

姓名：陈伟杰

学号：71066001

操作系统第2次作业

1. 动态内存分配需要对内存分区进行管理，一般使用位图和空闲链表两种方法。128MB的内存以  $n$  字节为单元分配，对于链表，假设内存中数据段和空闲区交替排列，长度均为 64KB。并假设链表中的每个节点需要记录 32 位的内存地址信息、16 位长度信息和 16 位下一节点域信息。这两种方法分别需要多少字节的存储空间？那种方法更好？

操作系统第2次作业

1.

位图：128MB =  $128 \times 1024 \times 1024 = 2^{27}$   
 $\Rightarrow 2^{27}/n$  字节  
 每个地址单元需要一个二进制数位  
 $\Rightarrow (2^{27}/n)/8$  字节

链表：128MB / 64KB =  $2^{27}/2^{16} = 2^{11}$  个单元  
 每个节点需要 32 + 16 + 16 + 1 = 64 Bit  
 存储空间： $2^{11} \times 8 = 2^{14}$  字节

当内存较小时，使用位图方法更加高效；而当内存较大时，空闲链表方法更好。

答：

2. 在一个交换系统中, 按内存地址排列的空闲区大小是: 10KB、4KB、20KB、18KB、7KB、9KB、12KB 和 15KB。对于连续的段请求: 12KB、10KB、9KB。使用 FirstFit、BestFit、WorstFit 和 NextFit 将找出哪些空闲区?

2. 10KB, 4KB, 20KB, 18KB, 7KB, 9KB, 12KB, 15KB  
请求: 12KB, 10KB, 9KB  
(对那个请求) First Fit 算法: 20KB, 10KB, 18KB  
(最相似那个) Best Fit 算法: 12KB, 10KB, 9KB  
(相差最少)  
(相差最多) Worst Fit 算法: 20KB, 18KB, 15KB  
(以 Best Fit 继续开始查) Next Fit 算法: 20KB, 18KB, 9KB

答:

//next fit 是根据 first fit 上次那个地址继续找 找到<=即可

3. 解释逻辑地址、物理地址、地址映射, 并举例说明

答：逻辑地址是指程序中使用的地址，它是虚拟的，在程序执行前就已经确定，是由程序员指定的，它与物理地址没有直接关联。

物理地址是指 CPU 访问主存储器时真正使用的地址，它是实际存在的，是由内存管理单元（MMU）将逻辑地址翻译为物理地址的结果。

地址映射是将逻辑地址映射为物理地址的过程。在计算机中，CPU 通过 MMU 将逻辑地址转化为物理地址，然后将物理地址传递给内存控制器访问主存储器。地址映射的主要目的是提供地址保护和地址共享的机制。

例子：举个例子，当程序需要访问内存中的某个数据时，它会使用一个逻辑地址来引用该数据，比如程序员将数据存储在地址为 0x800 的位置。在访问该数据时，CPU 会通过 MMU 将逻辑地址 0x800 映射到对应的物理地址然后物理地址为 0x8000。然后 CPU 使用物理地址 0x8000 来访问该数据

#### 4.解释页式（段式）存储管理中为什么要设置页（段）表和快表，简述页式（段式）地址转换过程。

答：页式（段式）存储管理中，物理内存被划分成大小相等的页框，虚拟内存也被划分成相同大小的页面。每个进程都有自己的页表，页表是一种数据结构，用于存储每个页（或段）的映射信息，包括页号（或段号）、页框号等等。还有设置页表是因为为了便于在内存找到进程的每个页面所对应块，分页系统中为每个进程配置一张页表，进程逻辑地址空间中的每一页，在页表中都对应有一个页表项，通过页表，操作系统可以快速地找到某个页（或段）在内存中的位置，并将它加载到需要的位置。设置段表是因为段表记录了段与内存位置的对应关系。所以，页式（段式）存储管理中的地址转换过程包括以下几个步骤：

从 PCB 中取出段表始址和段表长度，装入段表寄存器。

- 将段号与段表长度进行比较，若段号大于或等于段表长度，产生越界中断。
- 利用段表始址与段号得到该段表项在段表中的位置。

取出该段的页表始址和页表长度。

- 将页号与页表长度进行比较，若页号大于或等于页表长度，产生越界中断。
- 利用页表始址与页号得到该页表项在页表中的位置。
- 取出该页的物理块号，与页内地址拼接得到实际的物理地址

