Lab5：CPU虚拟化

问题

1. 请调研并用你的理解解释可虚拟化架构与不可虚拟化架构的概念（参考书籍 2.1 节）

   首先，需要理解特权指令和敏感指令。一些指令必须在CPU的最高特权级别执行，如果在非最高特权级中执行，则会触发特权切换，陷入到最高特权级中，这类指令成为特权指令；而敏感指令是指需要操作敏感物理资源的指令，如I/O指令、页表基地址切换指令。

   所有敏感指令都是特权指令的架构称为可虚拟化架构，该架构中所有的敏感指令都能陷入 Hyperisor，可以通过“陷入-模拟”的方式实现虚拟化；而存在敏感非特权指令的架构称为不可虚拟化架构，该架构存在“虚拟化漏洞”，需要解释执行、二进制翻译、扫描与修补以及半虚拟化等技术额外解决。

2. 请基于你的理解谈谈虚拟化的“陷入再模拟”的概念（参考书籍 1.3.3 节）

   在一般情况下，操作系统具有物理访问的最高权限级别，可以直接访问寄存器、内存、I/O设备等关键的物理资源；但在虚拟化的情况下，这些资源的访问由 Hypervisor 接管，即客户机的操作系统并不是最高权限。因此，当客户机操作系统试图访问物理资源时，就会触发异常，陷入 Hypervisor 的监控中；之后 Hypervisor 识别这些特权指令，通过模拟的方式执行这些指令，而不是直接访问物理设备。

3. 请调研并用你的理解解释 Intel VT-x 的特权级是如何划分的。这种非根模式为何有助于 Hypervisor “陷入再模拟”（参考书籍2.2节）？

   Intel VT-x引入了VMX操作模式，包括根模式和非根模式，根模式和非根模式分别包含 Ring0-Ring3 特权级。

   Hypervisor 运行在根模式，而虚拟机运行在非根模式。Intel VT-x 改变了非根模式下敏感非特权指令的语义，使所有的敏感指令都会触发 VM-Exit，陷入 Hypervisor 进行处理；在根模式下，Hypervisor 读取 VM-Exit 相关信息并进行相应处理，即进行模拟过程；处理完成后，再从根模式切换回非根模式，恢复虚拟机运行。通过这种方式，能够解决“虚拟化漏洞”问题，并减少资源开销。

实验

在 KVM_EXIT_IO 的处理中，将输出的字符转为大写即可，如下：

```
156         case KVM_EXIT_IO:
157             if (run->io.direction ==
                KVM_EXIT_IO_OUT && run->io.size == 1 &&
                run->io.port == 0x3f8 && run->io.count ==
                1)
158                 putchar(toupper(*(((char *)run) +
                    run->io.data_offset)));
159             else
160                 errx(1, "unhandled KVM_EXIT_IO");
```