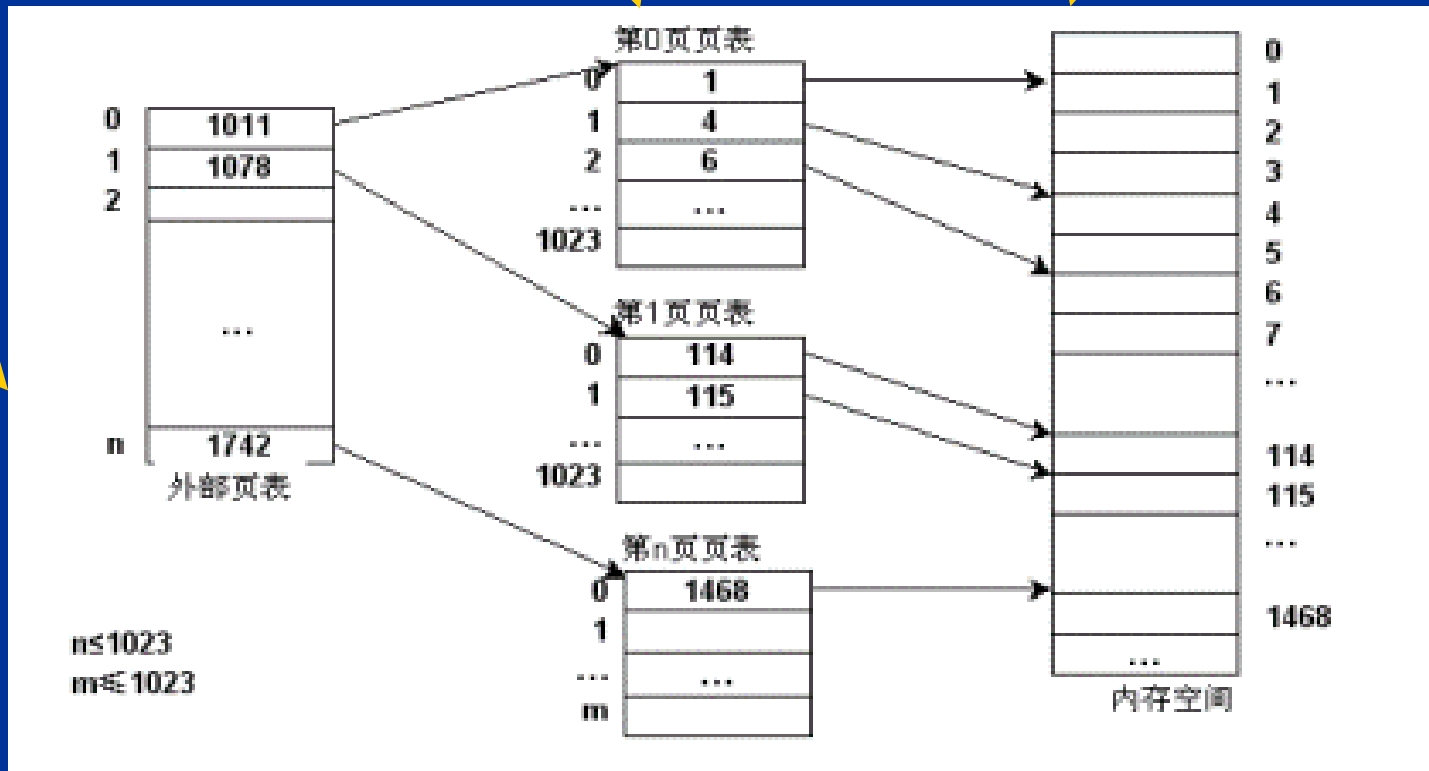
(2) 两级页表

■ 两级页表地址结构：



(2) 两级页表

■ 两级页表地址重新定位



4.3.3 两级和多级页表

(3) 多级页表

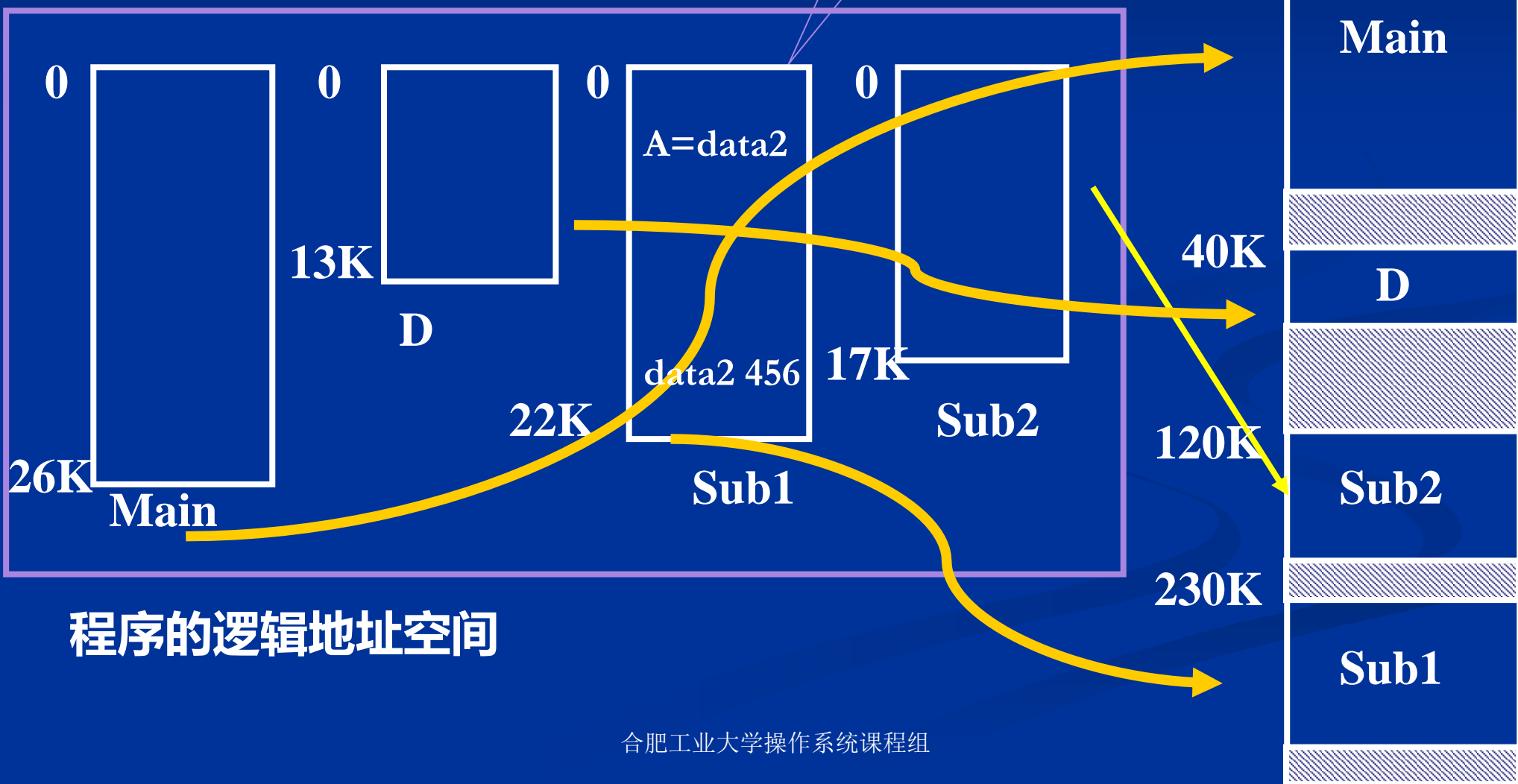
页表的2级索引推广到多级索引。

4.4 基本分段存储管理方式

- 分页系统存在的问题:
 - 信息共享和保护困难;
 - 编程不便;

4.4.1 基本设计思想

(1) 设计思想概述



4.4.1 基本设计思想

■ 要解决的问题：

- 分段划分；
- 逻辑地址结构；
- 记录每个分段储在内存位置；
- 程序如何正确运行（地址重定位）？
- 指令跨分段时的执行问题？
- 存储（内存）保护问题

4.4.1 基本设计思想

(2) 分段(Segment)

