虚拟内存

虚拟内存的目的是为了让物理内存扩展成更大的逻辑内存，从而让程序获得更多的可用内存；

为了更好的管理内存，操作系统将内存抽象成地址空间。每个程序拥有自己的地址空间，这个地址空间被分割成多个块，每一块成为一个页，这些页被映射到物理内存，但不需要映射到连续的物理内存，也不需要所有页都必须在物理内存中。当程序引用到不存在的物理内存中的页时，由硬件执行必要的映射，将缺失的部分装入物理内存，并重新执行失败的指令。

虚拟内存允许程序不用将地址空间中的每一页都映射到物理内存，也就是说一个程序不需要全部调入内存就可以运行，这使得有限的内存运行大程序成为可能。例如有一台计算机可以产生 16 位地址，那么一个程序的地址空间范围是 0~64K。该计算机只有 32KB 的物理内存，虚拟内存技术允许该计算机运行一个 64K 大小的程序。

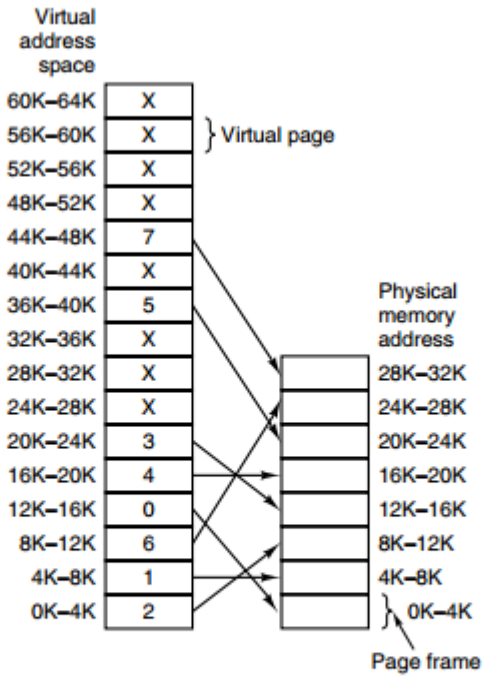


Figure 3-9. The relation between virtual addresses and physical memory addresses is given by the **page table**. Every page begins on a multiple of 4096 and ends 4095 addresses higher, so 4K–8K really means 4096–8191 and 8K to 12K means 8192–12287.

