第7章 内存管理

- 主要内容
  - 7.1 内存管理的需求 ★★★★★
  - 7.2 内存分区 ★★★★★
  - 7.3 分页 ★★★★★
  - 7.4 分段 ★★★★★
  - 7.5 安全问题（自学）

7.1 内存管理的需求

- 重定位
- 保护
- 共享
- 逻辑组织
- 物理组织

7.1.1 重定位

- 逻辑地址（虚地址）：CPU所生成的地址
  - 指与当前数据在内存中的物理分配地址无关的访问地址
- 物理地址（实地址）：内存单元所看到的地址
- 重定位（地址转换）：把逻辑地址转换为物理地址
- 静态重定位
  - 地址转换工作在进程执行前一次完成；
  - 无须硬件支持，易于实现，但不允许程序在执行过程中移动位置。
- 动态重定位
  - 地址转换推迟到最后的可能时刻，即进程执行时才完成；
  - 允许程序在主存中移动、便于主存共享、主存利用率高。

7.1.2 保护

- 保护操作系统不受用户进程所影响，保护用户进程不受其他用户进程所影响
- 方法

1) 存储键保护
   - 系统将主存划分成大小相等的若干存储块，并给每个存储块都分配一个单独的保护键（锁）；在程序状态字PSW中设置有保护键字段，对不同的作业赋予不同的代码（钥匙）；钥匙和锁相配才允许访问

2) 界限寄存器（下页图）
   - 上、下界防护：硬件为分给用户作业的连续的主存空间设置一对上、下界，分别指向该存储空间的上、下界
   - 基址、限长防护：基址寄存器存放当前正执行者的程序地址空间所占分区的始址，限长寄存器存放该地址空间的长度

下限寄存器	2000

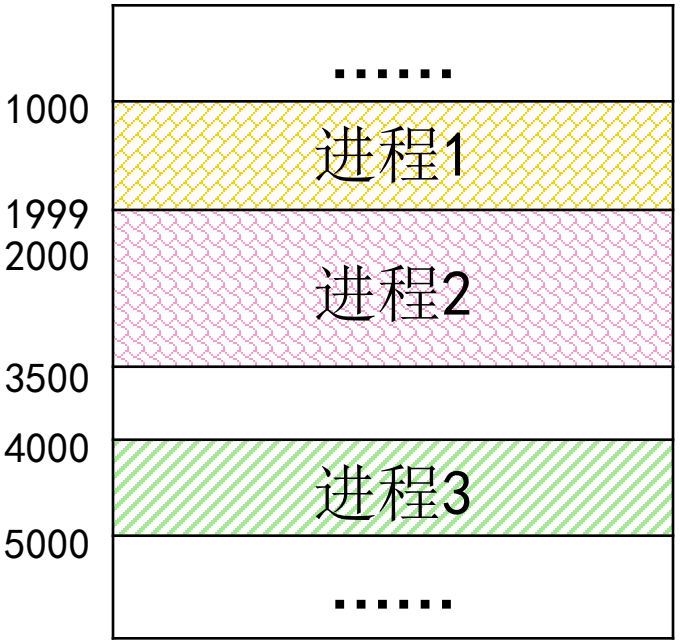
上限寄存器	3500

进程id	下限+上限寄存器	基址+限长寄存器
1	1000+1999	1000+999
2	2000+3500	2000+1500
3	4000+5000	4000+1000

基址寄存器	2000

限长寄存器	1500

内存映像



正运行的进程是进程2

7. 1. 3共享

- 某些场合，允许多个进程访问内存的同一部分。
  - 多个进程执行同一个程序，允许每个进程访问该程序的同一个副本
  - 合作完成同一个任务的进程可能需要共享访问相同的数据结构

7.1.4 逻辑组织

- 希望操作系统和计算机硬件能够有效地处理以某种模块的形式组织的用户程序和数据。

7.1.5 物理组织

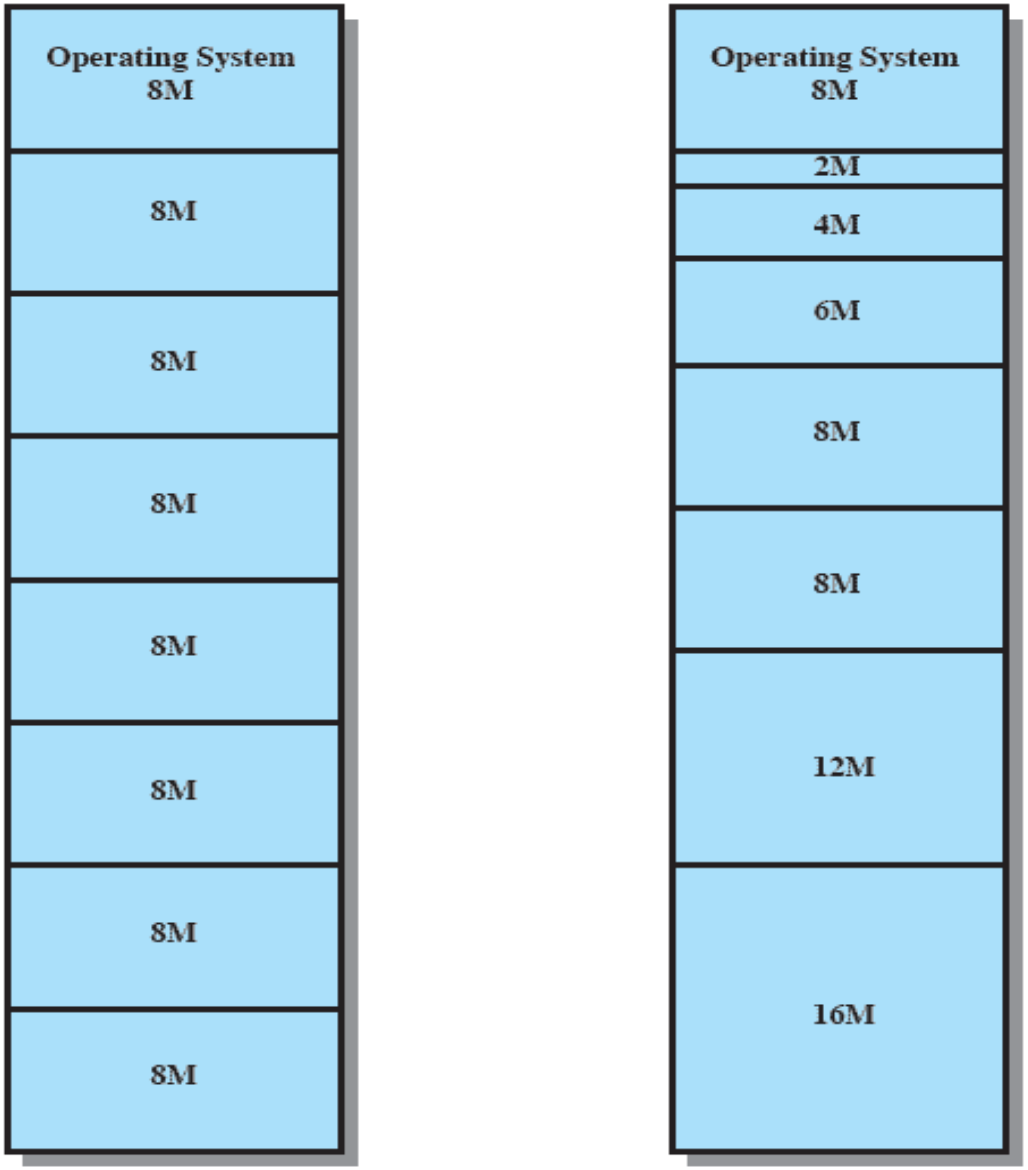
- 两级存储结构：内存和外存
  - 大容量的外存可以用于长期存储程序和数据
  - 较小的内存则用于保存当前使用的程序和数据
- 内存和外存之间信息流的组织

