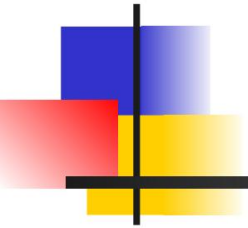
操作系统原理

操作系统原理

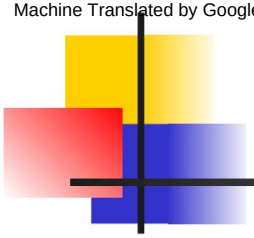


内存管理

李旭东

leexudong@nankai.edu.cn南

开大学



目标

没有内存抽象

基本内存管理

~~虚拟的~~ ~~记忆~~ ~~管理~~

帕金森定律

程序会扩展以填满可用来容纳它们的内存！

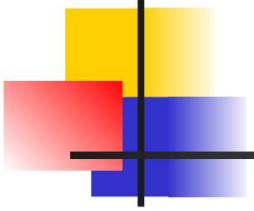
内存管理

内存是一种重要的资源,必须谨慎管理

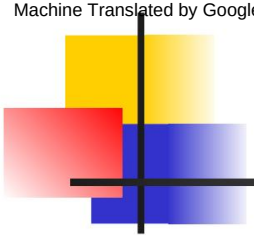
虽然如今普通家用计算机的内存是 IBM 7094（20 世纪 60 年代初期世界上最大的计算机)的 1000 倍

内存层次

- 易挥发的 高速缓存
  - 数量少,速度快,价格贵
- 易失性主存储器（RAM）
  - 几十兆,中速,中等价格
- 非易失性磁盘存储
  - 几十或几百 GB,速度慢,便宜



内存层次结构



内存管理

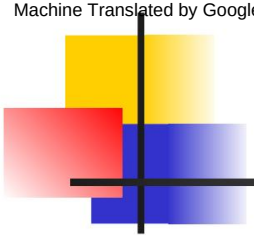
1.扩展主内存

2.控制主存与存储之间的数据传输

3. 主内存的分配与撤销

4. 主内存共享与保护

没有内存抽象



没有内存抽象

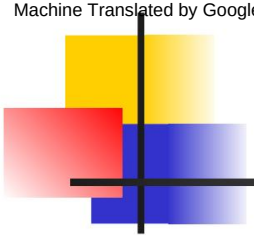
早期大型计算机（<1960 年）

早期小型计算机（<1970 年）

早期个人电脑（<1980 年）

RAM（随机存取存储器）、ROM（只读存储器）

使用操作系统组织内存的三种简单方法

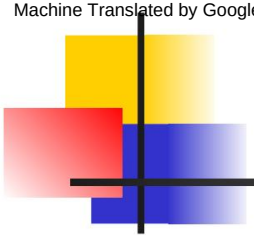


没有内存抽象的问题

两个过程

没有内存抽象的问题

?



没有内存抽象的问题

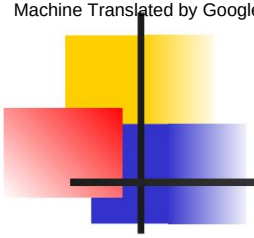
1. 独立的地址空间

基址寄存器
限制寄存器

JMP 16412

没有内存抽象的问题

2. 搬迁问题



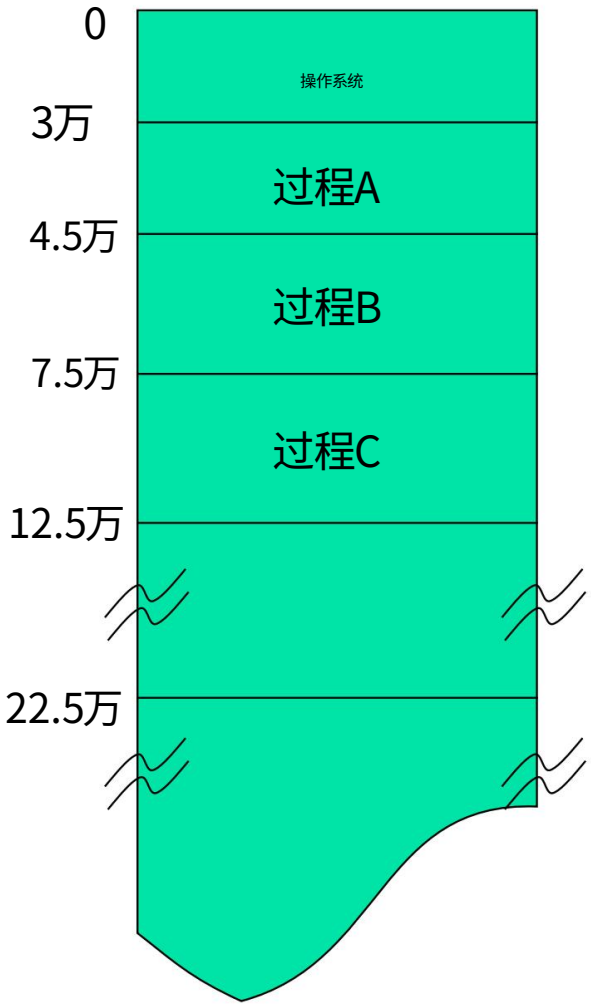
基本内存管理 - 连续分配

固定分区的多道编程

分区固定大小

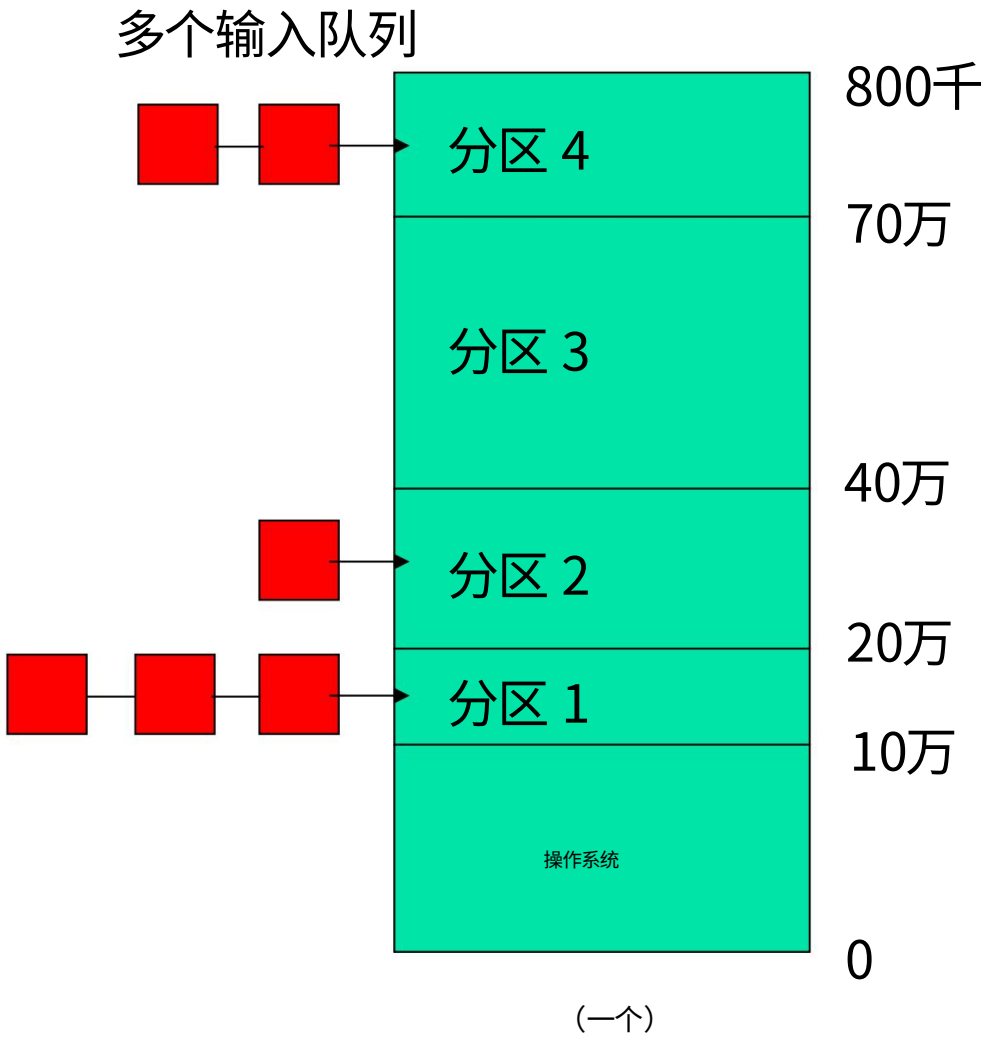
内径	尺寸（知识库）	开始地址（金）	状态
1	15 30		用过的
2	30 45		用过的
3	50 75		用过的
4	100 125	可用	

(a)分区描述表



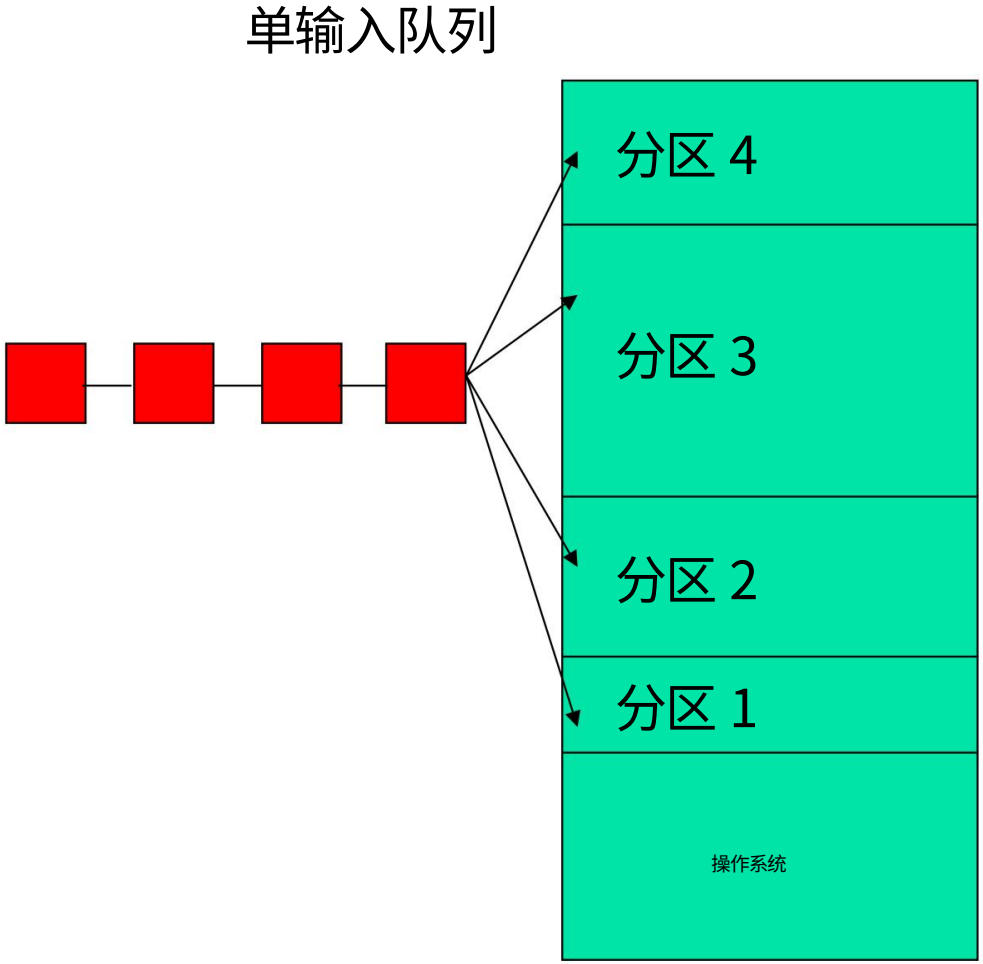
(b)内存布局

固定分区的多道编程



（一个）

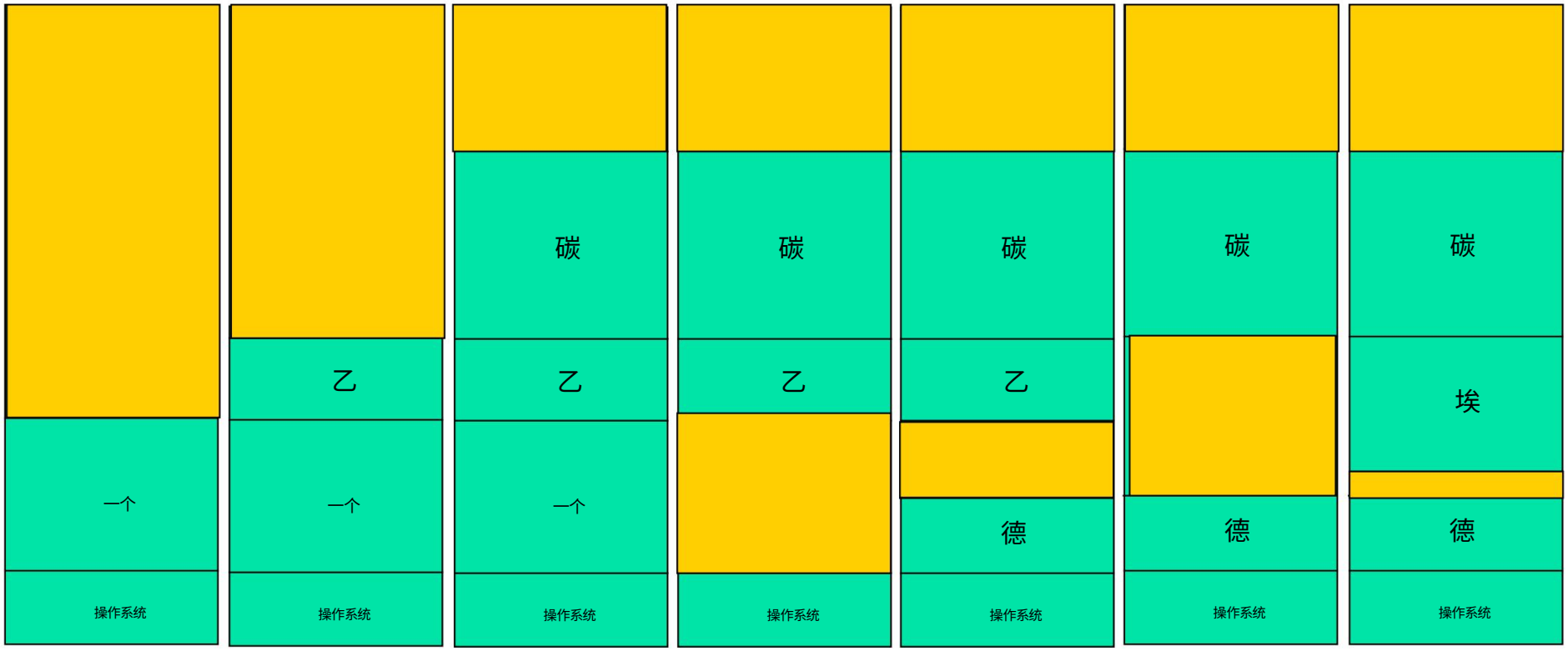
固定分区的多道编程

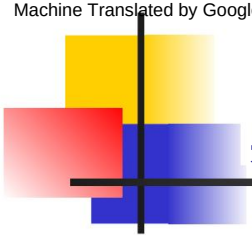


(二)

