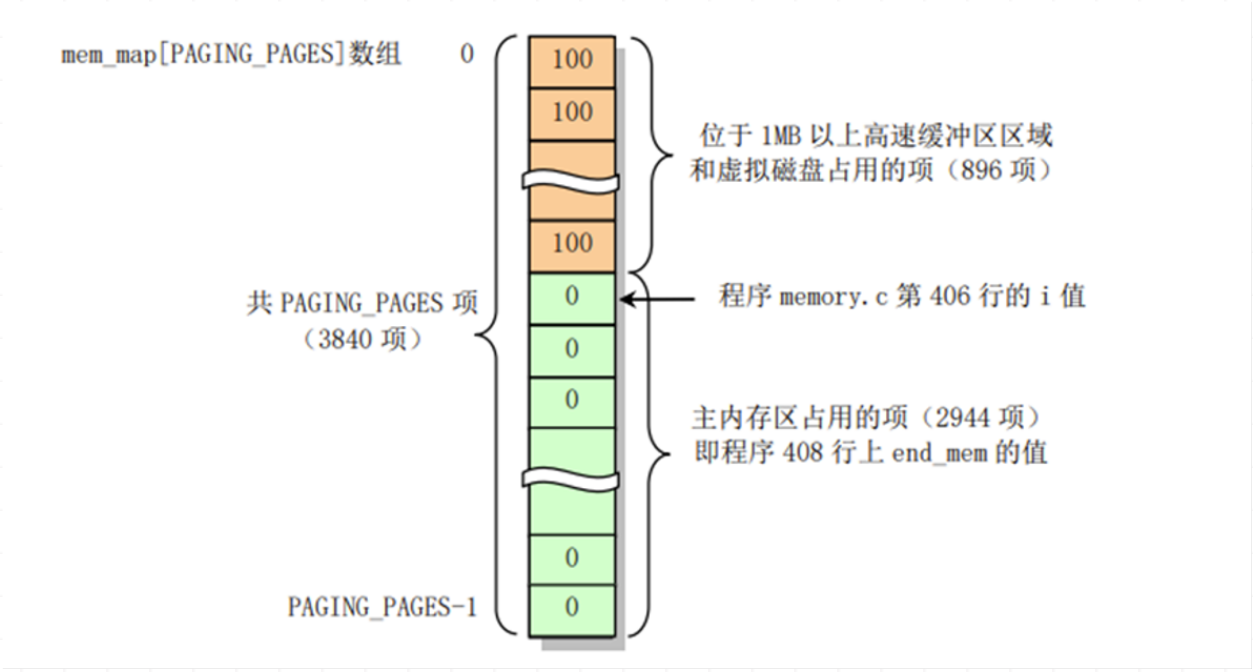
主题一 Part1

1.简述动态分区中，管理空闲区的数据结构设计，以及相关的分配和回收算法的实现。

数据结构主要采用两种设计，分别是：

1. 空闲分区表
2. 空闲分区链

第七组同学还从内核源码 `memory.c` 中的代码实现进行介绍，很清晰的说明出物理内存的划分情况。



关于分配算法伙伴分配器是关键，他的基本思想是。分配n阶页块时，查看是否有空闲的n阶页块，如果有直接分配，否则更大阶的内存块会被对半切分，切分之后的两个小块互为伙伴。其中一个子块用于分配，另一个加入空闲链表中。这些块在必要时会连续减半，直到达到所需大小的块为止。内存的释放是分配的逆过程，也可以看作是伙伴的合并过程。

2.从搜索速度、回收速度和空闲空间的利用率，比较分配算法的优缺点。

第八组同学用详细的动画演示了不同算法间的实际操作流程。

1. 首次适应算法
2. 循环首次适应算法
3. 最佳适应算法
4. 最坏适应算法

这四种算法综合来看，**首次适应算法效果最好**。但是低地址部分不断被划分，留下许多难以利用、很小的空闲区，每次查找又都从低地址部分开始，会增加查找的开销。第八组同学还分别从**搜索速度、回收过程**两个维度进行对比，不同算法对于不同形式的地址请求也是不一样的。

