PA3 实验报告

3.2

1. NEMU 在什么时候进入了保护模式?

Kernel/start 中 start.S

```
lgdt    va_to_pa(gdtdesc) # See i386 manual for more information
movl    %cr0, %eax        # %CR0 |= PROTECT_ENABLE_BIT
orl     $0x1, %eax
movl    %eax, %cr0
```

在把 cr0 寄存器的最低位 pe 设为 1 时，开启保护模式。

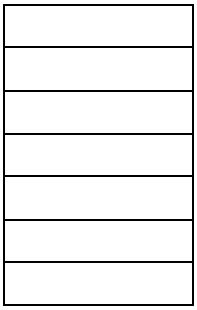
2. 在 GDTR 中保存的段表首地址是虚拟地址、线性地址、还是物理地址?为什么?

线性地址。因为 GDTR 中段表首地址为 32 位，所以不为虚拟地址，而此时并未经过段级地址转换所以并未得到物理地址。

3.3

1. Kernel 的虚拟页和物理页的映射关系是什么?请画图说明;

Kernel 的虚拟页起始地址为物理页起始地址加上 0xc0000000 得到的。



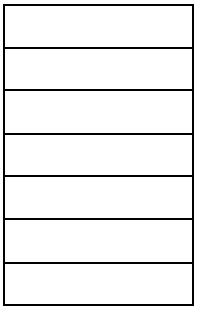
0xc0030000 虚拟地址

0x30000 物理地址

2. 以某一个测试用例为例,画图说明用户进程的虚拟页和物理页间映射关系又是怎样的?Kernel 映射为哪一段?你可以在 loader()中通过 Log()输出 mm_malloc 的结果来查看映射关系,并结合 init_mm()中的代码绘出内核映射关系。

```
Execute ./kernel/kernel.img ./testcase/bin/mov-c
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,28,loader] {kernel} ELF loading from ram disk.
nemu trap output: [src/elf/elf.c,38,loader] {kernel} paddr = 1000000

nemu trap output: [src/elf/elf.c,39,loader] {kernel} vaddr = 8048000

nemu trap output: [src/elf/elf.c,38,loader] {kernel} paddr = 1001000

nemu trap output: [src/elf/elf.c,39,loader] {kernel} vaddr = 804a000

nemu: HIT GOOD TRAP at eip = 0x08048197
NEMU2 terminated
```

mov-c:



0x8048000 虚拟地址

0x1000000 物理地址

Init_mm 函数，映射到 koffset = 0xc0000000 上的系统内核区。

3.“在 Kernel 完成页表初始化前,程序无法访问全局变量”这一表述是否正确?在 init_page()里面我们对全局变量进行了怎样的处理?

正确。只有在 init_page()函数中，才将页表和页目录表变量创建并初始化，页目录表构建与页表对应映射关系，与虚拟页面的对应映射关系，同时构建页表和物理页面的对应映射关系。而在调用之前，相关变量和内容均在系统内核区，用户程序无法访问。

```c
/* set up page tables for kernel */
void init_page(void) {
    CR0 cr0;
    CR3 cr3;
    PDE *pdir = (PDE *)va_to_pa(kpdir);
    PTE *ptable = (PTE *)va_to_pa(kptable);
    uint32_t pdir_idx, ptable_idx, pframe_idx;

    /* make all PDE invalid */
    memset(pdir, 0, NR_PDE * sizeof(PDE));

    /* fill PDEs and PTEs */
    pframe_idx = 0;
    for (pdir_idx = 0; pdir_idx < PHY_MEM / PT_SIZE; pdir_idx ++) {
        pdir[pdir_idx].val = make_pde(ptable);
        pdir[pdir_idx + KOFFSET / PT_SIZE].val = make_pde(ptable);
        for (ptable_idx = 0; ptable_idx < NR_PTE; ptable_idx ++) {
            ptable->val = make_pte(pframe_idx << 12);
            pframe_idx ++;
            ptable ++;

        }
    }
```