具有可变分区的多道编程

·动态分配连续分区





具有可变分区的多道编程

结构

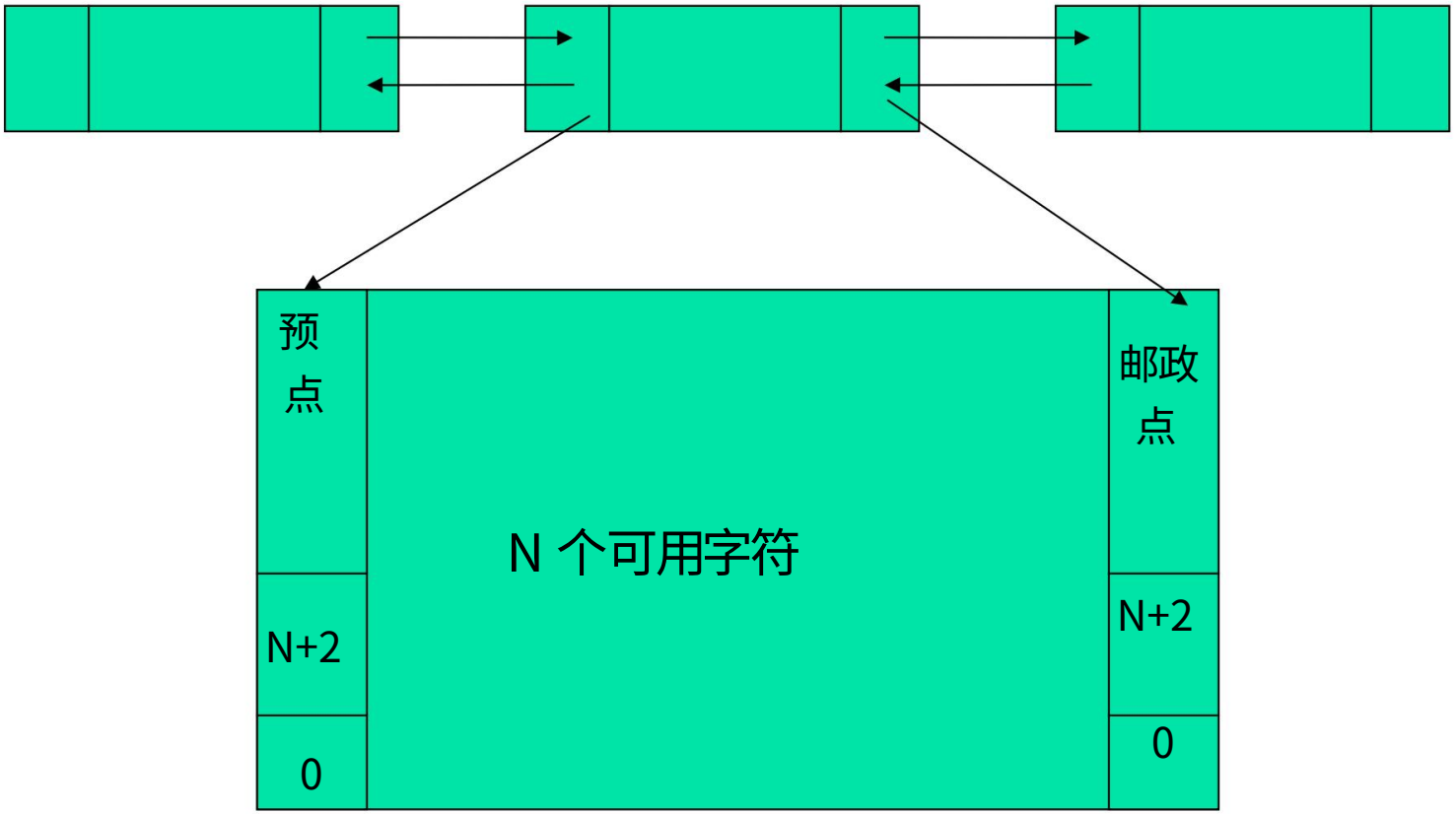
算法分配和撤

销程序

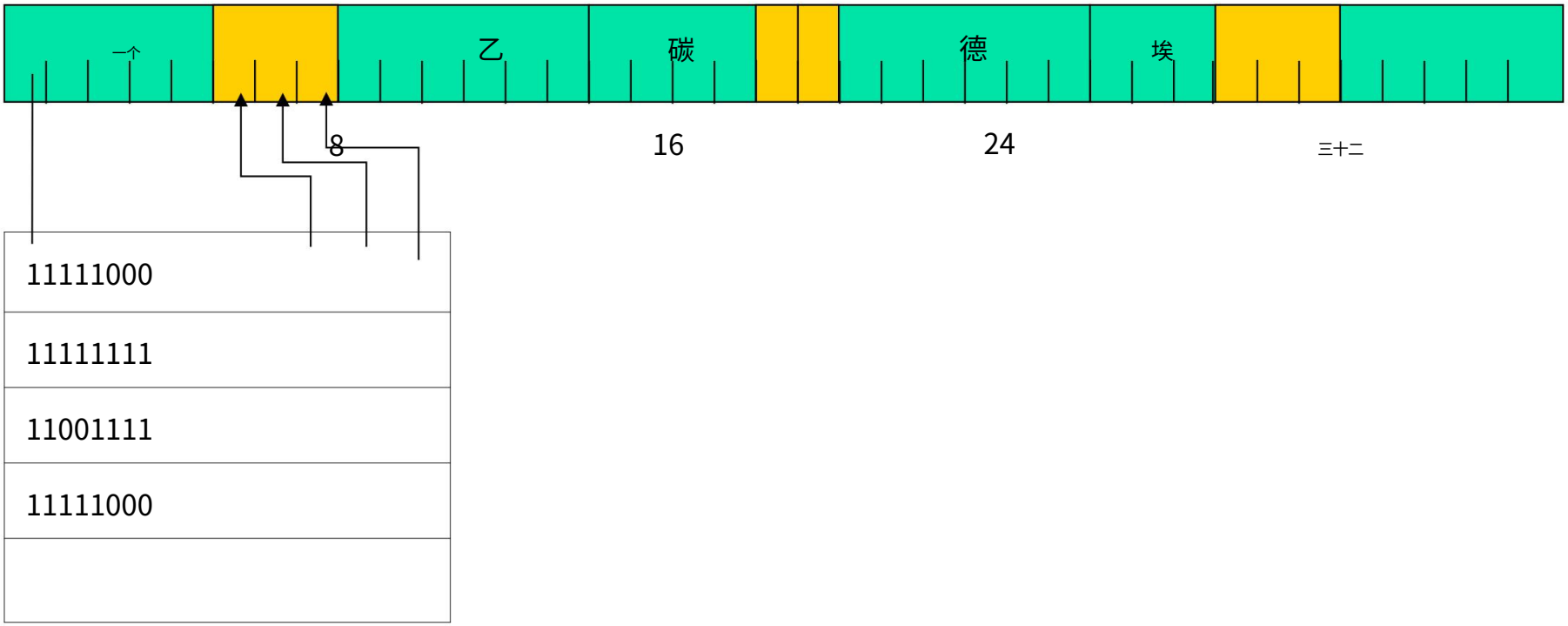
变量分区:分区阵列表

ID	尺寸 （知识库）	开始 地址 （金）	状态
1	64	四十四	可用的
2	24	132	可用的
3	40	210	用过的
4	三十	270	可用的
5	…	……	

变量分区:内联链接结构



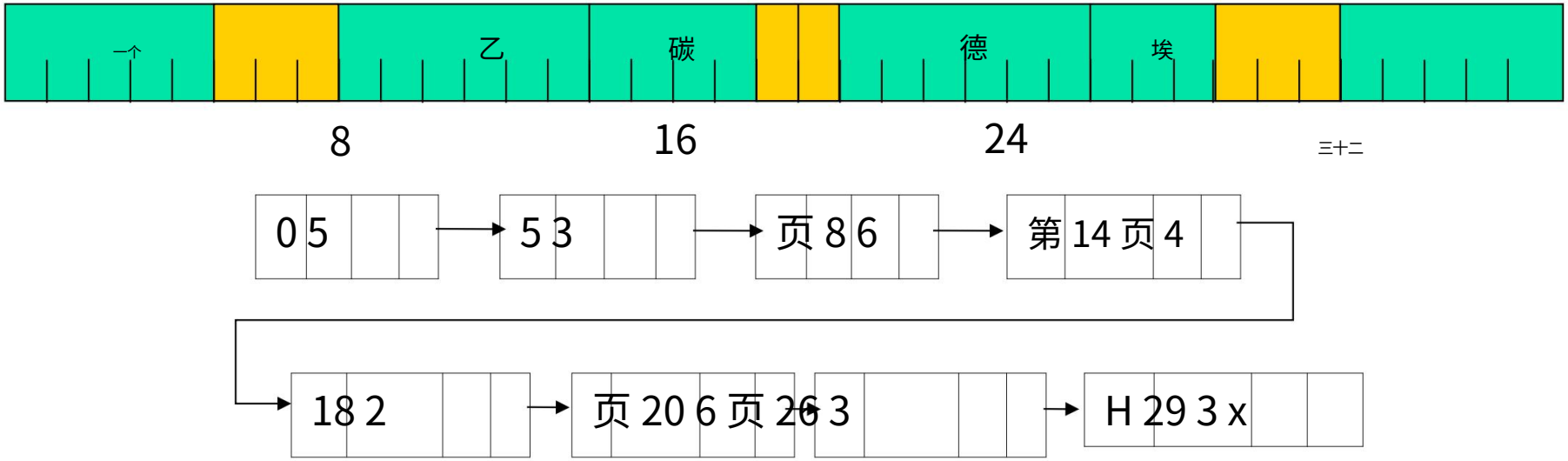
可变分区:位图位图



变量分区:链表

P:过程

H:自由空间



变量划分算法

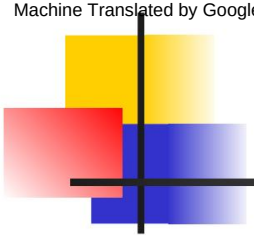
首次适配:FF

下一个 FF

最佳匹配:BF

- 最佳适应算法
- 外部碎片 · 碎片

最差拟合:WF



变量划分算法

快速安装

– 4KB、8KB、16KB 空闲连续空间的多队列

– 优势

– 缺点

·合并空闲分区的开销

变量划分算法

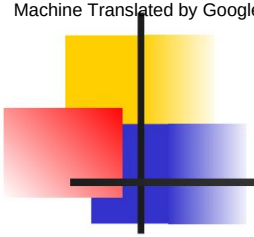
伙伴内存分配 – 伙伴式的内存管理

– 1963年,哈里·马科维茨,1990年诺贝尔经济学奖得主

– 每个区块又细分为两个较小的区块

– 2我

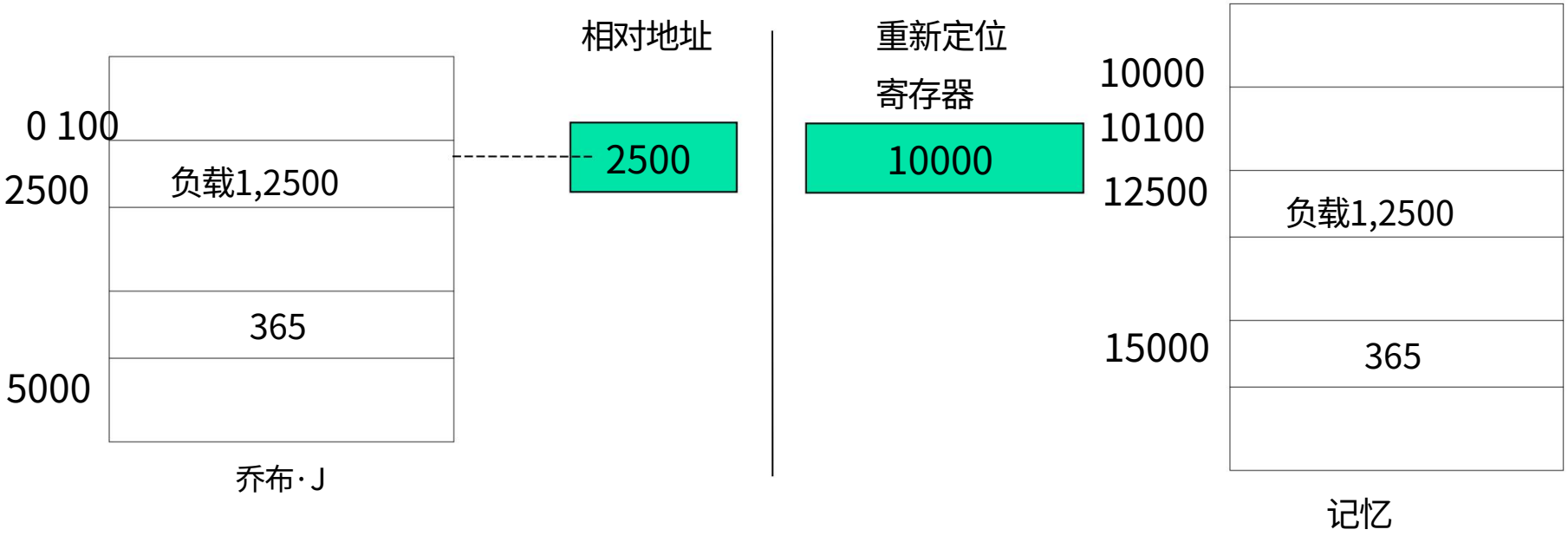
– 内部分裂



案例:伙伴内存分配

2 4 ->23 ->22 ->21 ->20 ->A ->C ->B ->

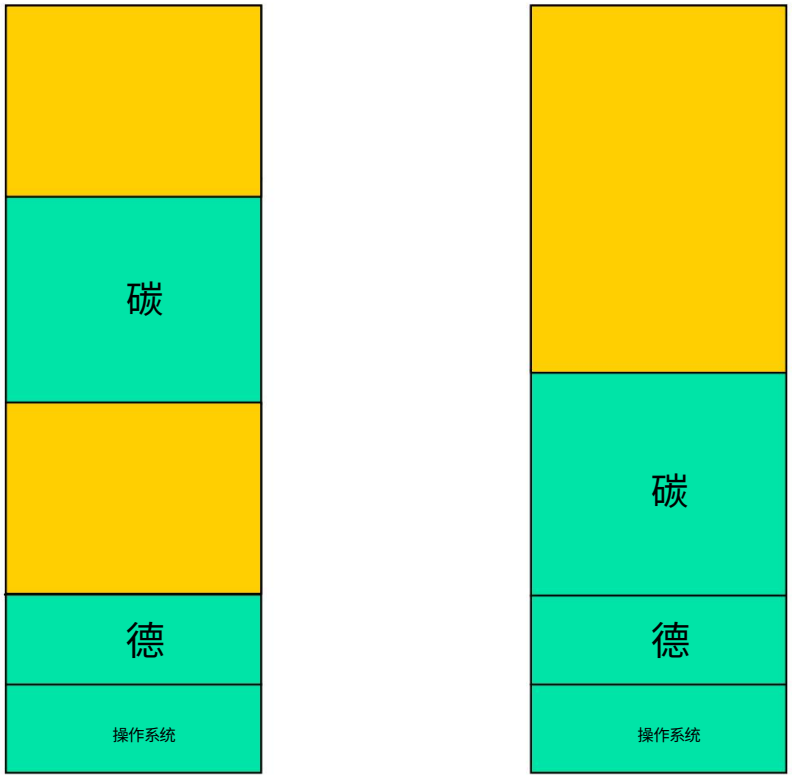
动态重定位



内存压缩

技术

?碎片化

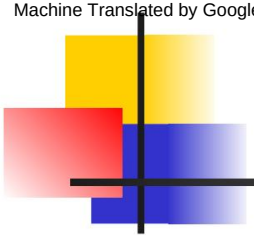


(六)

补偿前。

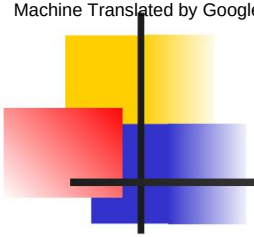
(f*)

压缩后。



Swapping 交换

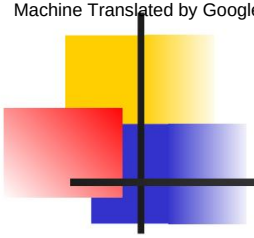
将每个进程全部导入,运行一段时间,然后将其放回磁盘



Overlay 覆盖

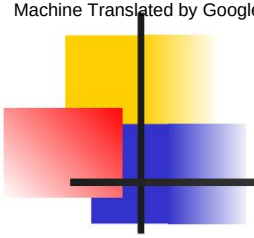
将一个存储的指令或数据块替换为另一个

一种描述将作为单个内存映像的一部分加载但将在同一内存地址运行的部分的方法

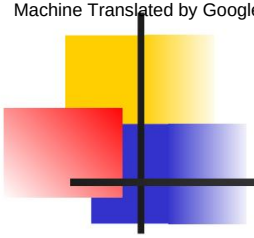


变量分区问题

(a)为不断增长的数据段分配空间。 (b)为不断增长的堆栈、不断增长的数据段分配空间。



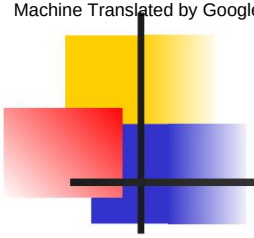
基本内存管理 - 离散分配



离散内存分配

分段分段

分页分页



分段

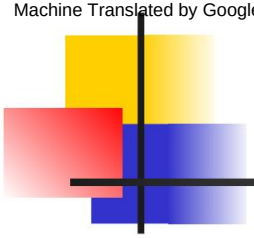
二元组:<段号,偏移量>

·逻辑分段

- 1. 代码
- 2.全局变量
- 3. 堆,用于分配内存
- 4. 使用的堆栈
  每个线程
- 5. 标准 C 库

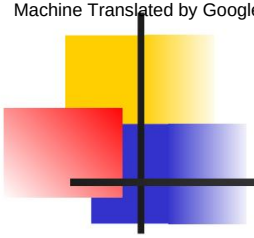
?碎片化

程序员对程序的看法



分割

分段硬件



分页

·页面·
框架 ·页表

逻辑和物理内存的分页模型

分页:页表

地址=（页码,偏移量）

一个元组

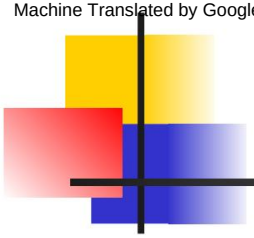
?碎片化

页码=INT [A　天]

偏移量=[A] MOD L

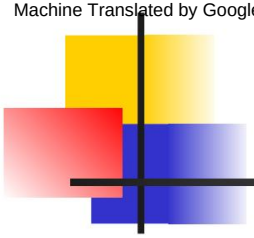
A :逻辑地址

L:页面大小



分页:硬件

·页表寄存器： PTR ·两次访问内存



分页:带有 TLB 的硬件

硬件缓存： ·翻译查看

预留缓冲区 (TLB)

·键（或标签)和值

·快表* ，

- “ 联想存储器 ”

联想记忆

快表

在操作系统中,为了提高系统的访问速度,在地址映射机制中增加一个小容量的映射寄存器,即快表,用来存放当前访问最频繁的少数活动页面的页号。　　当某用户需要访问数据时,根据数据所在的逻辑页号在快表中找到其对应的内存块号,再联系页内地址,物理地地址。

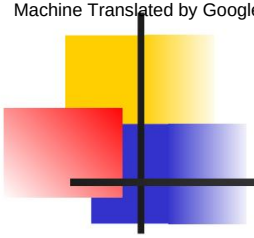
如果在快表中没有相应的逻辑页号,则地址映射仍然可以通过内存中的页表进行,得到空闲块号后须将块号填入快表的空闲区中。

如果快表中没有空闲块,则根据淘汰算法淘汰出所有行,再填入新的页号和块号。　　快表中查找内存块的物理地址消耗的时间大大降低了,使得系系统效率得到了极大的提高。

CPU 中的缓存

典型案例是高速缓冲存储Cache随着计算机硬件的发展,CPU的执行速度越来越快,系统架构越来越先进,而主存的结构和访问速度提高则较慢,因此高速存储技术将越来越重要。

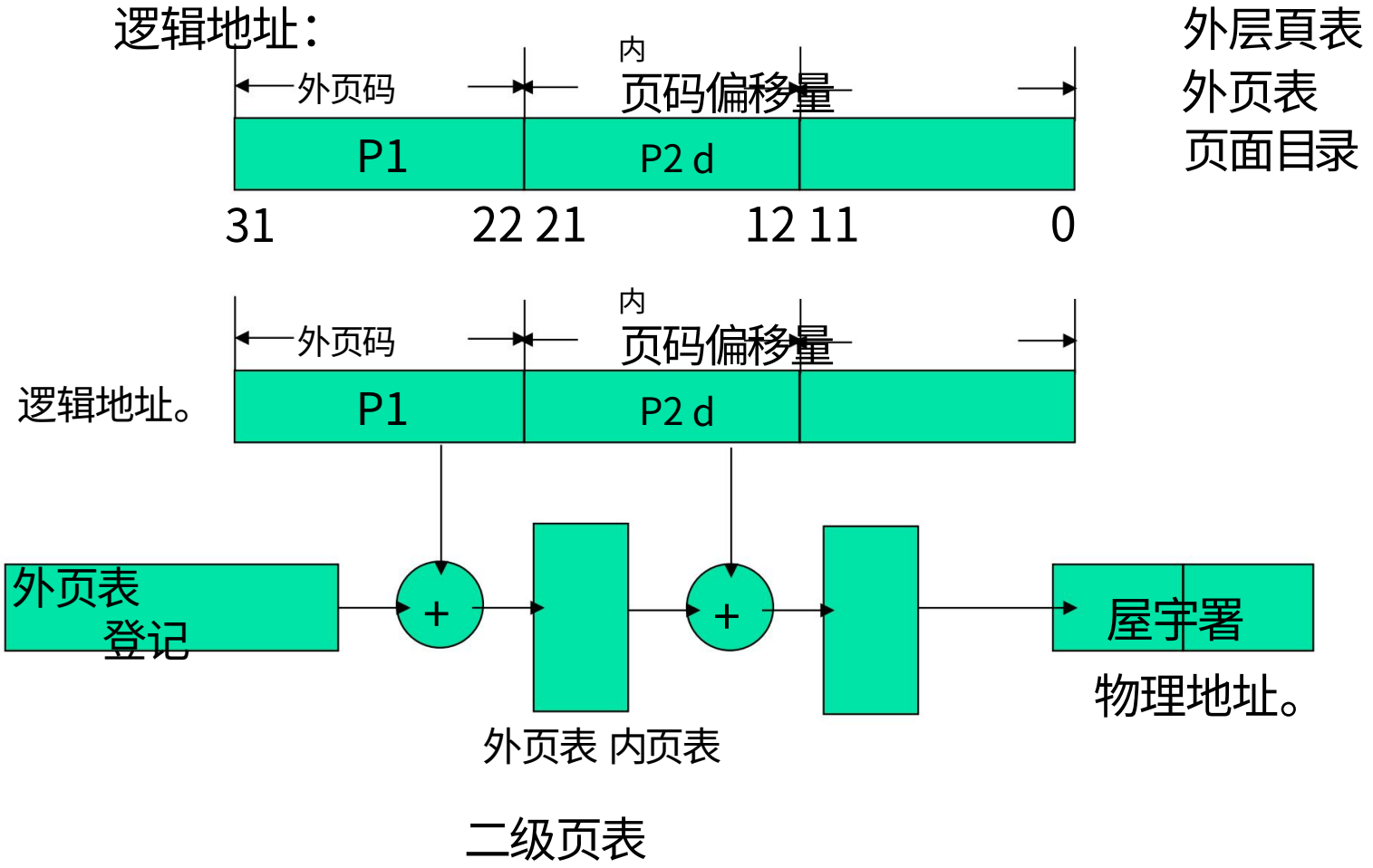
高速缓冲存储器 Cache 是位于 CPU 与内存之间的临时存储器,它的容量比内存小但交换速度快。在 Cache 中的是数据内存中的一小部分,但这小部分是瞬间 CPU即将访问的。当CPU调使用大量数据时,就可以察觉内存直接从Cache中调使用,从而加速读取。



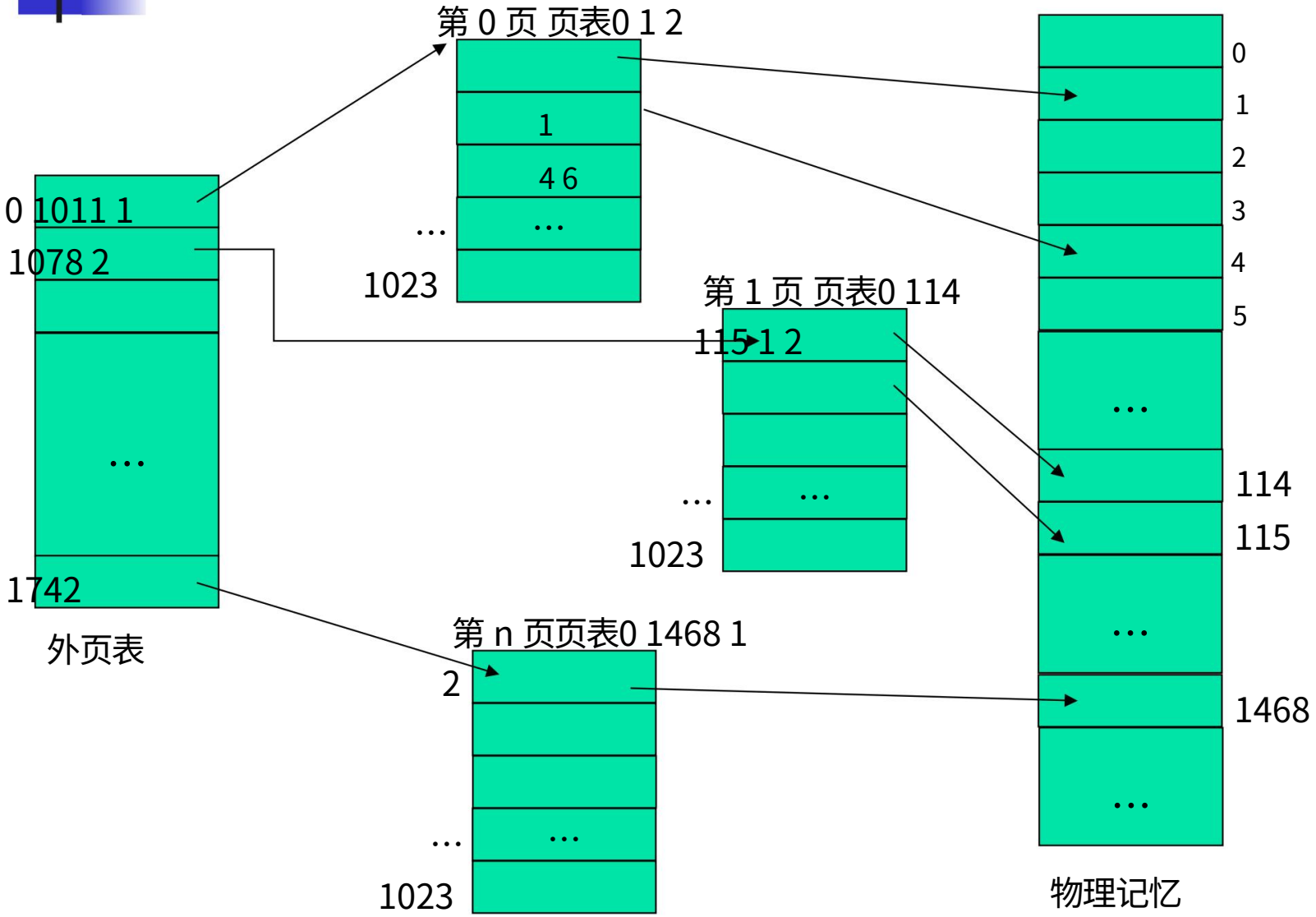
分页:带有 TLB 的硬件

用于加速分页的 TLB

分页:多级页表

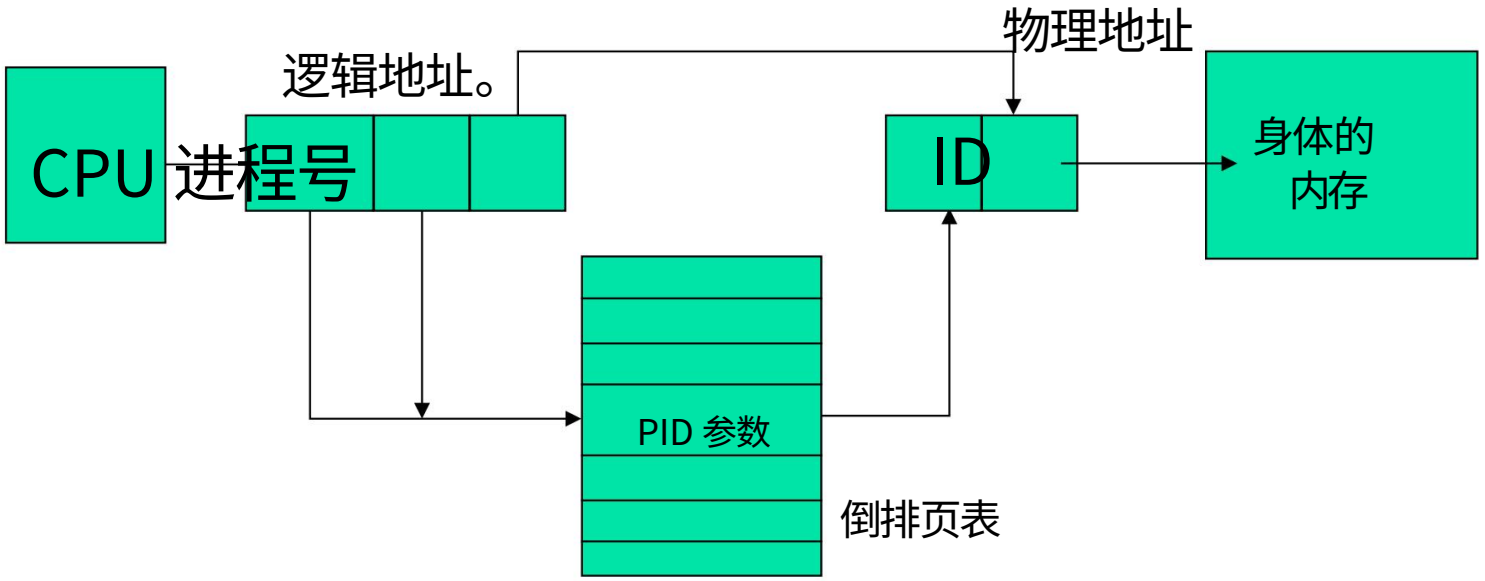


分页:多级页表



倒排页表

<进程 ID,页码,偏移量>

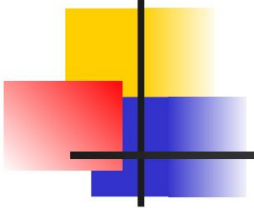


倒排页表

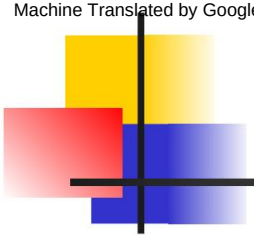
64位CP

4kb/页

传统页表与倒置页表的比较

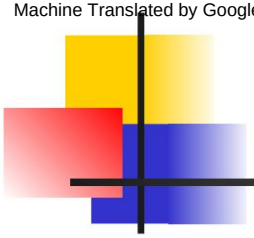


哈希页表

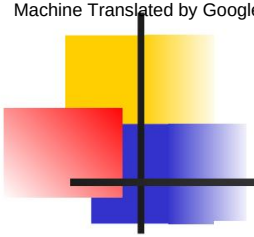


当前记忆翻译

逻辑地址 -> （虚拟/)线性地址 -> 物理地址



当前记忆翻译



概括

内存分区交换分段分页

问答?