3.奔腾和ARM处理器采用多级页表。如何组织多级页表？多级页表如何进行地址转换？为了解决多级页表的速度问题，设置了TLB，说明使用TLB前后的访问时间。

第三题是我们第六小组研讨的选题。多级页表是对于单页表的改进，多级页表的地址转换方式很值得我们搞清楚。奔腾和ARM处理器之间的区别与联系页比较有趣。Pentium处理器每个时钟周期可以执行两条程序指令，使得它的处理能力比前代的Intel芯片要大而快得多。在相同的处理速度下，Pentium处理器执行指令的速度比80486快大约五倍。 ARM处理器本身是32位设计，但也配备16位指令集，一般来讲比等价32位代码节省达35%，却能保留32位系统的所有优势。

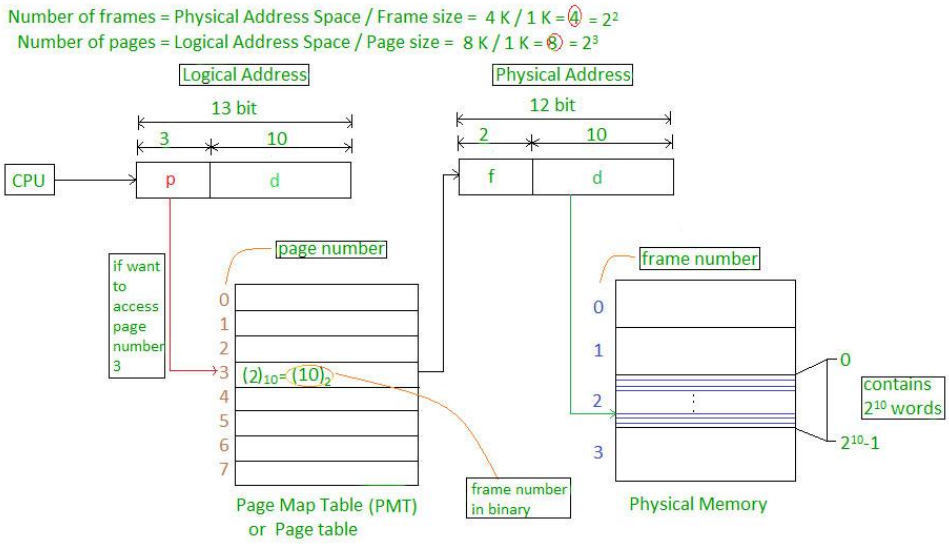
4.PowerPC处理器采用逆向页表。论述提出逆向页表的原因，和前向页表相比，逆向页表有和优点，存在哪些问题？一般如何解决？

逆向页表的提出解决了前向页表的缺点。在启动分页机制时需要用到页表，页表保存的是虚拟页号与物理页框之间的映射关系，其中页表项与虚拟内存页有一一对应的关系，当虚拟内存地址空间过大时页表项会占用过多内存（即使采用大页面，该问题也不能得到缓解）。

逆向页表主要由以下4部分组成：

1. 页码
2. 进程ID
3. 控制位
4. 链式指针

逆向页表的结构如下图所示：



5.奔腾处理器通过PAE模式实现超过4G物理内存的支持（可以映射到52位的物理地址），论述PAE模式地址映射过程。

PAE，指的是物理地址扩展。从奔腾处理器开始，intel引入了PAE机制：所有的32位应用程序都有4GB的进程地址空间，因为32位地址最多可以映射4GB的内存。

PAE映射的具体过程需要进行以下几个步骤：

1. 根据 CR3 找到 Page Directory Pointer Table
2. 根据一级索引在 Page Directory Pointer Table 中查询到 Page Directory
3. 根据二级索引在 Page Directory 中查询到 Page Table

4. 根据三级索引在 Page Table 中查询到4KB 页
5. 根据地址的11～0：确定页内偏移