Design

1. PageTable

The pages for page directory and page tables are allocated from process_mem_pool, instead of the so limited 4MB kernel memory pool, so that we can manage a large address space.

Since the page table pages are free-mapped, and the address issued by CPU is logical address, we need to do some trick to manage the page directory and page tables. The policy here is Recursive Page Table Lookup. So we put the physical address in the last entry of the page directory.

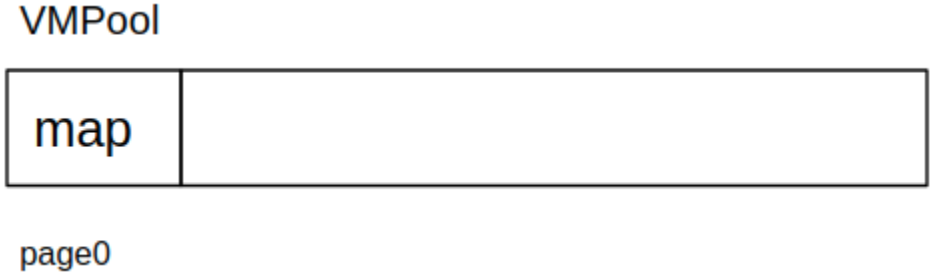
How to access page directory and page table?
Assume addr is the page fault address, so the PageDirectoryIndex is addr>>22, the PageTableIndex is (addr>>12) & 0x3FF.
Now the logical address to access the page directory is 0xFFFFF000, the logical address to access the corresponding pagetable is 0xFFC00000 + (PageDirectoryIndex<<12).

2.VMPool

(1)VMPool management information

For simplicity, I use the first page of the VMPool to store the VMPool management information.Define a Map structure to keep trace of the base address and size of each VM block.

```
typedef struct Map {
    unsigned long base;
    unsigned long size;
} Map;
```



(2)Allocate VM

Each time allocating a block of vm, I will keep track of its base address and size in the Map structure; also keep track of the available vm size of this VMPool.

(3)Release VM

When releasing a block of vm, I will free the corresponding pages, clear the map information, and also reload the page table in order to flush the TLB.