- 分段是一段有意义的信息集合;
- 分段的划分由程序员完成;
- 分段的长度不定;
- 指令不存在跨分段情况;

(3) 段表

段表:

段号	段基址	段长	存取控制
0	10K	26K	X
1	40K	13K	RW
2	120K	17K	X
3	230K	22K	X

- 段表存储在内存;
- 段表起始地址保存在PCB;

内存/主存

10K

0 Main

40K

1 D

120K

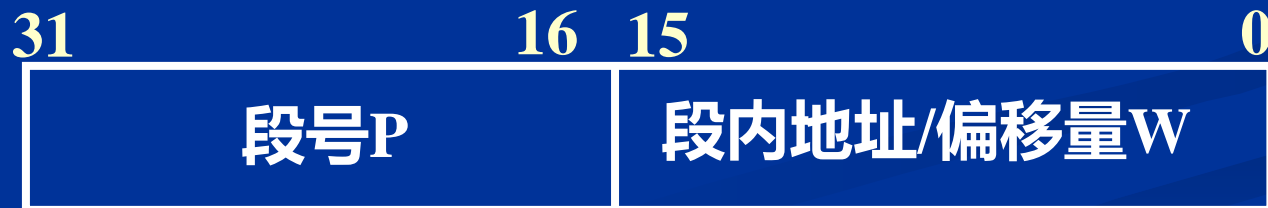
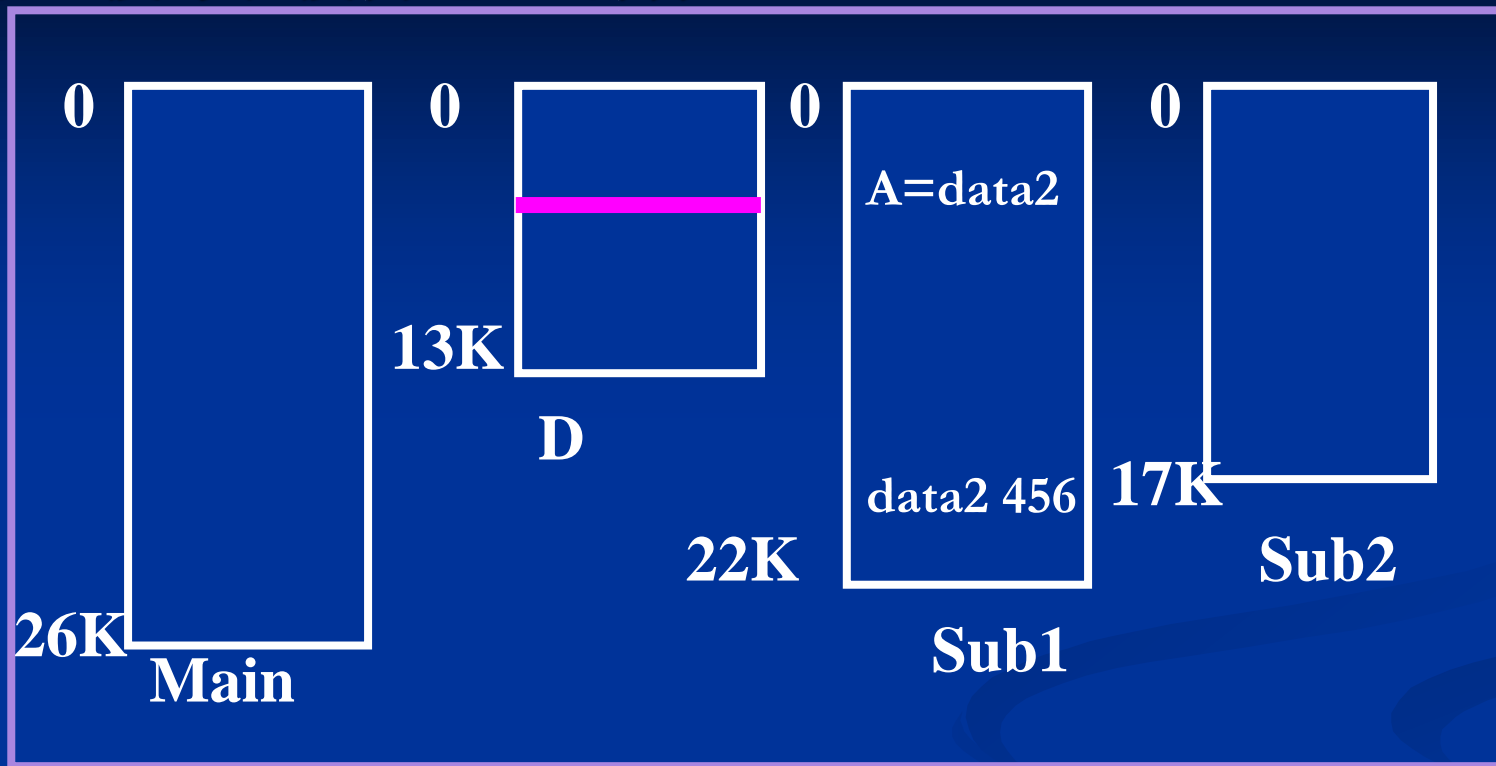
2 Sub2