7.2 内存分区

7.2.1 固定分区

1、分区大小

- 大小相等的分区
  - 程序可能太大而不能放到一个分区中
  - 内存的利用率非常低，会有内部碎片
- 大小不等的分区
  - 可缓解上述问题，但不能完全解决。



(a) Equal-size partitions

(b) Unequal-size partitions

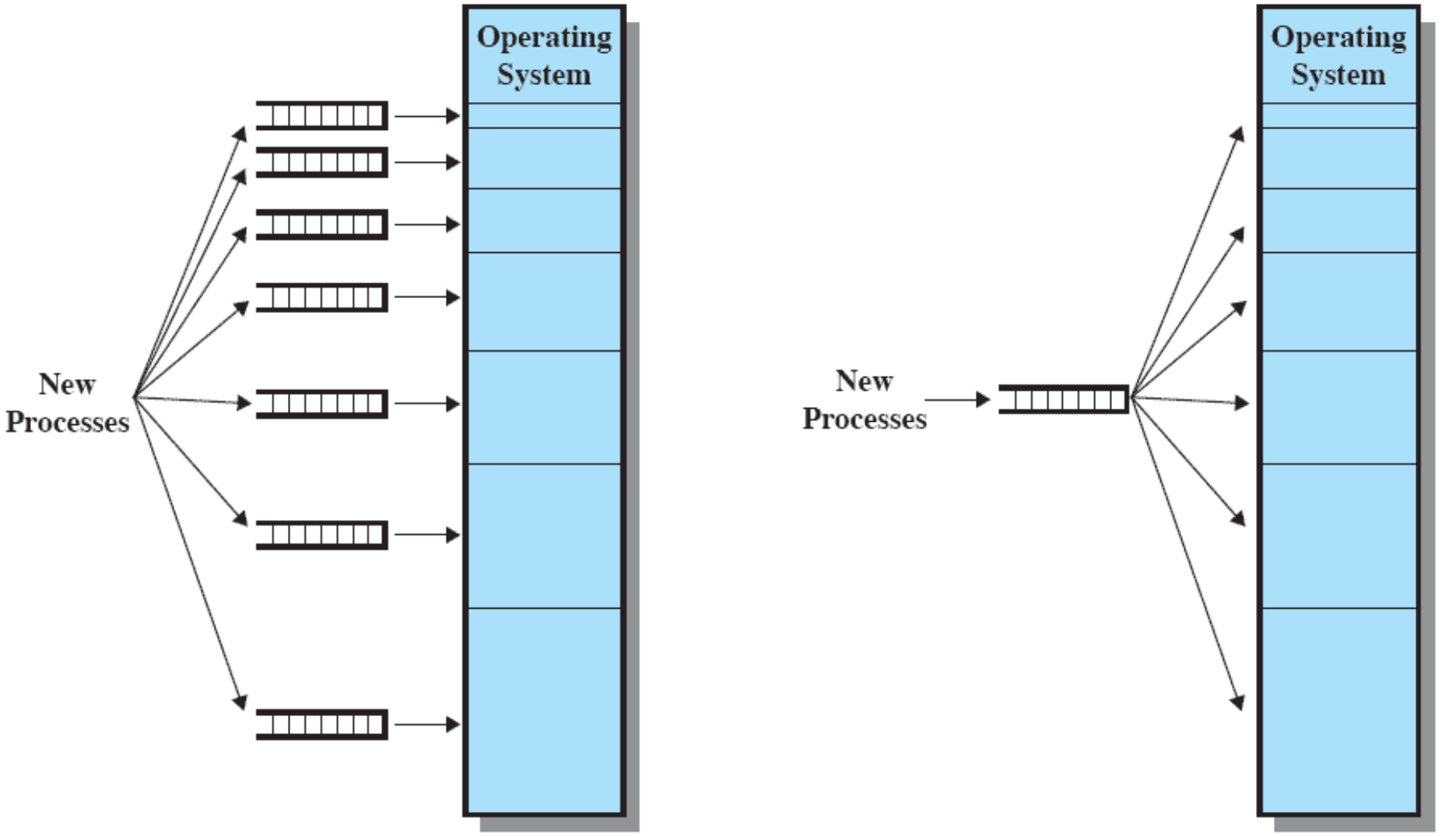
Figure 7.2 Example of Fixed Partitioning of a 64-Mbyte Memory

2、放置算法

- 大小相等的分区
  - 只要存在可用的分区，进程就可以装入分区。
  - 若所有分区都被处于不可运行状态的进程所占据，则选择其中一个进程换出，为新进程让出空间。
- 大小不等的分区
  - 把每个进程分配到能够容纳它的最小分区。

    每个分区维护一个调度队列，用于保存从这个分区换出的进程。

    为所有进程只提供一个队列。



(a) One process queue per partition

(b) Single queue

Figure 7.3 Memory Assignment for Fixed Partitioning

固定分区方案的缺陷

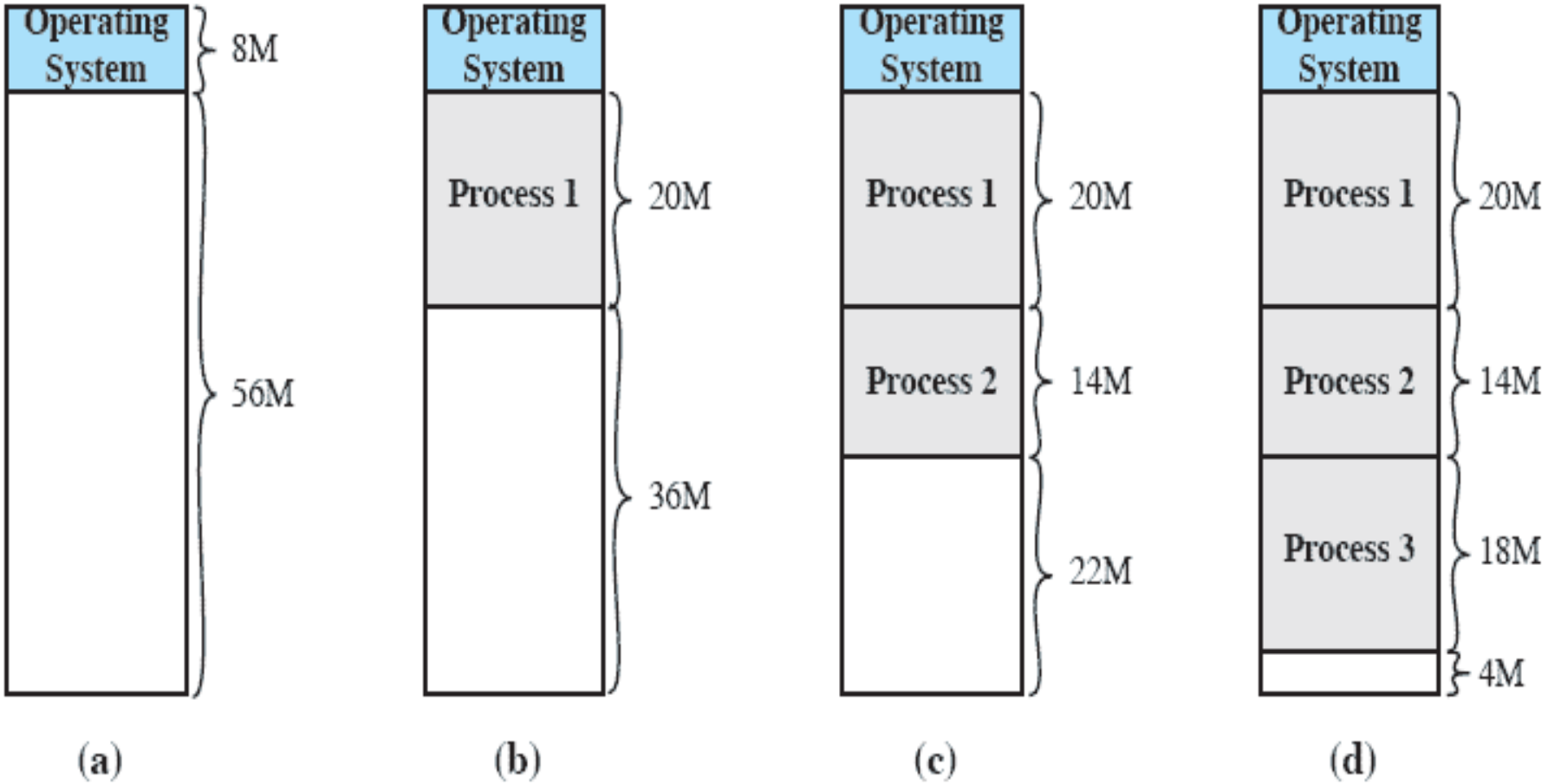
- 分区的数目在系统生成阶段已经确定，限制了系统中活动进程的数目。
- 分区大小在系统生成阶段事先设置，小作业不能有效地利用分区空间。

固定分区案例

- 早期的IBM主机操作系统OS/MFT
  - 具有固定任务数的多道程序设计系统

7.2.2 动态分区

- 分区长度和数目是可变的，当进程被装入内存时，系统会给它分配一块和它所需容量完全相等的内存空间。
- IBM主机操作系统OS/MVT
  - 具有可变任务数的多道程序设计系统
- 缺陷
  - 外部碎片
- 外部碎片解决方法
  - 压缩
  - 费时且浪费处理器时间


Operating System
8M
56M
(a)
Operating System
Process 1
20M
36M
(b)
Operating System
Process 1
20M
Process 2
14M
22M
(c)
Operating System
Process 1
20M
Process 2
14M
Process 3
18M
4M
(d)

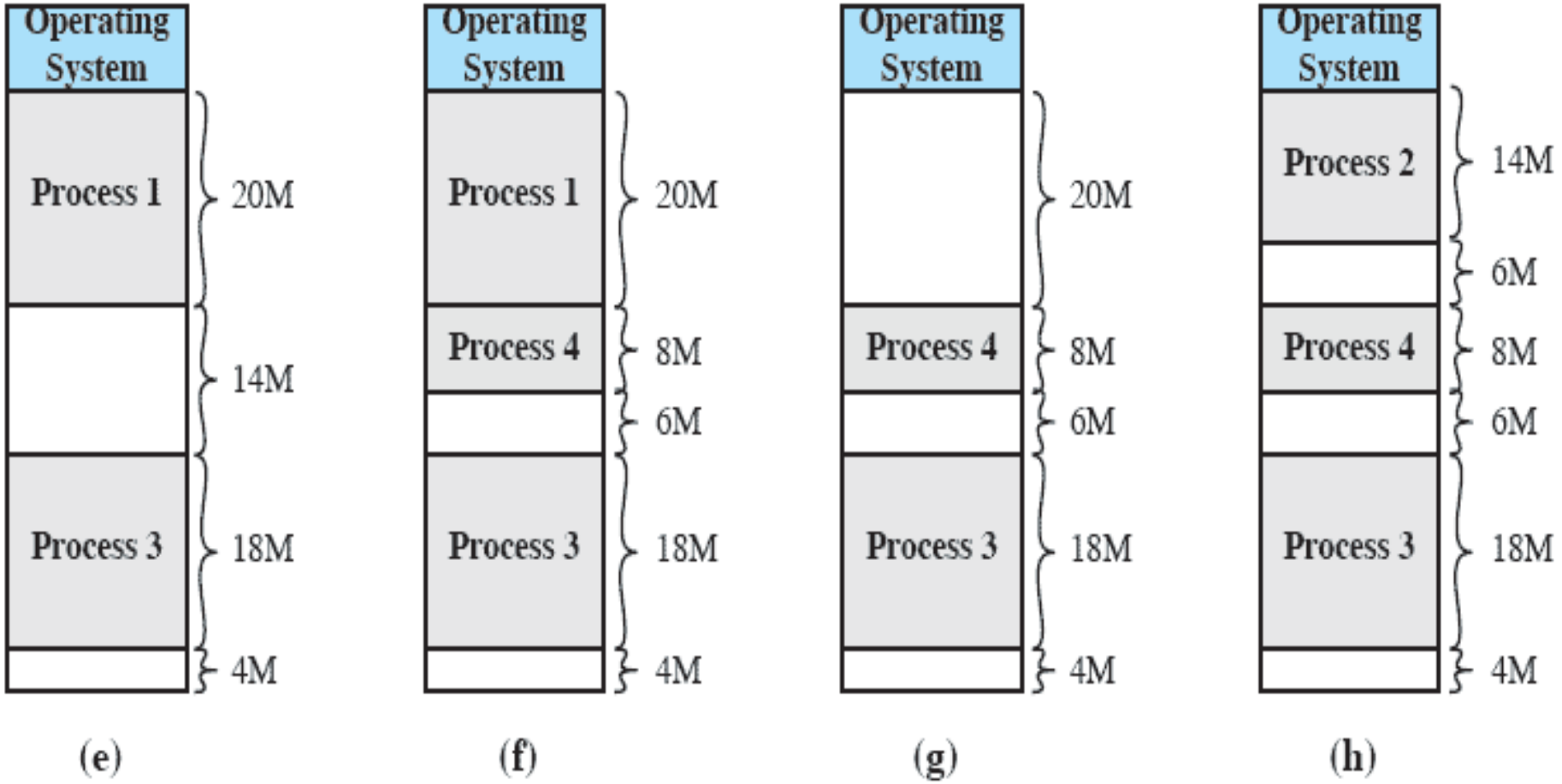
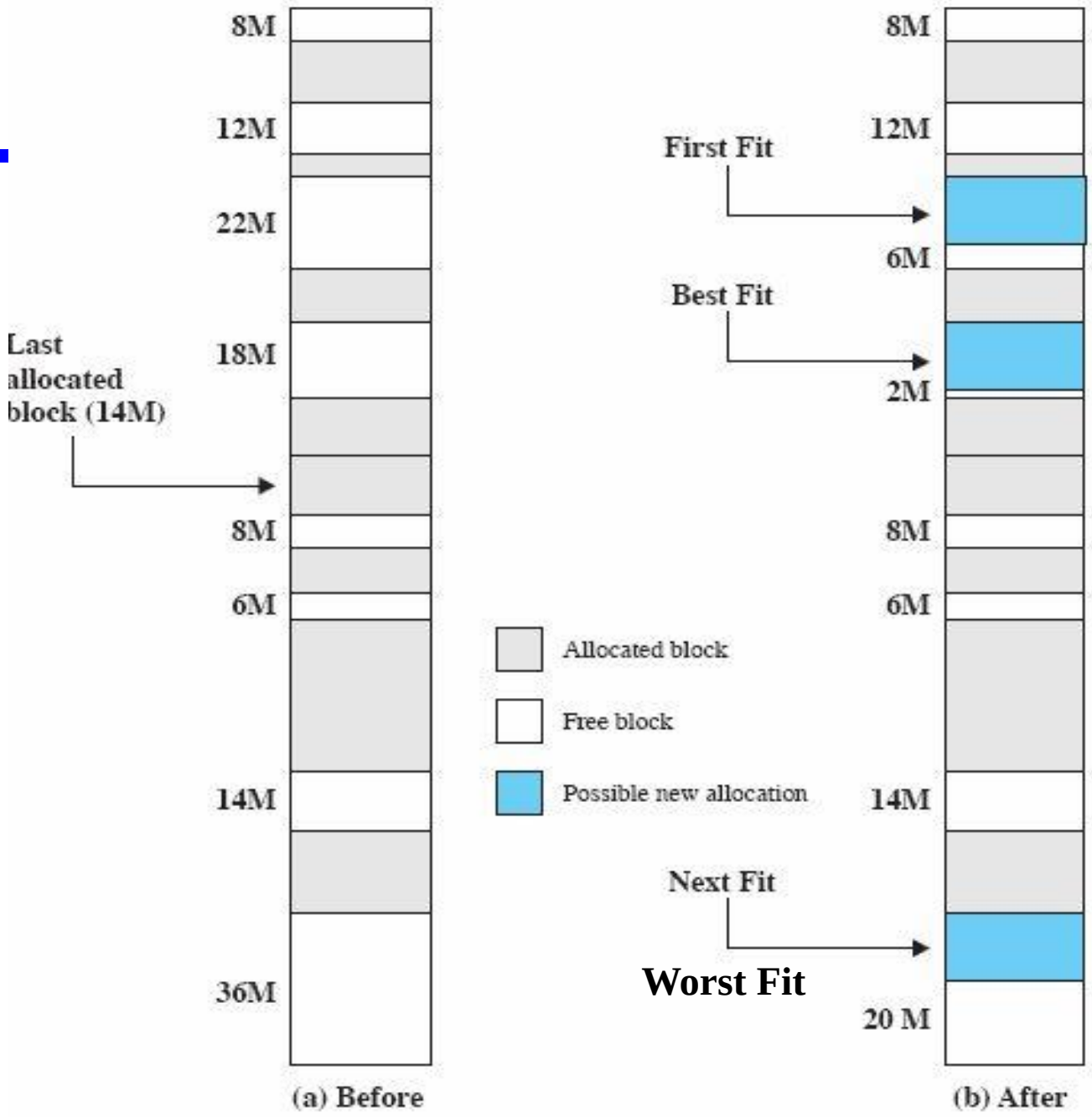


Figure 7.4 The Effect of Dynamic Partitioning

放置算法

- 首次适配（First Fit）
  - 从开始扫描内存，选择大小足够的第一个可用块；
- 下次适配（Next Fit）
  - 首次适配的变种，每次分配时从未分配区的上次扫描结束处顺序查找，选择下一个大小足够的可用块。
- 最佳适配（Best Fit）
  - 选择与要求的大小最接近的块。
- 最差适配（Worst Fit）
  - 选择符合要求大小的最大容量的块。


8M
12M
22M
18M
Last allocated block (14M)
8M
6M
14M
36M
(a) Before
Allocated block
Free block
Possible new allocation
8M
12M
First Fit
6M
Best Fit
2M
8M
6M
14M
Next Fit
Worst Fit
20 M
(b) After

提问

1. 固定分区方案会产生（）。

A. 内部碎片　　B. 外部碎片

2. 在动态分区方案中，（）通常是最好、最快的。

A. 首次适配算法　　B. 最佳适配算法

C. 下次适配算法　　D. 最差适配算法

3. 保护操作系统不受用户进程所影响，保护用户进程不受其他用户进程所影响，保护的方法通常有存储键方法、界限寄存器方法。

A. 正确　　B. 错误

7.3 伙伴系统

- 伙伴系统是一种固定分区和可变分区折中的主存管理算法。

- 伙伴系统涉及三个参数，L, K, U.

  在伙伴系统中, 可用内存块的大小为2^K个字, , 其中

–2^L表示分配的最小块的尺寸

–2^U表示分配的最大块的尺寸；通常2^U是可供分配的整个内存的大小

7.2.3 伙伴系统

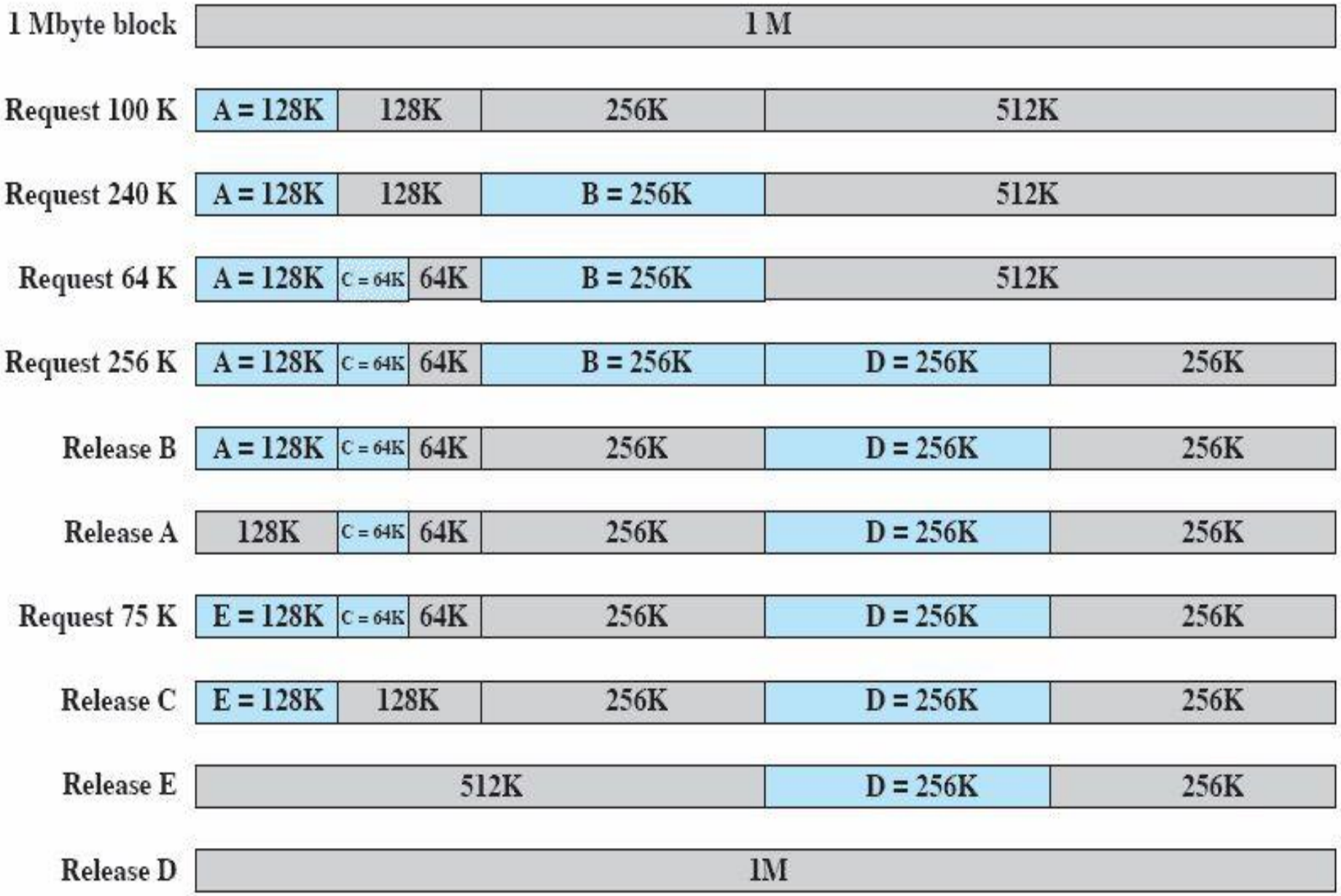
- 伙伴系统是一种固定分区和可变分区折中的主存管理算法。
- 伙伴系统分配原理
  - 可用于分配的整个空间被视为一个大小为2^U的块，若请求的大小s满足$2^{U-1} < s \leqslant 2^U$, 则分配整个空间；
  - 否则，该块被分成两个大小相等的伙伴2^{U-1}，如果有$2^{U-2} < s \leqslant 2^{U-1}$ ，则给该请求分配两个伙伴中的任何一个，否则，其中一个伙伴又被分成两半……，该过程一直继续，直到产生大于或等于s的最小块。

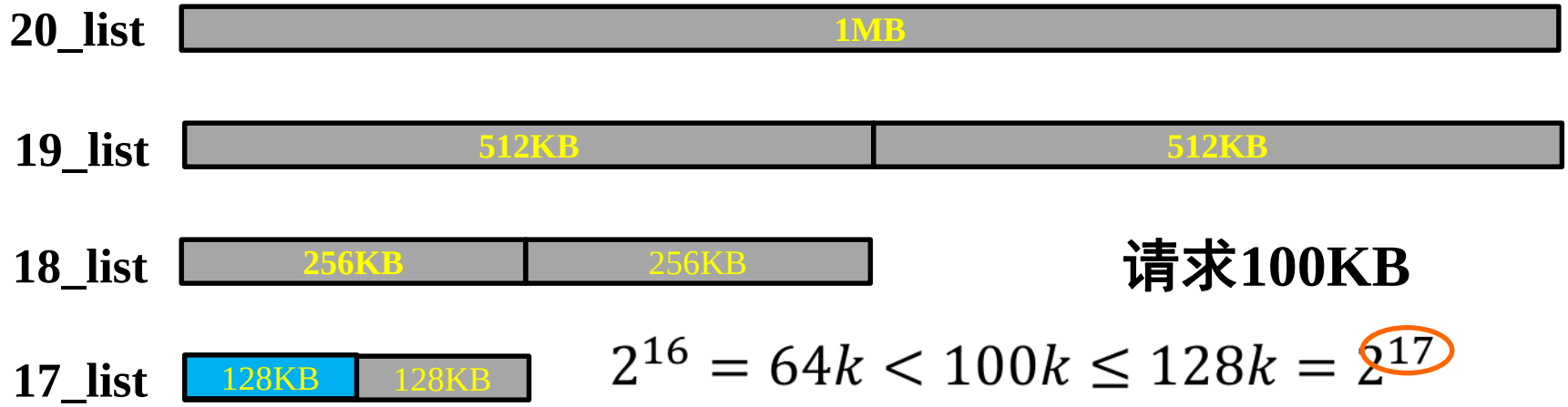
若请求的块大小 k 满足 $2^{i-1} < k \leq 2^i$

7.2.3 伙伴系统

调用get_hole(i)

```
void get_hole(int i)
{
     if (i == (U+1)) <failure>;
     if (i_list empty) {
          get_hole(i + 1);
          <split hole into buddies>;
          <put buddies on i_list>;
     }
     <take first hole on i_list>;
}
```


1 Mbyte block
1 M
Request 100 K
A = 128K
128K
256K
512K
Request 240 K
A = 128K
128K
B = 256K
512K
Request 64 K
A = 128K
C = 64K
64K
B = 256K
512K
Request 256 K
A = 128K
C = 64K
64K
B = 256K
D = 256K
256K
Release B
A = 128K
C = 64K
64K
256K
D = 256K
256K
Release A
128K
C = 64K
64K
256K
D = 256K
256K
Request 75 K
E = 128K
C = 64K
64K
256K
D = 256K
256K
Release C
E = 128K
128K
256K
D = 256K
256K
Release E
512K
D = 256K
256K
Release D
1M



请求100KB

$$2^{16} = 64k < 100k \leq 128k = 2^{17}$$

$$1\text{MB} = 2^{20}\text{B}$$

$$U = 20$$

get_hole(17)

↓

get_hole(18)

↓

get_hole(19)

↓

get_hole(20)

```
void get_hole(int i)
{
        if (i == (U+1)) <failure>;
        if (i_list empty) {
                get_hole(i + 1);
                <split hole into buddies>;
                <put buddies on i_list>;
        }
        <take first hole on i_list>;
}
```

