第 4 章 記憶體管理

本章概要

- 單一分割區的配置
- 重定址暫存器(Relocation Reg.)
- 多個行程時的記憶體配置-固定長度
- 變動長度的多重分割區

單一分割區的配置

- 將記憶體空間分成兩塊，一塊給作業系統使用，另一塊則給行程使用
- 可以利用硬體的基底暫存器來避免行程存取到作業系統的記憶體位址
- 如果程式所產生的位址小於基底暫存器中的位址，則會產生定址錯誤的例外中斷

圖4-1.a 合法的邏輯位址

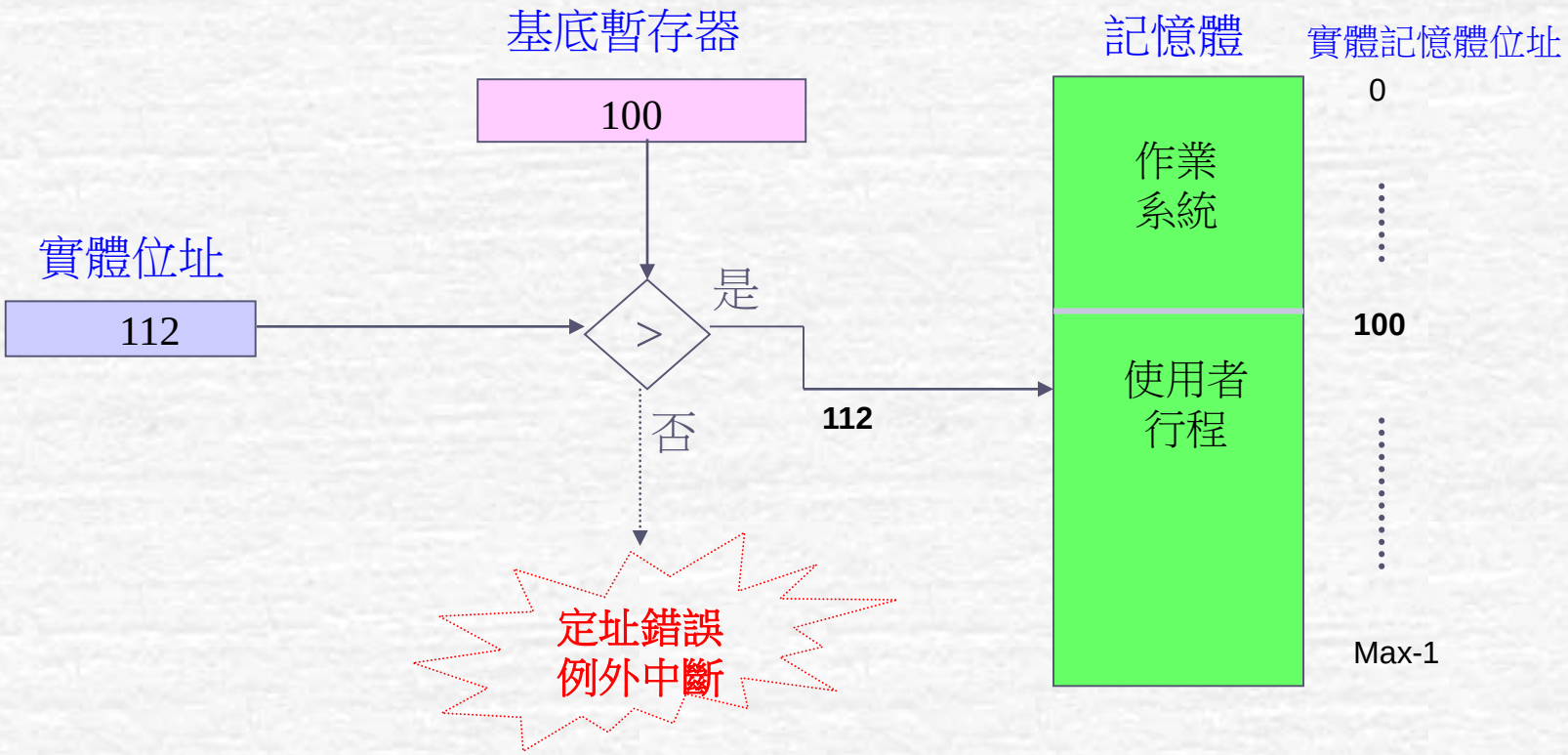
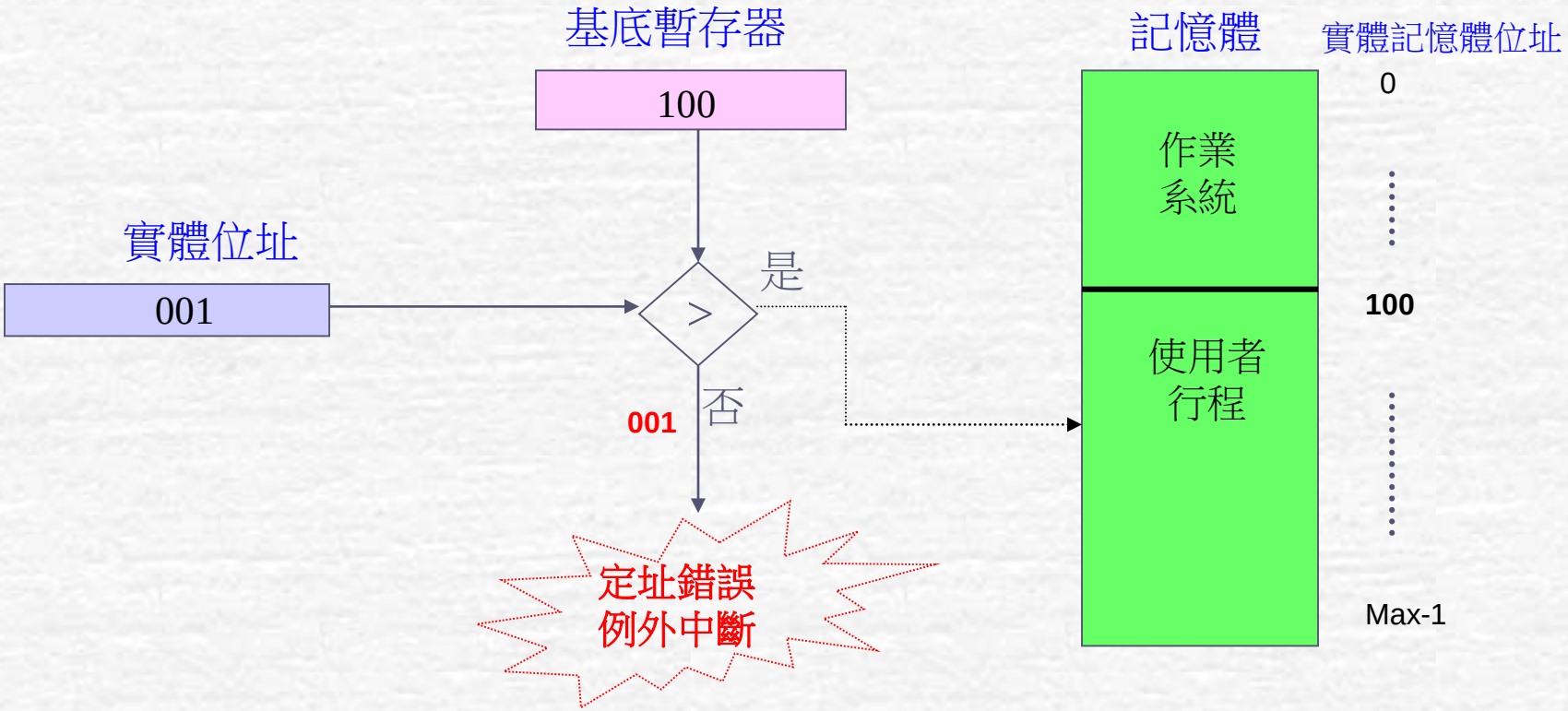


圖4-1.b 存取到作業系統的記憶體位址

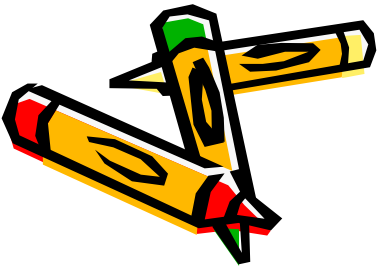


課堂練習

- 假設作業系統是位於實體記憶體由0開始的連續區段之中，且長度為256KB
  - 請問基底暫存器的值至少應該設為多少？
  - 請問下列位址何者為合法的行程位址？
    1. 300K
    2. 25K
    3. 300

練習解答

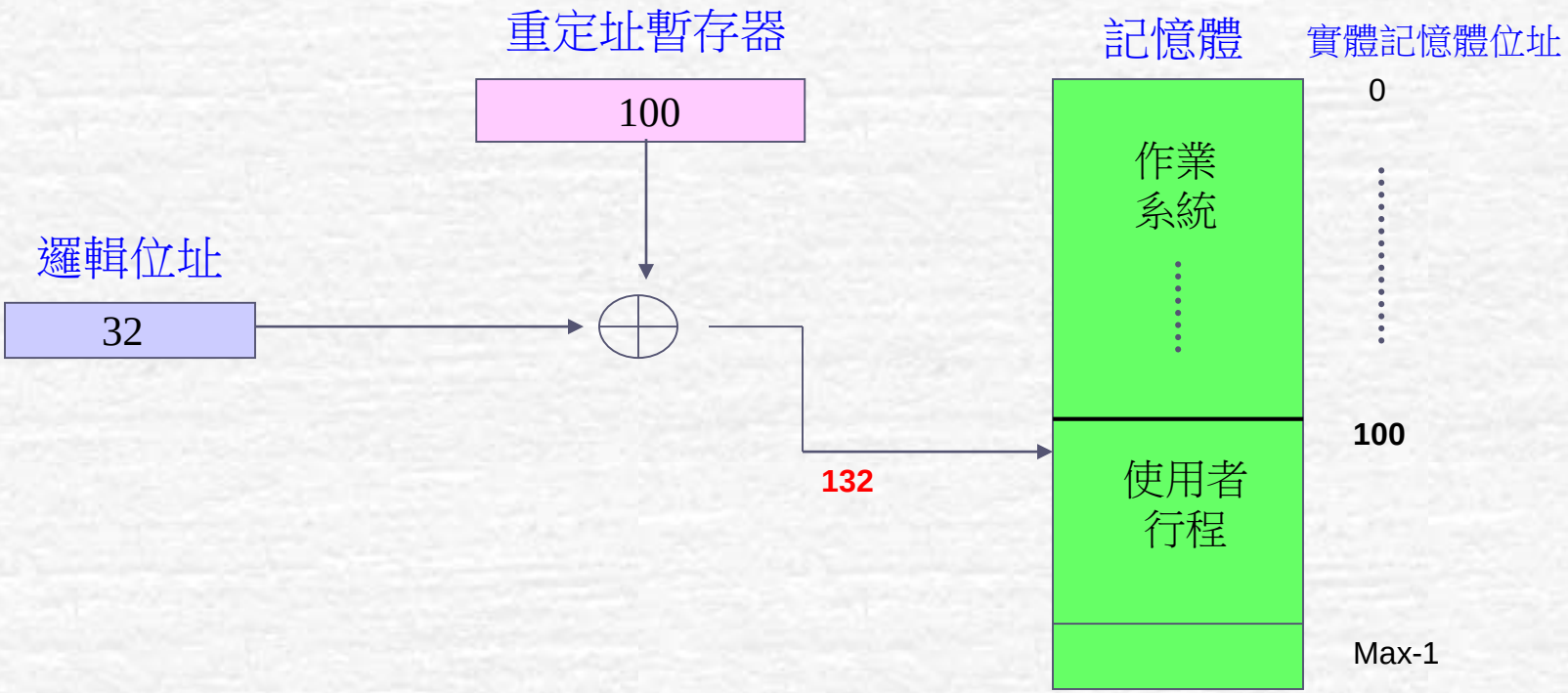
- 基底暫存器的值：大於等於256K
- 行程：
  1. 300K-合法
  2. 25K-不合法
  3. 600-不合法



重定址暫存器(Relocation Reg.)

- 另一種保護的方式是使用重定址暫存器，用來存放行程的起始位址
- 在編譯的時候，將程式從0開始配置
  - 邏輯位址：程式好像是運作在一個從0開始的一大片連續的邏輯位址空間中
- 在執行的時候
  - 實體位址：由記憶體管理硬體負責將邏輯位址轉換為在實體記憶體中的真正位址

圖4-2 利用重定址暫存器的記憶體位址轉換



程式的載入

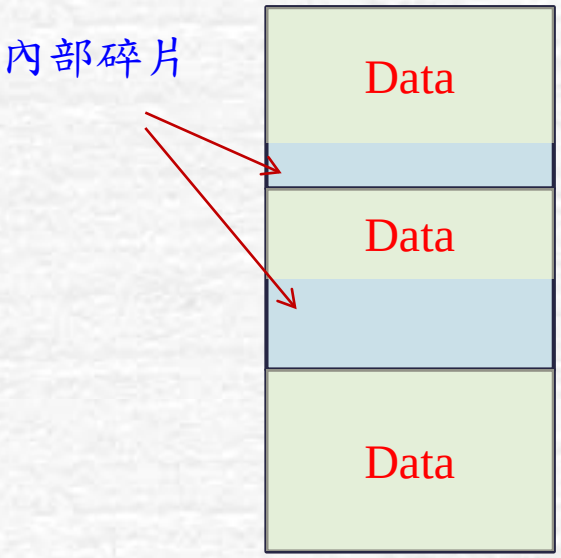
- 絕對程式碼：在程式編譯時預先決定這支程式要放在實體記憶體的哪個位置執行，則編譯出來的程式碼邏輯位址會等於實體位址
  - 缺點在於程式碼只能放在記憶體的固定位置，當多個行程要共享記憶體時，就可能彼此產生干擾
- 可重定址程式碼：在編譯時是從0開始配置位址，然後在載入時才決定要放在哪個位置
  - 必須透過邏輯位址加上重定址暫存器的內容，才能取得行程的實體位址
  - 如果需要改變行程在實體記憶體中的位置時，只要在搬移行程之後改變重定址暫存器的值就可以了

多個行程時的記憶體配置-固定長度

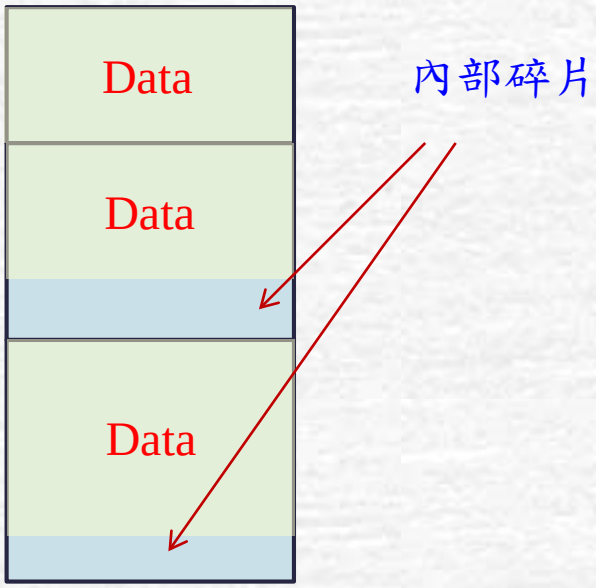
- 將記憶體切割為相同大小的分割區，作業系統只需要記住每個行程分配到哪個分割區就可以了--最簡單的多行程配置方式
- 因為分割區的長度都相同，所以在開機的時候就可以設定長度暫存器的值
- 然後在某個行程取得CPU控制權的時候，作業系統就將重定址暫存器的內容重新設定到該行程所在的分割區
- 如果將記憶體分割成相同大小的分割區，有些區域內就會剩下很多空間，稱為內部碎片
- 可以將記憶體分割為大小不等的數個固定長度分割區，並且將不同大小的行程配置到不同長度的分割區中
- 當輪到某個行程使用CPU時，除了要設定重定址暫存器之外，作業系統還必須將長度暫存器的值設定為該行程所在分割區的長度

