1. Kernel的虚拟页和物理页的映射关系是什么？请画图说明；

是全相联的映射关系：

地址 内容

0xc8000000 – 0xc7fff000 操作系统栈区

0xc7fff000 – 0xc0032000 --

0xc0032000 – 0xc0031000 操作系统代码区

0xc0031000 – --

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。

是全相联的映射关系：

地址 内容 地址

0xc8000000 – 0xc7fff000 Kernel栈区 0x8000000 – 0x7fff000

0xc0031000 – 0xc0030000 Kernel代码区 0x31000 – 0x30000

0xc0000000 – 0xbffff000 用户进程栈区 0x1102000 – 0x1101000

0x804b000 – 0x804a000 用户进程数据区 0x1002000 – 0x1001000

0x8049000 – 0x8048000 用户进程代码区 0x1001000 – 0x1000000

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。全局变量的定义和链接使用虚拟地址，在Kernel完成页表初始化前，无法确定它们对应的物理地址，则程序无法对其进行访问。在init\_page()中可以看到使用宏va\_to\_pa对全局变量进行了处理，将虚拟地址减去KOFFSET得到物理地址，这时就可以访问了。