分页系统地址映射

内存管理单元MMU管理着地址空间和物理内存的转换，其中的页表存储着页（程序地址空间）和页框（物理内存空间）的映射表。

一个虚拟地址分为两部分，一部分存储页面号，一部分存储偏移量；

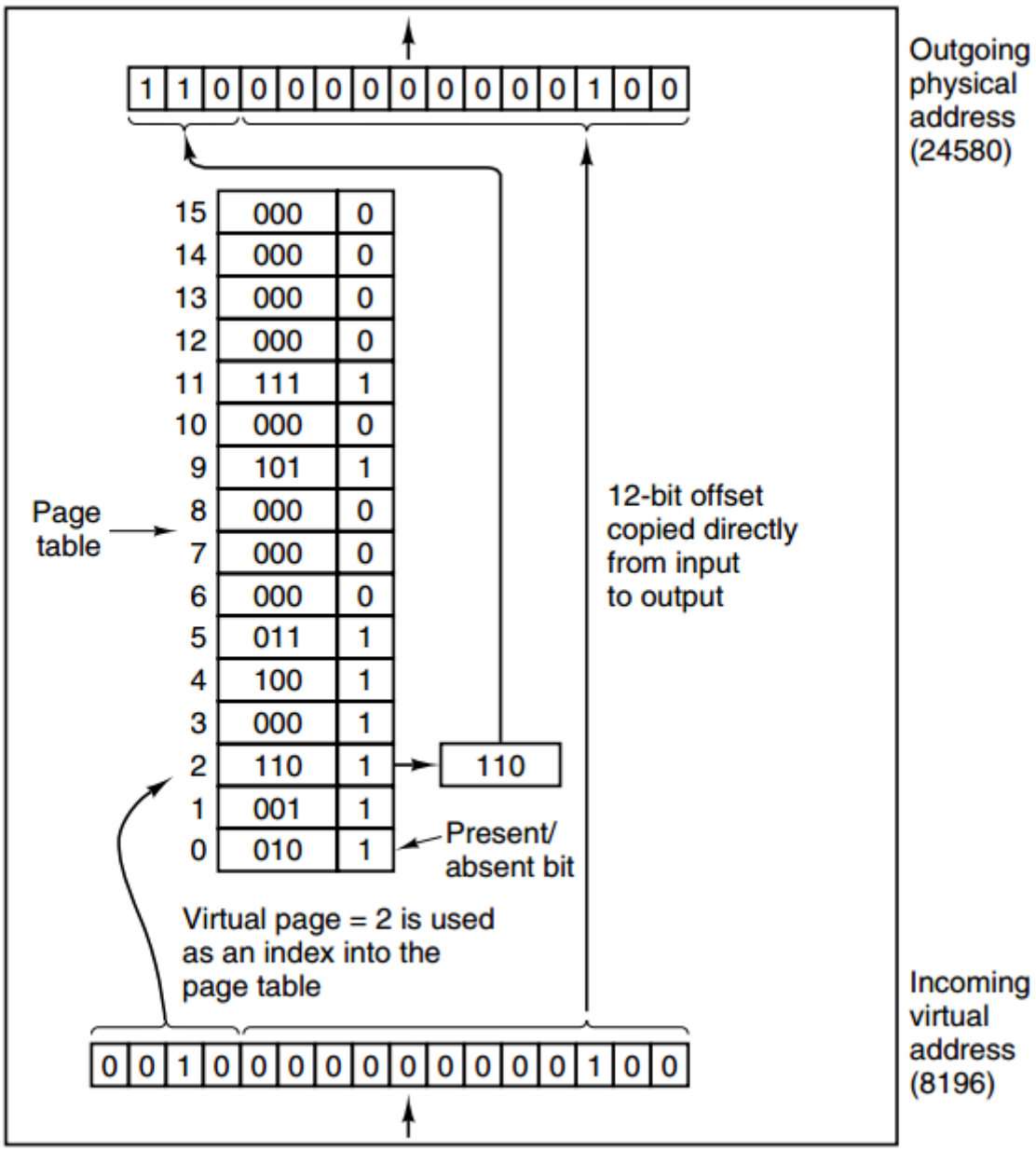


Figure 3-10. The internal operation of the MMU with 16 4-KB pages.

图中页表存着16个页，这16个页需要用4个比特位来进行索引定位。例如对于虚拟地址（0010000000000100），前面4位是存储页面号2，读取表项内容为110 1，页表项最后一位表示是否存在于内存中，1表示存在。后12位存储偏移量。这个页对应的页框地址为110 000000000100；

页面置换算法

程序运行中，如果访问的页面不在内存中，就发生缺页中断从而将该页调入内存中。此时如果内存无空闲空间，系统必须从内存中调出一个页面到磁盘兑换区中腾出位置；

页面置换算法类似于缓存淘汰算法。

目标是使得页面置换频率最低；

1. 最佳 所选择被换出的页面将是最长时间内不再被访问，通常可以保证获得最低的缺页率。

   理论上的算法，因为无法知道一个页面多长时间不再被访问。

2. 最近最久未使用LRU 虽然无法知道将来要使用的页面情况，但是知道过去使用页面的情况。LRU将最近最久未使用的页面换出。为了实现LRU，需要在内存中维护一个所有页面的链表。当一个页面被访问时，将这个页面移到链表表头。这样保证表尾的页面是最近最久未被访问的。

   每次访问都要更新链表，因此这种方式实现的LRU代价很高；

3. 最近未使用 每个页面都有两个状态位：R 与 M，当页面被访问时设置页面的 R=1，当页面被修改时设置 M=1。其中 R 位会定时被清零。可以将页面分成以下四类：

   R=0，M=0 R=0，M=1 R=1，M=0 R=1，M=1

   当发生缺页中断的时候，NRU算法随机的从类编号最小的非空类中挑选一个页面将他换出。NRU优先换出被修改的脏页面，而不是频繁使用的干净页面。

4. 先进先出 选择换出最先进入的页面，该算法会将经常被访问的页面换出使得缺页率升高；

5. 第二次机会算法 为避免FIFO算法的问题； 当页面被访问时，设置该页面R为1，需要替换的时候，检查最老页面的R位。如果R位为0，那么这个页面既老又没有被使用，可以立刻置换掉。如果是1，就将R请0，并把该页面放到链表的尾端，修改他的装入时间使得想刚刚装入一样，然后从链表头部开搜索。