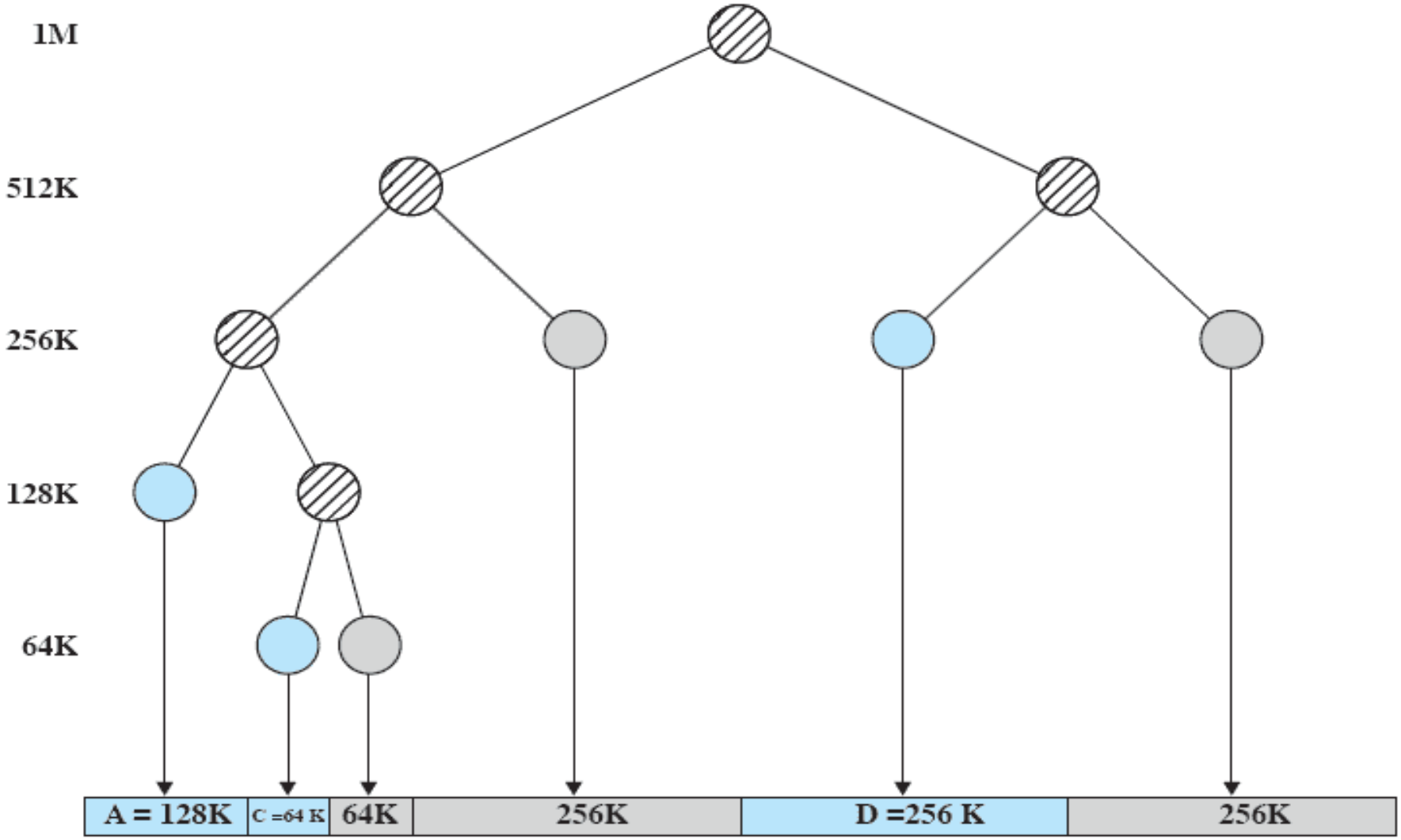


Figure 7.7 Tree Representation of Buddy System

7.3 分页

- 内存被划分成大小固定相等的块（页框），且块相对比较小，每个进程也被分成同样大小的块（页）。
- 进程中称为页的块可以指定到内存中称为页框的可用块。
- 分页与固定分区的区别
  - 分页技术的分区相当小
  - 一个程序可以占据多个分区
  - 一个程序占据的多个分区不需要是连续的

7.3 简单分页技术

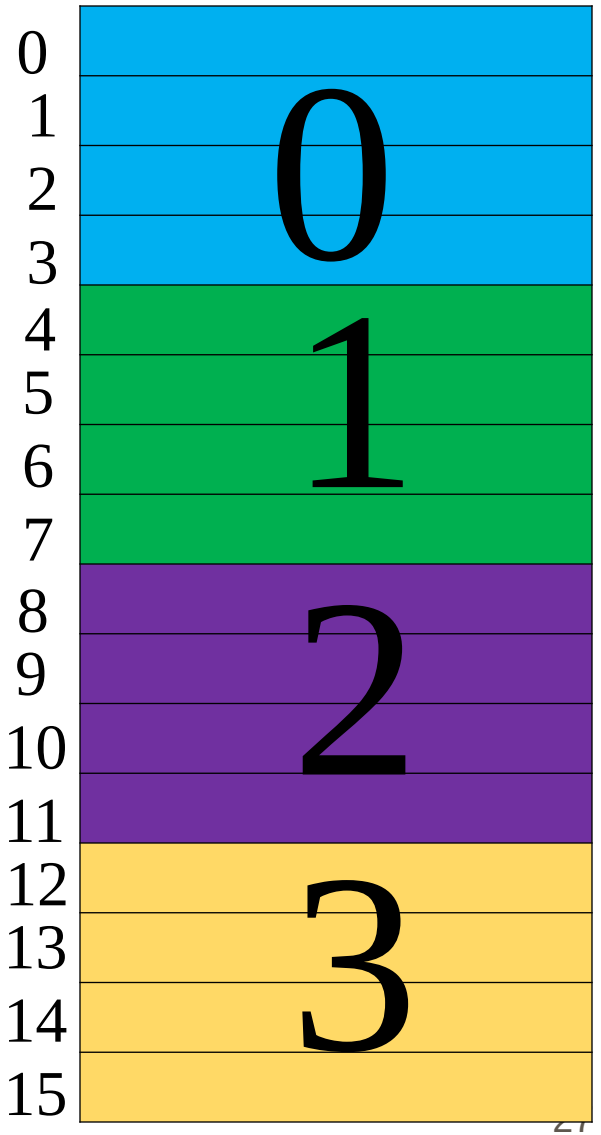
页表

给出了这个进程的每一页对应的页框的位置。在程序中,每个逻辑地址包含一个页号和在该页的偏移量。

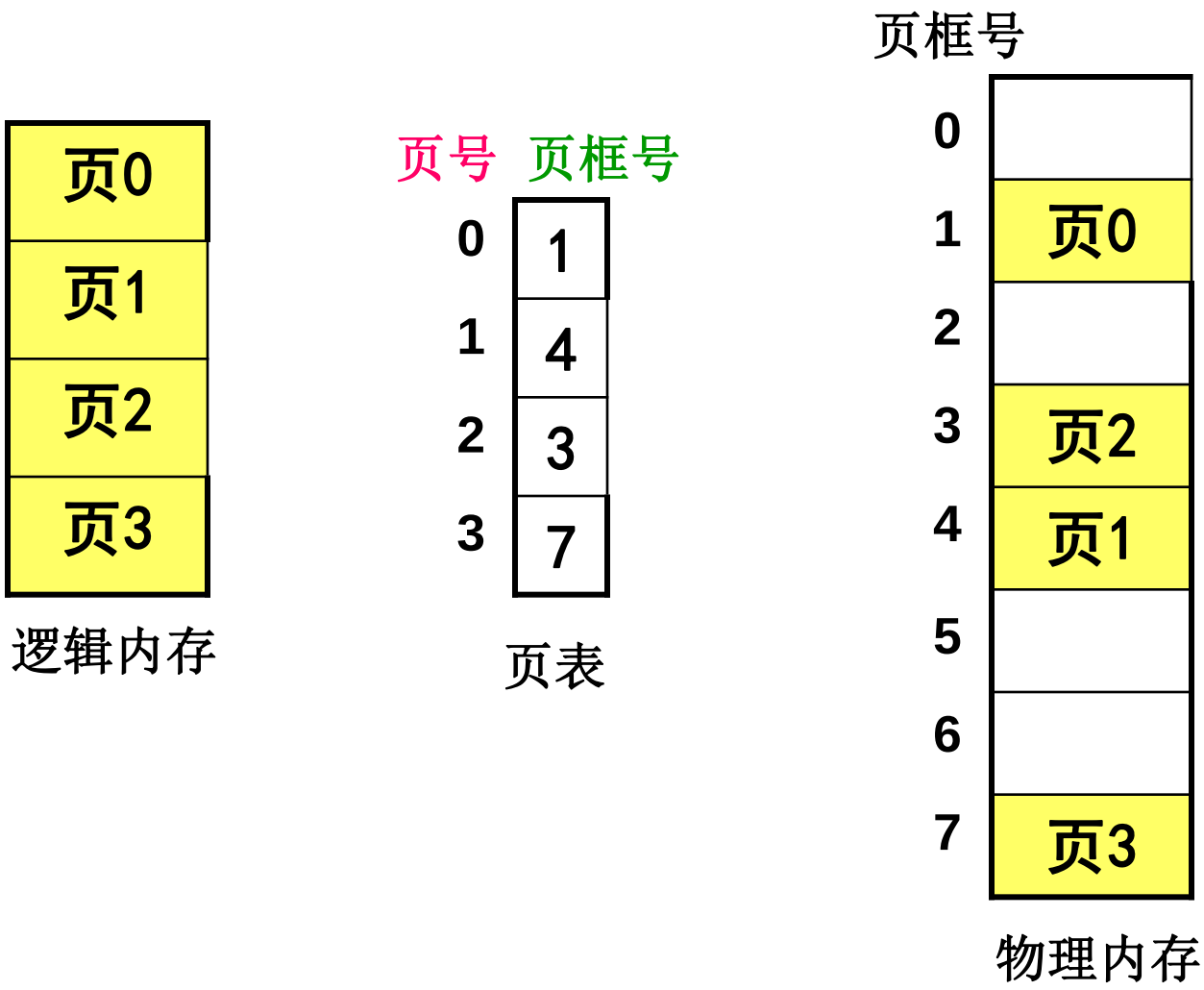
- 在简单分区的情况下,逻辑地址是一个字相对于程序开始处的位置,处理器把它转换成一个物理地址。
- 在分页中,逻辑地址到物理地址的转换仍然由处理器硬件完成,并且处理器必须知道如何访问当前进程的页表。