內部碎片

相同大小的分割區



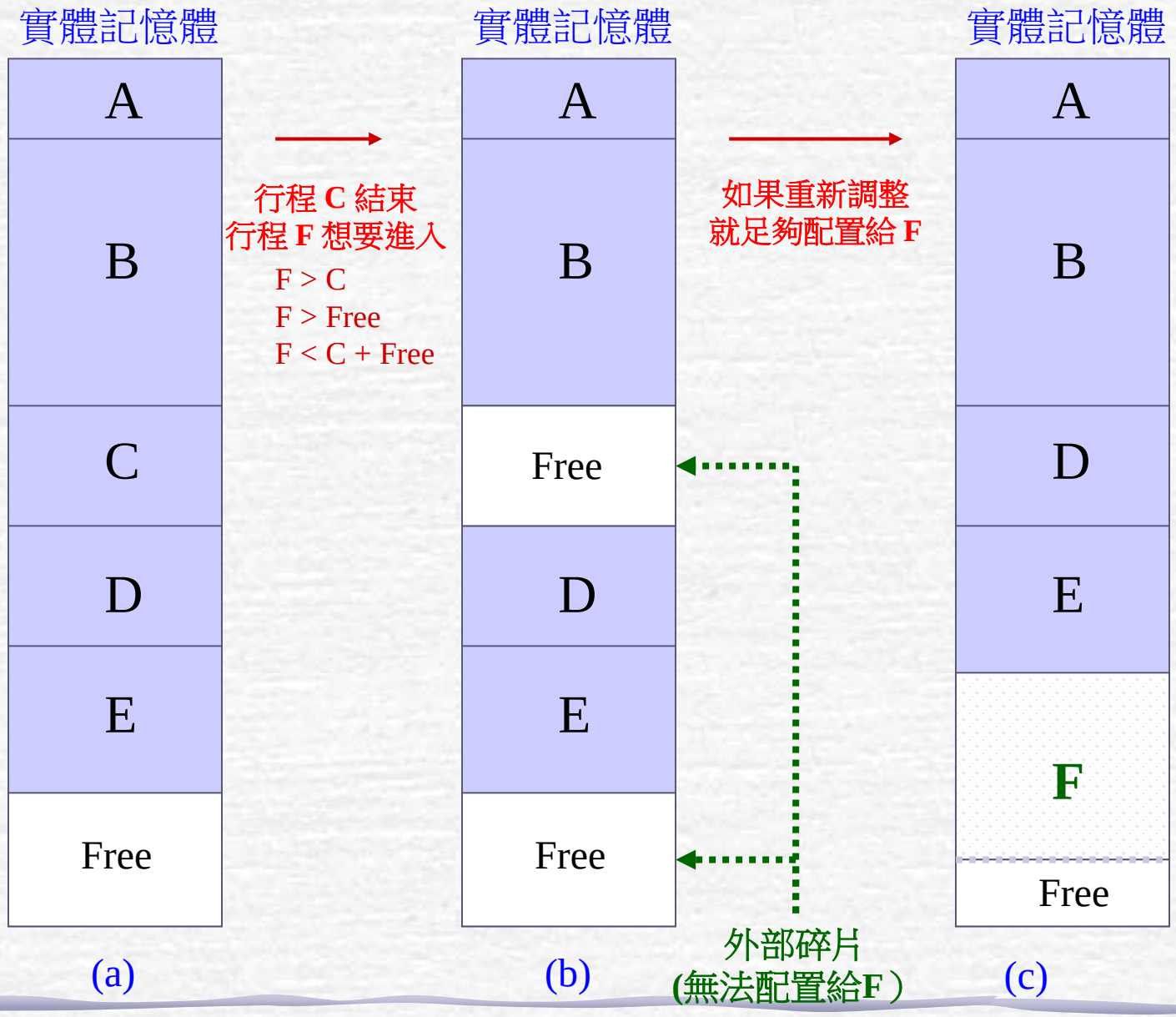
大小不等的數個固定長度分割區



變動長度的多重分割區

- 另一種減少內部碎片的方法，就是不要事先分割記憶體，而是根據目前的記憶體使用情況，與即將要載入的行程長度，來動態分割記憶體
- 可以將內部碎片降至最低，但是卻可能造成外部碎片
- 如果分割區配置的選擇不當，就可能會加重外部碎片的問題
- 可利用聚集法重新搬移各個行程，將這些碎片聚集成為完整的一大塊連續空間

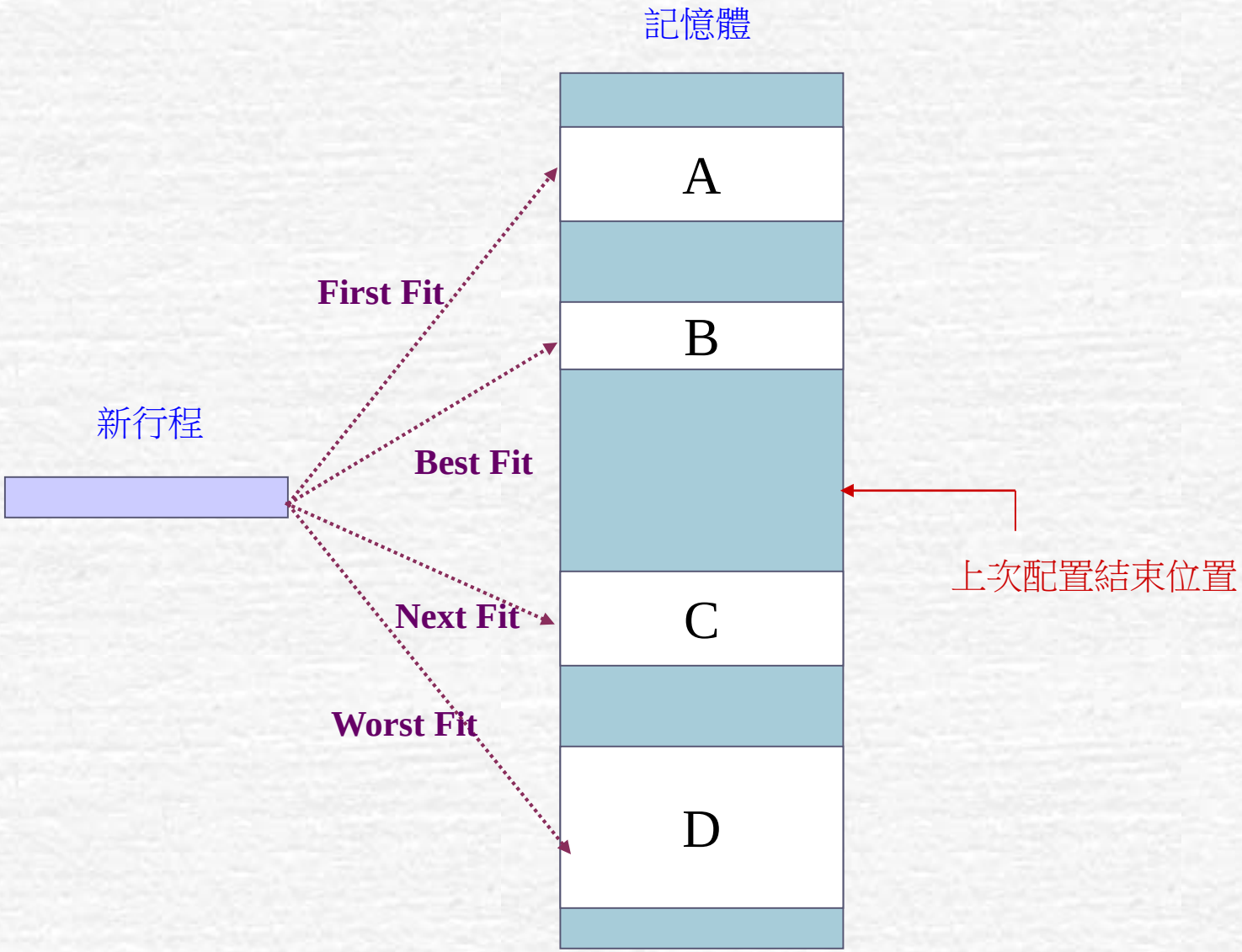
圖4-3 外部碎片範例


實體記憶體
A
B
C
D
E
Free
(a)
行程 C 結束
行程 F 想要進入
F > C
F > Free
F < C + Free
實體記憶體
A
B
Free
D
E
Free
(b)
如果重新調整
就足夠配置給 F
外部碎片
(無法配置給F)
實體記憶體
A
B
D
E
F
Free
(c)

分割區的選擇演算法

- 先適法：從記憶體的開頭開始搜尋閒置的記憶體空洞，然後將行程放入第一個可以容納的洞中
- 次適法：從上次搜尋結束的位置往下繼續搜尋，以改善搜尋的時間
- 最適法：從所有的閒置區域中找出長度大於這個行程，而且最接近這個行程的洞
- 最不適法：將最大的洞分配給行程

圖4-4 分割區的選擇演算法範例

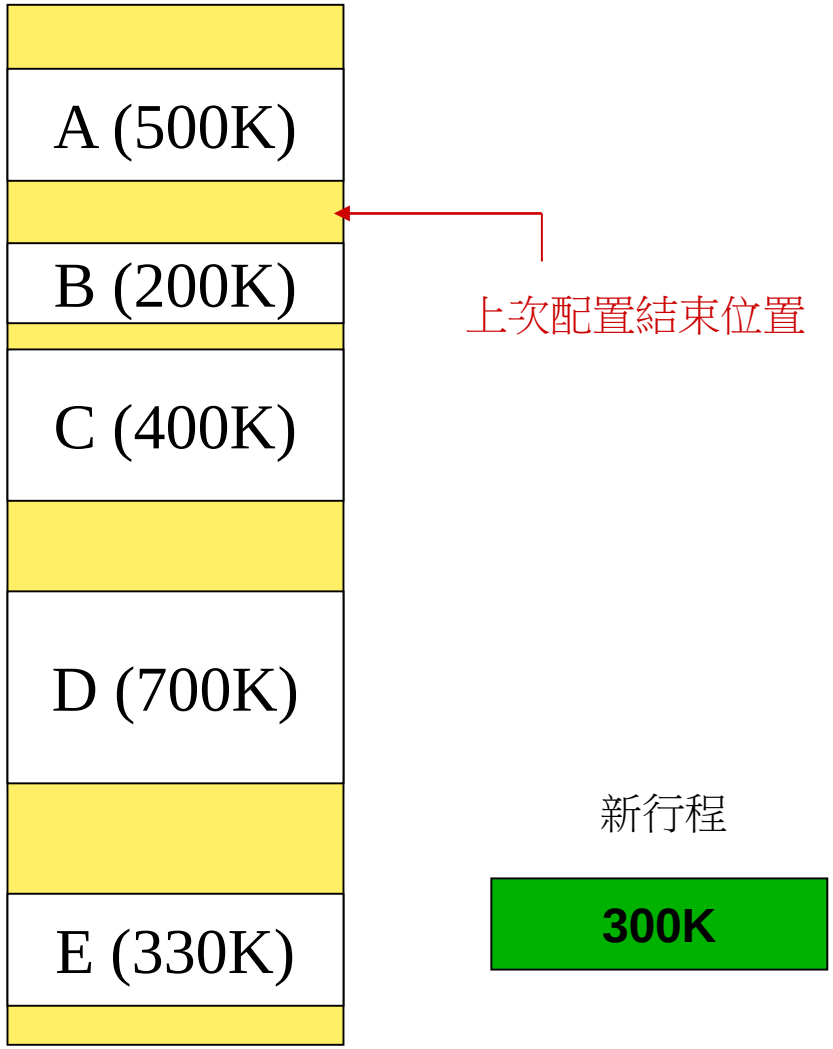


課堂練習

範例補充包-Extra~

Q：假設記憶體目前的配置狀況如圖，白色區域為尚未使用的部份。請分別指出使用先適法、次適法、最適法、與最不適法時：

1. 新行程會被分配到哪個區域？
2. 所產生的內部碎片大小為何？



解答

演算法	選擇區域	內部碎片
先適法	A	200K
次適法	C	100K
最適法	E	30K
最不適法	D	400K