6. 时钟 第二次机会算法需要在链表中移动页面，降低了效率。时钟算法使用环形链表将页面连接起来，再使用一个指针指向最老的页面；

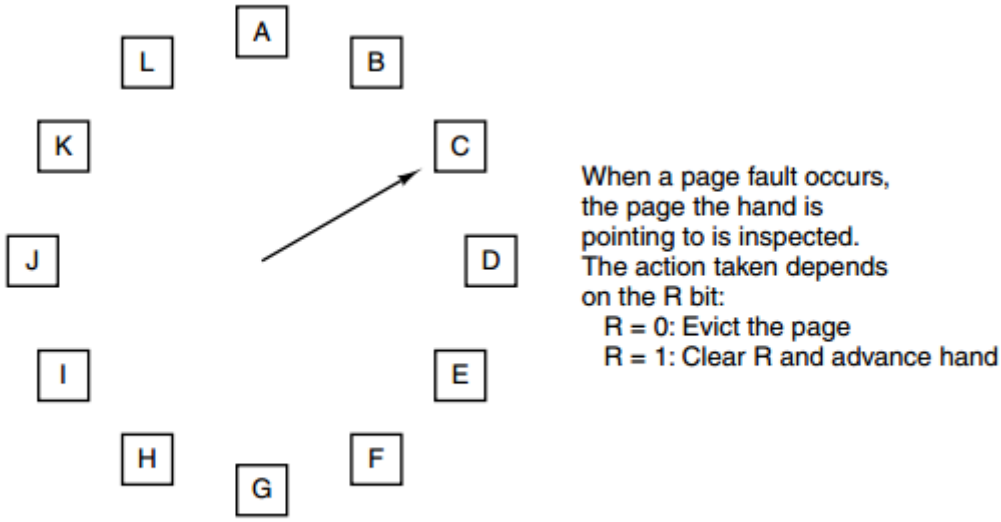
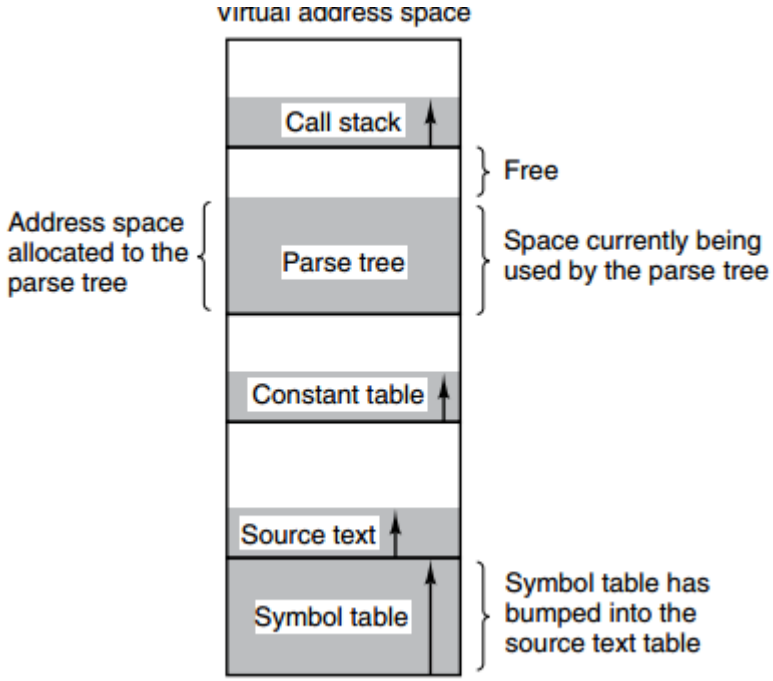


Figure 3-16. The clock page replacement algorithm.

分段

虚拟内存采用的是分页技术，也就是将地址空间划分成固定的大小的页，每一页和内存进行映射。

下图为一个编译器在编译过程中建立的多个表，有 4 个表是动态增长的，如果使用分页系统的一维地址空间，动态增长的特点会导致覆盖问题的出现。



分段的做法是把每个表分成段，一个段构成一个独立的地址空间。每个段的长度可以不同，并且可以动态增长.

分段和分页的区别

对程序员的透明性：分页透明，但是分段需要程序要显示的划分每个段；

地址空间的维度：分页是一维地址空间，分段是二维的。

大小是否可以改变：页的大小不可变，段的大小可以动态改变。

出现的原因：分页主要用于实现虚拟内存，从而获得更大的地址空间；分段主要是为了使程序和数据可以被划分为逻辑上独立的地址空间并且有助于共享和保护