230K

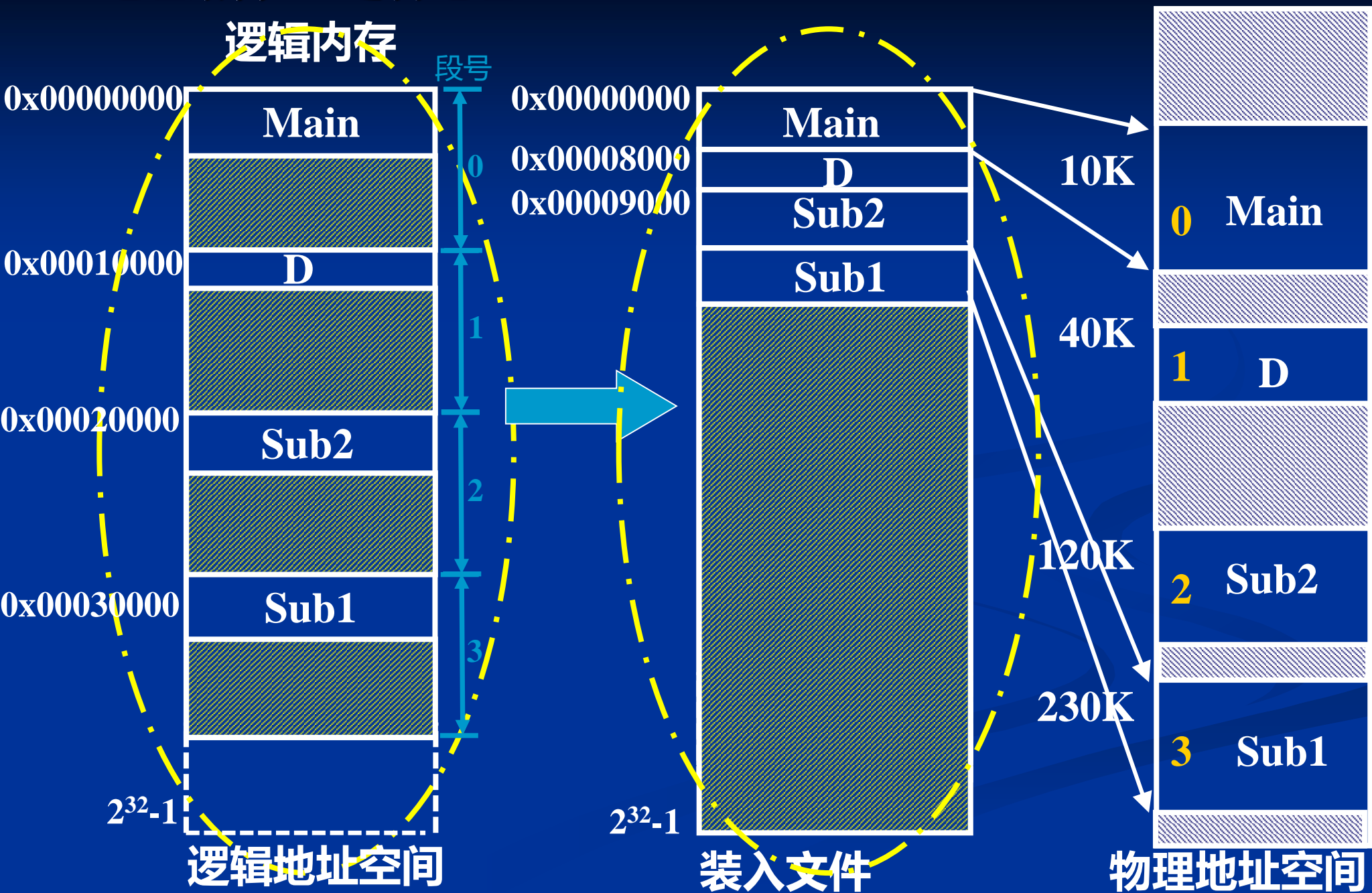
3 Sub1

(4) 地址结构

包括：段号和段内地址2部分。

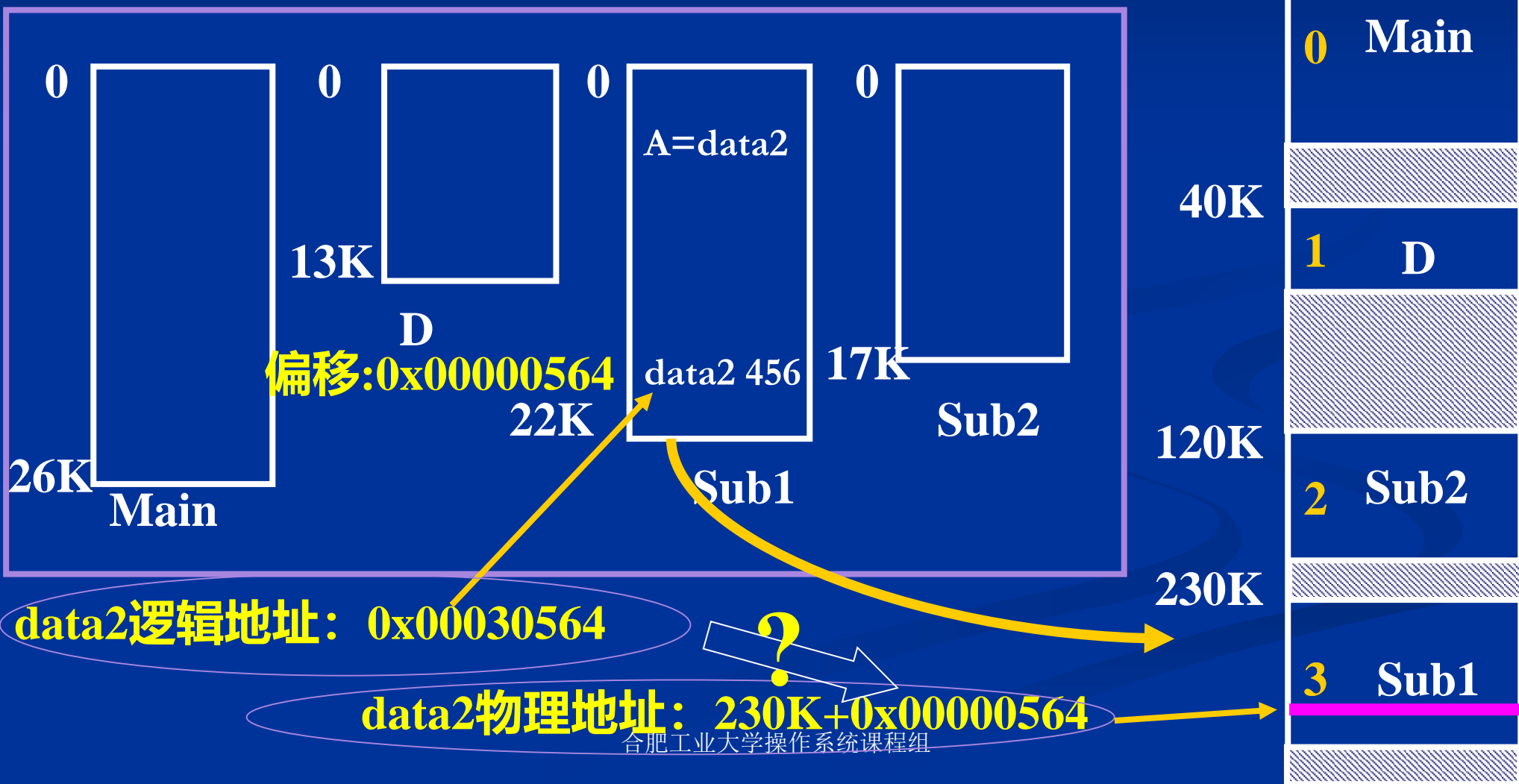


■ 地址结构 vs. 逻辑地址

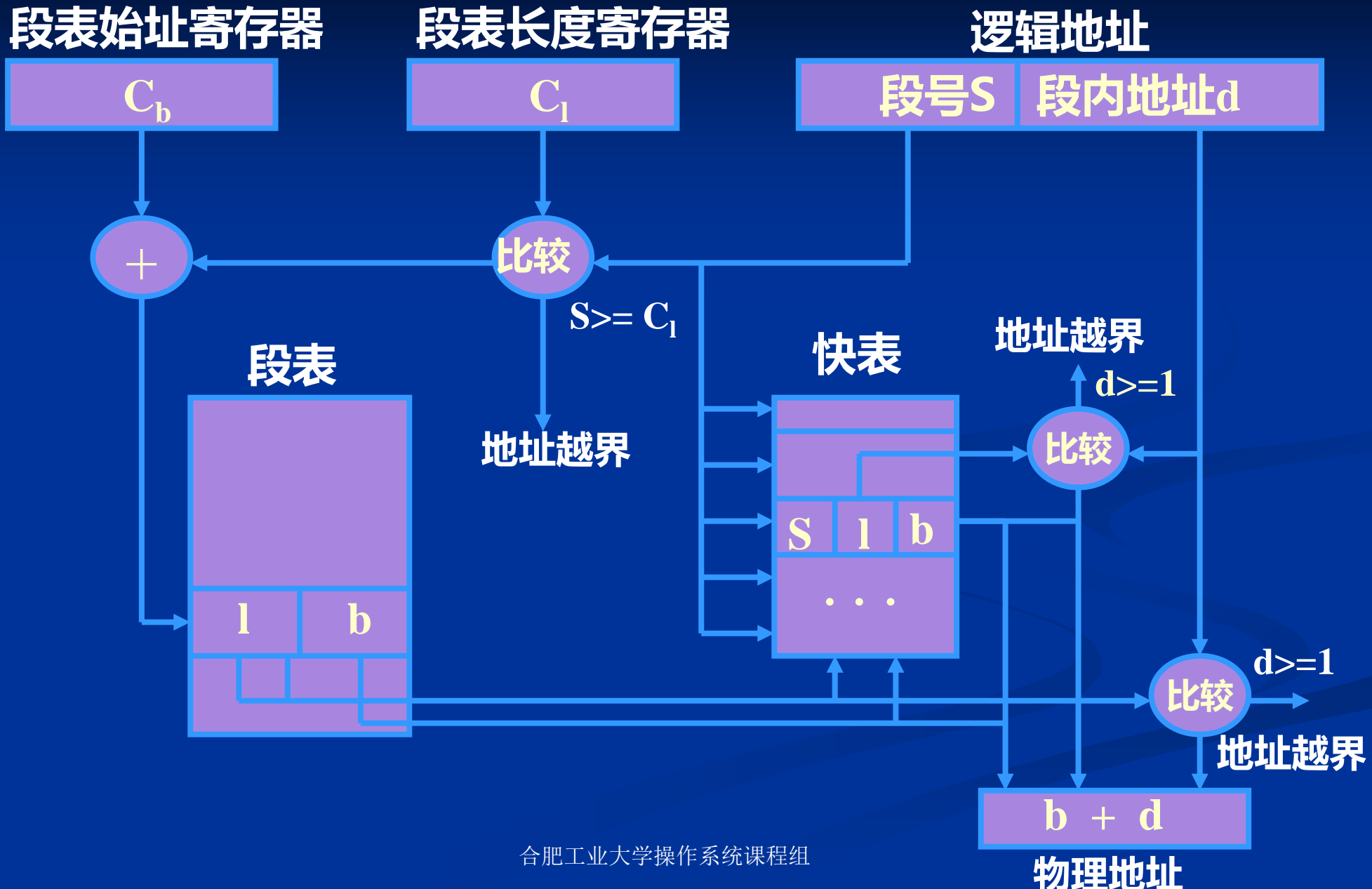


4.4.2 地址重定位

(1) 问题（假设32位地址，分段最大尺寸64K）：