然而，由于页表通常较大，在每次内存访问时都需要访问页表，会造成较大的延迟，因此需要使用快表进行缓存。快表（TLB）是一种高速缓存，存储了最近经常使用的一些虚拟页到物理页的映射关系。使用快表可以有效减少访问页表的次数，提高地址转换效率。

#### 5. 叙述缺页中断的处理流程。

答：对于操作系统而言，当发生缺页中断的时候，处理有以下步骤：

操作系统会将缺失的页面从磁盘中加载到内存中的某个空闲帧（内存块）中。如果内存中没有空闲帧，操作系统会选择一个页面置算法（如采用 LRU 算法或 FIFO 算法和 Optimal 算法等）。当页面调入内存后，CPU 再次尝试访问该页面，即可顺利执行程序，如果操作系统发现该页面已经被修改，它会将其写回到磁盘上，以确保数据的一致性，然后，CPU 恢复到中断点并继续执行程序，最后重新执行引发缺页中断的指令，进行存储访问。

6. 假设一个机器有 38 位的虚拟地址和 32 位的物理地址。

(1) 与一级页表相比，多级页表的主要优点是什么？

(2) 如果使用二级页表，页面大小为 16KB，每个页表项有 4 个字节。应该为虚拟地址中的第一级和第二级页表域各分配多少位？

答：多级页表的主要优点是减少内存浪费，在一级页表中，每个进程都需要占用一个连续的页表，即使它只使用了很小的内存空间。而多级页表将页表分为多个级别，只有在进程实际需要占用的部分才会被映射到内存中。

6.

$$16KB = 16 \times 1024 = 16384 = 2^{14} \text{ (offset)}$$

38 位虚拟地址

$$\Rightarrow 38 - 14 = 24 \text{ 虚页号}$$

每个页表项为 4 个字节，所以每个页表的页表的页表项  $2^{14} / 4 = 2^{12}$

外页表的大小： $2^{24} / 2^{12} = 2^{12}$

第一级：12 位      第二级页表：12 位

7. 假设页面的访问存在一定的周期性循环，但周期之间会随机出现一些页面的访问。例

如: 0,1,2...,511,431,0,1,2...,511,332,0,1,2,...,511 等。请思考:

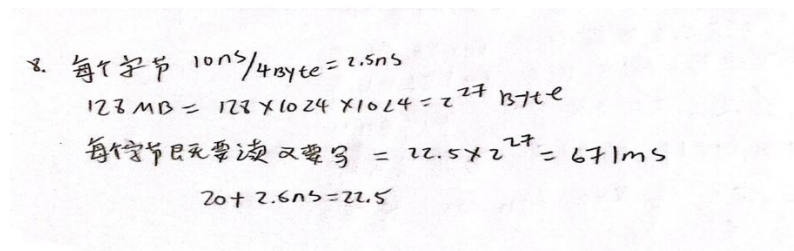
(1) LRU、FIFO 和 Clock 算法的效果如何?

(2) 如果有 500 个页框, 能否设计一个优于 LRU、FIFO 和 Clock 的算法?

答: 对于循环性访问页面和随机出现的页面访问, LRU 算法和 FIFO 算法表现会较差, 因为 LRU 算法不能很好地处理循环性访问, 而 FIFO 算法可能会淘汰长时间内未被访问的页面, 导致循环性访问时频繁缺页。相比之下, Clock 算法通常能更好地处理循环性访问, 因为它使用了时钟指针的概念, 能够较好地追踪页面访问的时间。

如果有 500 个页框, 可以考虑使用类似于 ARC (Adaptive Replacement Cache) 算法的策略, ARC 算法是一种自适应替换策略, 可以根据访问序列的变化自适应选择要缓存的页面。它将缓存页面分为两个部分, 最近访问的页面和最近不访问的页面, 并根据缓存命中率动态调整这两部分的大小。该算法在一些测试中表现比 LRU 算法更好。

8. 一个交换系统通过紧缩技术来清理碎片。如果内存碎片和数据区域是随机分配的。而且假设读写 32 位内存字需要 10nsec. 那么如果紧缩 128MB 的内存需要多久? 简单起见, 假设第 0 个字是碎片的一部分而最高位的字包含了有效的数据。



8. 每个字节  $10\text{ns}/4\text{Byte} = 2.5\text{ns}$   
 $128\text{MB} = 128 \times 1024 \times 1024 = 2^{27}\text{Byte}$   
每个字节既要读又要写  $= 22.5 \times 2^{27} = 671\text{ms}$   
 $20 + 2.5\text{ns} = 22.5$

答: