Xv6 音频播放报告

一、 概述

应用 Intel ac97 声卡，实现了 wav 格式文件的播放。

AC97 是一种 PCI 声卡的规范，具体硬件信息可参考官方文档《Intel 82801AA AC'97 programmer's manual》。

在本次实验中，主要了解 AC97 声卡在程序当中的初始化流程和使用方法以及 wav 格式文件的相关结构，需要掌握的相关概念包括：

- ✓ 遍历总线查找声卡
- ✓ PCI 配置空间（PCI Configure Space）
- ✓ 声卡缓存的使用（Buffer Descriptor List）
- ✓ wav 文件格式

二、 具体实现

1. 硬件初始化

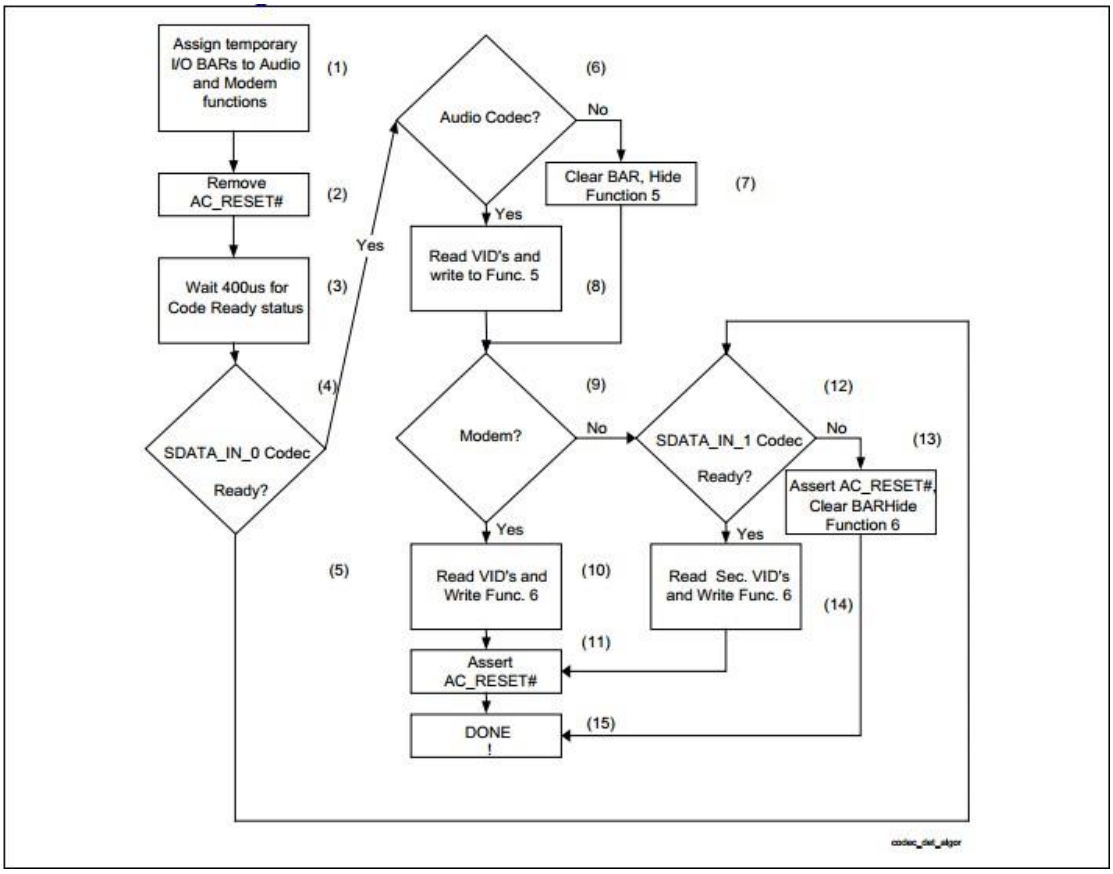
在 qemu 虚拟机中实现 xv6　ac97 声卡驱动，分为以下几步：

a. 让 qemu 提供虚拟的声卡。实现这个步骤需要在用 qemu 运行 xv6 时输入以下的参数：qemu –soundhw ac97 –hda xv6.img –hdb fs.img

b. 找到声卡设备在 PCI 的总线地址。PCI 配置空间为 CPU 控制外部设备提供了方便。在操作系统中控制声卡，首先需要找到 PCI 的设备。做法是询问所有 PCI 地址的标识，找到与 AC97 匹配的一项（0x24158086），该地址就是声卡在 PCI 配置空间的地址。具

体代码实现见 sound.c (soundinit 函数)。

c. 声卡初始化：



查阅 Intel 82801AA AC'97 programmer's manual，可以获得 ac97 声卡初始化的流程。在这里，我们仅仅需要实现 1-8 步，就可以成功初始化声卡。每一步需要写入 PCI 控制段和数据段的地址可以从该文献中获得。具体代码见 sound.c(soundcardinit 函数)。

d.建立 Buffer Descriptor List(具体结构参见官方文档),将该结构物理内存(通过 v2p 函数进行地址转换) 的起始地址写入 Buffer Descriptor List Base Address register。

2. 声音播放与系统调用

播放 wav 涉及两个进程，一个是读文件(play.c)，另外一个是缓存播放(sysaudio.c)。这样，我们可以做到在缓存声音数据和播放声音的同时用户程序可以继续读取声音文件。

这里涉及用户程序与内核程序的协同工作，所以提供多个系统调用来满足多进程播放程

序。

- setSampleRate：设置声卡采样率。
- write：传递读取的文件到内核。
- setVolume：设置声音大小。
- pause：通知内核暂停播放。
- wavdecode：开始缓存用户读取的音频数据。

3. 播放逻辑和进程通信

play.c:

循环读取数据，调用 kwrite 存入系统音频缓冲区 buf，并唤醒解码进程。由于读取文件的速度瓶颈，采用了一次读取较大的数据量，分次写入音频缓冲区的方法。

sysaudio.c:

循环调用 wavdecode 等待新的音频缓冲区数据 buf。等待期间阻塞，一旦有新的数据，就被唤醒，接着等待声卡空闲时转存入声卡的缓冲区。

sound.c:

将音频缓冲区 buf 转存入声卡的缓冲区，即将 Buffer Description List 的每一个 Buffer Pointer 指向缓冲区物理内存地址的相应位置，并写入 Commend 和 Length 数据。

播放流程：

程序入口为 play.c，该进程（进程 1）打开 wav 文件，并启动子进程（进程 2）运行 sys_decode 函数进行解码。进程 1 每读取一段文件数据，分段写入音频缓冲区（kwrite），唤醒进程 2，它将把数据写入声卡的缓冲区进行播放。声卡缓冲区使用了 3 个

缓冲块进行队列式缓存，防止播放间断。

4. 注意事项

在向声卡写入缓存数据时（实际是写入数据段内存的地址），注意要使用物理地址，包括 Buffer Descriptor List 结构中的所有相关的指针均需要使用 v2p 函数进行转换

三、 存在的问题与可做的工作

- 当前版本在 Ubuntu 14.04 上运行无法播放声音
- 实际使用中，有时仍会出现少许卡顿的情况，可进一步优化缓存机制消除此现象
- 解码过程启用两个进程，实际依旧为串行运行，可优化其效率
- 在整合版本中，如果同时运行其他 GUI 进程，可能会造成播放卡顿的现象