(2) 分段地址变换及存储保护机构



(2) 地址变换机构/地址映射机制：计算示例

假设32位地址空间，分段最大64K



data1逻辑地址: 0x00030564



data1物理地址: 230K+0x00000564

段表:

段号	段基址	段长	存取控制
0	10K	26K	X
1	40K	13K	RW
2	120K	17K	X
3	230K	22K	X

- 段表的作用：分段系统的核心数据结构
 - 记录程序各分段所在的内存位置；
 - 支持进行地址重定位；
 - 实现分段访问控制；
 - 存储保护：限制程序在操作系统指定的内存区域内运行。

(3) 分段和分页区别

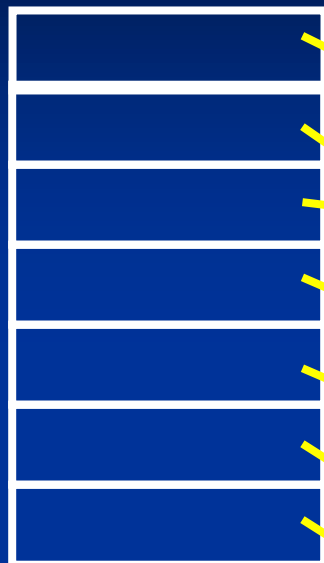
项目	分段	分页
信息单位	信息的逻辑单位	信息的物理单位
大小	不定	固定
可见性	程序员确定，可见	系统确定，程序员不可见
地址空间	二维地址空间	一维线性地址空间
信息共享保护	方便	不方便

4.4.3 信息共享

(1) 分页信息共享

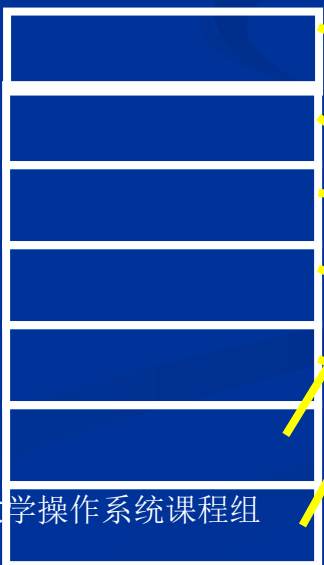
A页表:

0
1
2
3
4
5
6



B页表:

0
1
2
3
4
5
6



内存/主存

0
1
2
3
4
5
6



n

4.4.3 信息共享

(2) 分段信息共享

A段表:

段号	段基址	段长	存取控制
0	10K	26K	X
1	40K	13K	RW
2	120K	17K	X
3	230K	22K	X

B段表:

