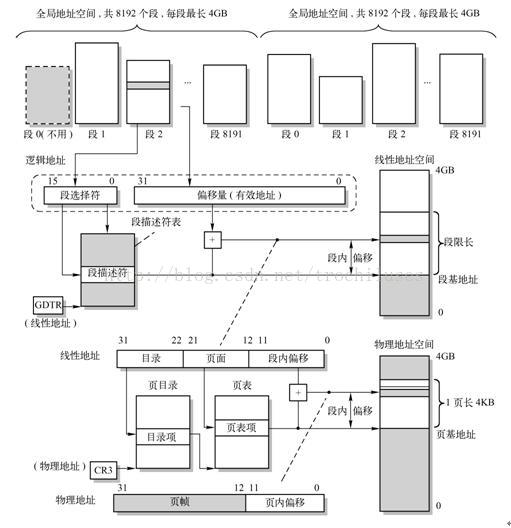
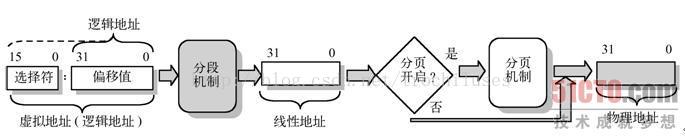
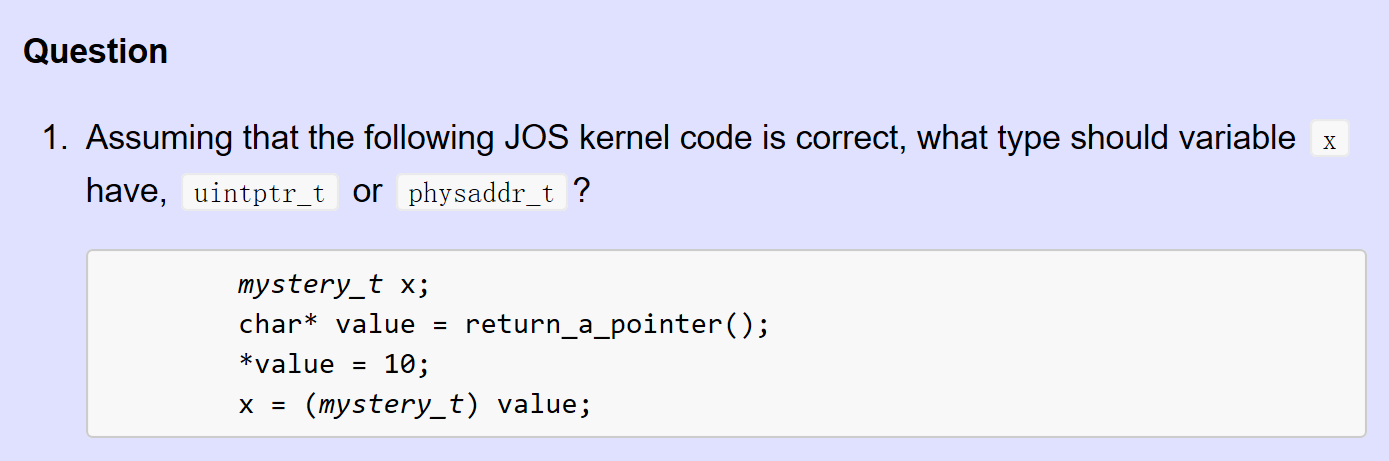
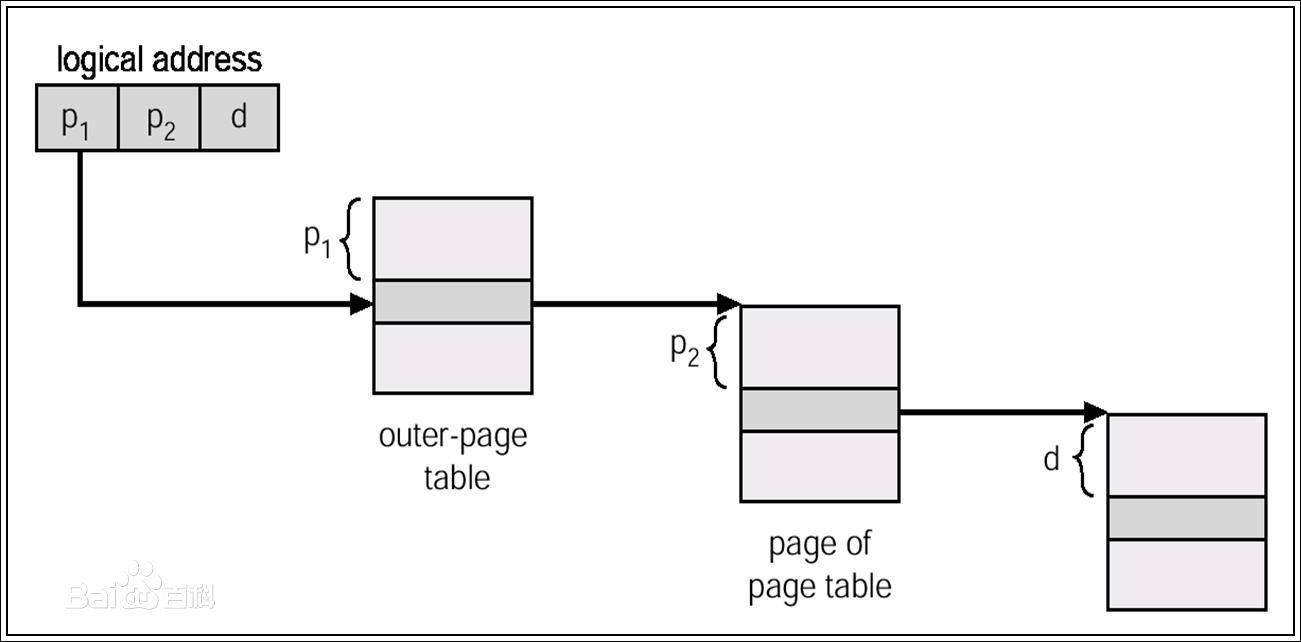
ROUNDUP（a,b ）是一个数学函数，把a变成b对齐的形式，可以简单地理解为把a变成b的倍数，然后返回





是 uintptr\_t因为程序里面只会存在虚拟地址

？？？如何解析physaddr\_t



Question 1:

是 uintptr\_t因为程序里面只会存在虚拟地址

Question 2:

|  |  |  |
| --- | --- | --- |
| Entry | Base Virtual Address | Points to(logically) |
|  | 0xffc00000 | Page table for top 4MB of phys memory |
|  |  |  |
|  | 0xf0000000 | PA range [0, 2^32 - KERNBASE)( Page table for bottom 4MB of phys memory) |
|  |  |  |
|  | 0xef800000 | Kernel stack,actually,0xef800000-0xefbf8000 is guard page  0xefbf8000 -0xefc00000 is |
|  | 0xef400000 | Page directory |
|  | 0xef000000 | Page arrays |
|  |  |  |
|  | 0x00800000 |  |
|  | 0x00400000 |  |
| 0 | 0x00000000 |  |

Question 3:

虚拟地址的低12位里面有权限位PTE\_U, MMU在做地址翻译的时候会检查这一位，检查用户是否有权限访问该地址

Question 4:

当前OS的物理内存是256MB，不过按pages = boot\_alloc(npages \* sizeof(struct Page)); pages大小是4MB的能力来说，是可以支持(4MB/8byte)\*4K = 2G

Question 5:

管理当前内存：当前物理内存66556K，就是16639个page，需要17个二级页表

及 4k(PD) + 68k(PT) + 4MB(pages) = 4MB + 72k

管理2G内存所需的内存：4k(PD) + 2M(PT) + 4MB(pages) = 6MB + 4K

Question 6:

什么时候above KERNBASE： 在entry.S第67行

When low EIP: 因为在entrypgdir里面，同一个物理页对应两个虚拟页，一个高地址，一个低地址。

Why： 因为在内核地址空间, EIP要小于KNERBASE。