逻辑地址/页大小=页号….页内偏移

物理地址/页框大小=页框号….页内偏移



页表

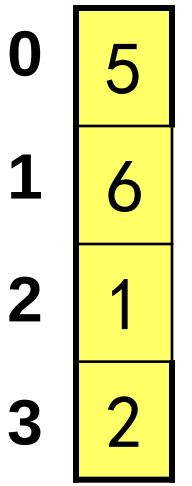

页0
页1
页2
页3
逻辑内存
页号
页框号
0
1
2
3
1
4
3
7
页表
页框号
0
1
2
3
4
5
6
7
页0
页2
页1
页3
物理内存

逻辑地址（页号，偏移量）

物理地址（页框号，偏移量） 转换

例：

- 说明：页大小为4B，页表如图所示，将逻辑地址0、3、4、13转换为相应物理地址
- 答案：20、23、24、9



页表

0/4=0…0　　5×4+0=20

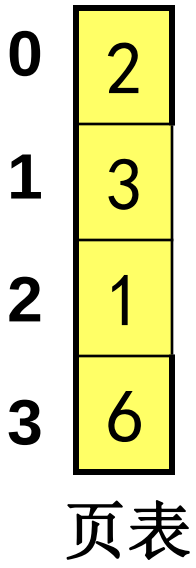
3/4=0…3　　5×4+3=23

4/4=1…0　　6×4+0=24

13/4=3…1　　2×4+1=9

练习：逻辑地址到物理地址的转换

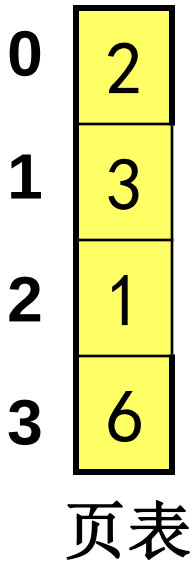
- 说明：页大小为1024B，页表如图所示，将逻辑地址1011、2148、3000、4000、5012转换为相应物理地址



练习：逻辑地址到物理地址的转换

- 说明：页大小为1024B，页表如图所示，将逻辑地址1011、2148、3000、4000、5012转换为相应物理地址
- 答案：3059、1124、1976、7072、逻辑地址非法

1011/1024=0…1011	2×1024+1011=3059
2148/1024=2…100	1×1024+100=1124
3000/1024=2…952	1×1024+952=1976
4000/1024=3…928	6×1024+928=7072
5012/1024=4…916	页号4不存在



页表

7.4 分段

- 把程序和其相关的数据划分到几个段中。
- 段有一个最大长度限制，但不要求所有程序的所有段的长度都相等。
- 分段与动态分区的区别
  - 一个程序可以占据多个分区
  - 一个程序占据的多个分区不需要是连续的
  - 会产生外部碎片，但跟动态分区比，会很小

逻辑地址到物理地址转换

- 段表：将逻辑地址映射为物理地址
- 段基地址：包含该段在内存中的开始物理地址
- 段界限：指定该段的长度
- 逻辑地址：段号s＋段内偏移d
- 逻辑地址到物理地址的转换
  1）段号与段表长度进行比较，若段号超过了段表长度，则越界（非法地址），否则转2）
  2）根据段表始址和段号计算出该段对应段表项的位置，从中读出该段在内存的起始地址，检查段内地址是否超过该段的段长，若超过则越界（非法地址），否则转3）
  3）将该段的起始地址与段内位移相加，从而得到要访问的物理地址

✓逻辑地址到物理地址转换例

- 说明：段表如表1所示，将表2所示逻辑地址转换为相应物理地址
- 答案：见表3

段号	内存起始地址	段长
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

表1

段号	段内位移
2	88
4	100

表2

178	90+88
非法	100>96

表3

✓逻辑地址到物理地址转换练习

- 说明：段表如表1所示，将表2所示逻辑地址转换为相应物理地址
- 答案：见表3

段号	内存起始地址	段长
0	210	500
1	2350	20
2	100	90
3	1350	590
4	1938	95

表1

段号	段内位移
0	430
1	10
2	500
3	400
4	112
5	32

表2

640	210+430
2360	2350+10
非法	500>90
1750	1350+400
非法	112>95
非法	段号5不存在

表3

作业

- 复习题 7.2， 7.8
- 习题 7.6， 7.7a， 7.14
- 补充习题：页式存储管理系统中，某进程页表如下。已知页面大小为1024字节，问逻辑地址600，2700，4000所对应的物理地址各是多少？

页号	页框号
0	7
1	3
2	5

- 1110　1001001001
- 页大小 1024
- 页号?
- 页内偏移?

- 14-→ 3
- 0011 1001001001