段号	段基址	段长	存取控制
0	2100K	7K	X
1	1040K	18K	RW
2	120K	17K	X
3	230K	22K	X

内存/主存



120K

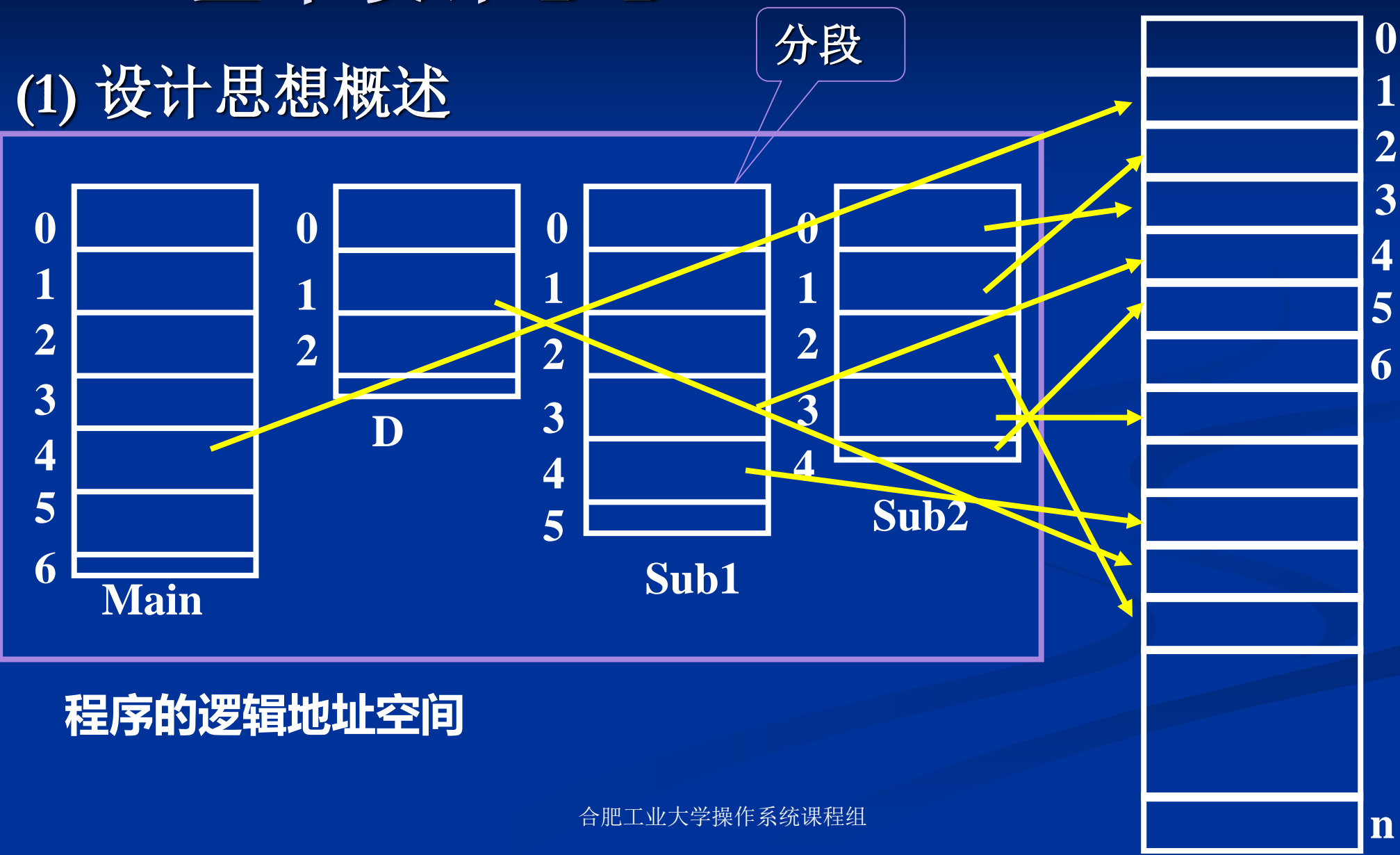
230K

4.5 段页式存储管理方式

- 分段系统存在的问题：
碎片（内外“零头”）问题难以解决；
- 解决方案(分段+分页):
 - 分页系统：
负责解决主存分配问题：内存按页分配；
 - 分段系统：
负责解决逻辑地址空间管理问题：按段为应用程序分配逻辑地址空间；

4.5.1 基本设计思想

(1) 设计思想概述



(2) 段表和页表



4.5.1 基本设计思想

(3) 地址结构



4.5.2 地址变换

