A Special Case in Lab4

受众范围

陆续收到很多反馈进不了中断，所以统一写个帮助文档解惑

如果你：

1. 中断程序只进了一次/某次运行之后再也进不去了
2. 查KBSR发现始终为xC000
3. 在interrupt中没有写/写错了读取KBDR的部分

那恭喜你是本文档的受众，否则请仔细检查boot是否出错。

太长不看篇

解决方案：

1. 在interrupt routine中加上无条件的从KBDR中进行读取（反正最后肯定会用到的）
2. 重启模拟器

简易原理篇

Memory-Mapped I/O不能简单理解为内存的存取，每一步存取都意味着对I/O设备的指令。而当前模拟器使用的Keyboard实现并不是课本中一个14位15位的与门就当中断信号的简单实现，实际上当一个输入没有被处理（即读取一次KBDR）时它不会触发第二次中断。解决的方法就是每当一个字符触发中断时就把他从缓存里拿出来。
因此很有可能你们的缓存被一个在**某次运行**中触发了中断的字符堵住了。
顺便提一下，你进editor之类的操作并不会关掉模拟器的实例，包括其中的Keyboard等device，所以重启模拟器是最简单的重置Keyboard缓存的方法

好奇宝宝篇

你可能很好奇，既然KBSR是 `xC000` 那按照状态机图INT应该始终为1，因此每个周期都会进入中断才对吧

书上其实不是很关心I/O设备的具体实现，这些确实不是重点，但是如果你真的（像某Head TA一样）好奇的话，你可以从这里下载到模拟器的源代码
https://github.com/chiragsakhuja/lc3tools.git
克隆之后关注到 `src/backend/device.cpp`
在一个 `tick` 函数（听名字就是按周期执行的）里能找到背锅的代码

```
PIMicroOp KeyboardDevice::tick(void)
{
    // Set ready bit.
    char c;
    if(inputter.getChar(c)) {
        key_buffer.emplace(c);
    }

    if(! key_buffer.empty()) {
        status.setValue(status.getValue() | 0x8000);
        data.setValue(static_cast<uint16_t>(key_buffer.front().value));

        if(! key_buffer.front().triggered_interrupt && (status.getValue() & 0x4000) == 0x4000) {
            key_buffer.front().triggered_interrupt = true;
            return std::make_shared<PushInterruptTypeMicroOp>(InterruptType::KEYBOARD);
        }
    }

    return nullptr;
}
```

inputter是来自前端的字符流（模拟器中就是console，我们的自动判题程序则简单的用字符串代替）
key_buffer则是device内部的缓存机制，类型为 `queue<KeyInfo>` 具体如下所示
除了我们熟悉的status最高位置一和data存放数据之外，interrupt的触发被单独提出来了（而不是课本介绍的KBSR[14]&KBSR[15]）
这里返回内容（相对平时无中断返回的nullptr）就是代表键盘中断的值
结合KeyInfo的结构，除了保存字符信息之外它额外保存是否已经触发中断。而触发中断的前提条件就是它之前没有触发过中断。

```
struct KeyInfo
{
    char value;
    bool triggered_interrupt;

    KeyInfo(void) : KeyInfo(0) { }
    KeyInfo(char value) : value(value), triggered_interrupt(false) { }
};
```

~~我的评价是小天才的设计~~其本意大概是让输入流能正常工作，不过其实这个有点偏离课本中提供的简洁范例，所以这里只供好奇宝宝看看，大家重点还是要知道书上生成INT信号的方式
解决的方法可以看同文件的read函数（以及一大堆实现微指令的奇妙东西），直接把triggered_interrupt=true的那一项pop出去就完事了，落实到程序上就是对KBDR这个位置进行一次读取访问。