MMIO和PIO

1、概念

内存映射I/O（MMIO）【统一编址】和端口映射I/O(PMIO)【独立/单独编址】是两种互为补充的I/O方法，用于设备驱动程序和设备通信，即在CPU和外部设备之间。

（1）在MMIO中，内存和I/O设备共享同一个地址空间。 MMIO是应用得最为广泛的一种IO方法，它使用相同的地址总线来处理内存和I/O设备，I/O设备的内存和寄存器被映射到与之相关联的地址。当CPU访问某个内存地址时，它可能是物理内存，也可以是某个I/O设备的内存。因此，用于访问内存的CPU指令也可来访问I/O设备。每个I/O设备监视CPU的地址总线，一旦CPU访问分配给它的地址，它就做出响应，将数据总线连接到需要访问的设备硬件寄存器。为了容纳I/O设备，CPU必须预留给I/O一个地址区域，该地址区域不能给物理内存使用。

实现MMIO：内核使用ioremap()将IO设备的物理内存地址映射到内核空间的虚拟地址上； 用户空间程序使用mmap(2)系统调用将IO设备的物理内存地址映射到用户空间的虚拟内存地址上，一旦映射完成，用户空间的一段内存就与IO设备的内存关联起来，当用户访问用户空间的这段内存地址范围时，实际上会转化为对IO设备的访问。iowrite8(u8 value, void *addr)；iowrite16/iowrite32

（2）PMIO（IO端口也可以映射到虚拟地址空间进行访问ioport_map）。在PMIO中，内存和I/O设备有**各自的地址空间**。 端口映射I/O通常使用一种特殊的CPU指令，专门执行I/O操作。在Intel的微处理器中，使用的指令是IN和OUT。这些指令可以读/写1,2,4个字节(例如：outb, outw, outl)从/到IO设备上。I/O设备有一个与内存不同的地址空间，为了实现地址空间的隔离，要么在CPU物理接口上增加一个I/O引脚，要么增加一条专用的I/O总线。

用户空间想访问IO端口：必须使用ioperm和iopl系统调用(#include) 来获得进行操作I/O端口的权限。ioperm 为获取单个端口的操作许可，iopl 为获取整个I/O空间许可。这2个函数都是x86特有的。

x86 CPU的I/O空间就只有64KB（0－0xffff)。

Linux内核必须使用“资源”来记录分配给每个硬件设备的I/O端口。资源表示某个实体的一部分，这部分被互斥地分配给设备驱动程序。在这里，资源表示I/O端口地 址的一个范围。每个资源对应的信息存放在resource数据结构中

2、区别

（1）在MMIO中，IO设备和内存共享同一个地址总线，因此它们的地址空间是相同的；而在PMIO中，IO设备和内存的地址空间是隔离的。

（2）在MMIO中，无论是访问内存还是访问IO设备，都使用相同的指令（mov类型的读写内存的指令）；而在PMIO中，CPU使用特殊的指令访问IO设备，在Intel微处理器中，使用的指令是IN和OUT。

（3）对MMIO操作是申请-映射-访问-释放（访问流程：request_mem_region() -> ioremap() -> ioread8()/iowrite8() -> iounmap() -> release_mem_region() ）；PMIO是申请-访问-释放（不映射到内存空间，直接使用 intb()/outb()之类的函数来读写IO端口。

（4）MMIO：CPU需要截获虚拟机访问的具体地址，并发生了异常，从VM-mode下退出来，让qemu继续处理，模拟硬件的行为即可。这就是MMIO下的设备模拟过程，CPU截获MMIO的是misconfig异常；
PMIO：CPU只要截获VM（Virtual Machine）的in、out指令，就可以知道CPU想要访问设备，那么用软件来模拟硬件的行为，就可以让VM觉得自己有设备。

（5）MMIO：cat /proc/iomem命令查看外设的**IO 内存物理地址**分布情况；PMIO：cat /proc/ioport，列出了系统所有的IO端口分布情况，注意这边看到的**地址不是物理地址**，而是**IO 端口号的分布情况**，跟**物理地址没有关系**，CPU访问外设寄存器就是通过传入这些端口号来访问外设寄存器的。