实 验 报 告

报告名称： 基于 xv6 的音频播放

姓　　名：和嘉晅、何承昱
徐浩博、马越洲

指导教师：闻立杰老师

报告日期：2022 年 6 月 25 日

软件学院

目录

1. 最终成果 2

2. 运行指南 2

2.1. 运行环境 2
2.2. 命令 2

3. 实现过程 4

3.1. 项目结构 4
3.2. AC97 声卡驱动 4
3.2.1. 声卡初始化 4
3.2.2. 配置 DMA 引擎 4
3.3. 内核 I/O 软件 5
3.4. 音频解码 7
3.4.1 PCM 基础知识 7
3.4.2 WAV 文件解码 7
3.4.3 MP3 文件解码 8
3.4.4 FLAC 文件解码 8
3.5. 音频播放器 8

4. 困难与挑战 9

4.1. 如何访问 I/O 独立编址的 AC97 寄存器 9
4.2. 修改 xv6 程序的栈空间大小 10
4.3. 允许更大的音频文件 10
4.4. 解码库的兼容性问题 11
4.5. xv6 目录项的长度 11

5. 未来展望 11

5.1. 优化内存管理和文件系统 11
5.2. 支持更多格式音频文件 11
5.3. 支持 MP4 视频文件 11
5.4. 制作音乐播放器 GUI 12

6. 人员分工 12

附录 1　　个人信息 13

附录 2　　参考资料 13

1. 最终成果

在 xv6 上实现了音频播放器：

- 支持 MP3、WAV、FLAC 多种音频格式的播放。
- 支持播放，暂停/恢复，停止，切歌，调节音量。
- 有用户友好的命令行播放器界面。

2. 运行指南

2.1. 运行环境

- **Ubuntu 22.04**

  Windows 平台请使用支持音频的虚拟机（如 VMware），WSL 不支持原生的音频播放。

- **QEMU 6.2.0**

  在 Ubuntu20.04.4 中输入命令 `apt-get update && apt-get install qemu-system-misc` 得到的 QEMU 为 4.2.1 版本，经过测试，使用 QEMU 4.2.1 版本运行 xv6 音频播放器会发生未知错误，不能播放音频。

2.2. 命令

- 配置 xv6-riscv 运行环境：

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install git build-essential gdb-multiarch qemu-system-misc gcc-riscv64-linux-gnu binutils-riscv64-linux-gnu
```

- 运行 xv6：`make qemu`
- （在 xv6 中）启动音频播放器：`player`

```
$ player
Welcome to the music player!
Local music list:
class.mp3      long5.wav      1.mp3          summer.mp3
1minute.mp3    novia.mp3      15.mp3         haoyunlai.mp3
bgm.flac
Enter Command:
```

- player 支持的命令：

指令	效果
`play filename`	播放音乐

`pause`	暂停播放（可恢复）
`resume`	恢复播放
`stop`	停止播放（不可恢复）
`volume {int 0~100}`	调节音量（默认值 50）
`list`	显示可以播放的音频列表
`exit`	退出音频播放器

- 添加音频文件
  - 将新的音频文件放在：`./audio/`下
  - 修改 `Makefile` 中的 `AUDIOS` 项，将文件添加到虚拟硬盘中
  - 若 `fs.img` 存在：`rm fs.img`
  - `make qemu`

3. 实现过程

3.1. 项目结构

项目可以划分为 AC97 驱动程序、内核 I/O 软件、MP3 解码、前端音频播放器 4 个模块，其组织结构如下图：

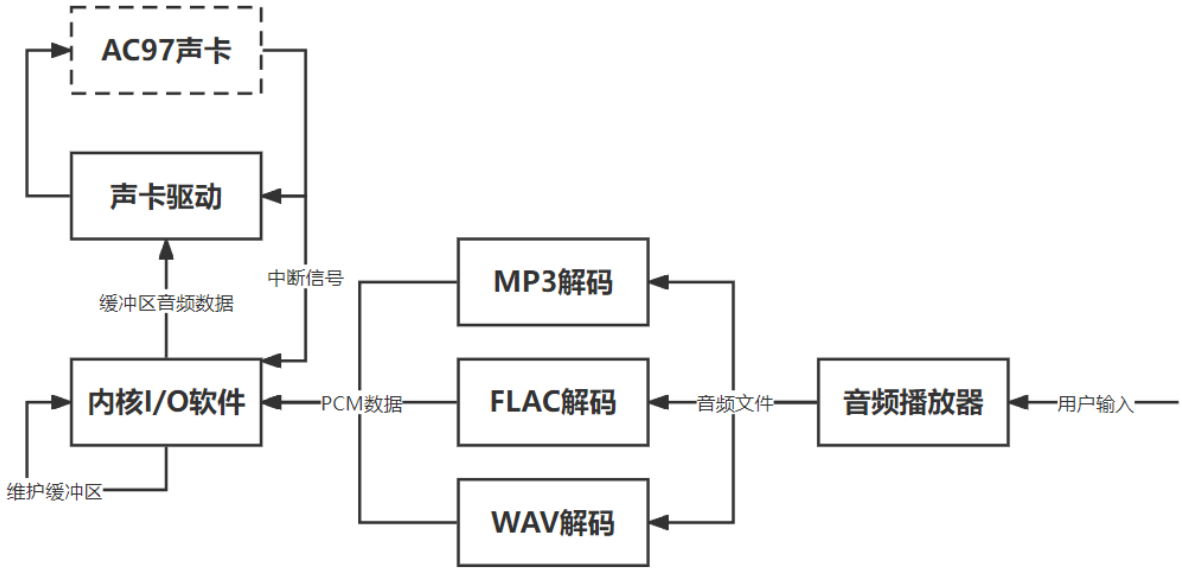


图 1. 项目的组织结构

3.2. AC97 声卡驱动

AC97 PCI Configuration Space Register 的详细定义请参考 Intel® 82801AA (ICH) and Intel® 82801AB (ICH0) I/O Controller Hub DataSheet 的第 12.1 章，Native Audio Bus Master Control Registers（以下简称主控寄存器）和 Native Audio Mixer registers（以下简称 Mixer 寄存器）的定义参见第 12.2 章。

3.2.1. 声卡初始化

首先定位枚举 AC97 PCI Configuration Space：PCIE_ECAM 中所有设备，通过内存映射方式读取 VID 寄存器与 DID，编号为 0x24158086 的设备为 AC97 声卡。

配置 PCI Configuration Space 中的 NAMBAR 寄存器与 NABMBAR 寄存器，以允许访问 I/O Space 中 Mixer 寄存器和主控寄存器。之后按照 AC '97 Programmer's Reference Manual 进行 RESET 以及配置 Codec，完成声卡初始化。

3.2.2. 配置 DMA 引擎

AC97 使用 DMA 引擎将内存中的音频数据传输到声卡。

首先需要在内存中设置一个 Buffer Descriptor List，包含 32 个 Buffer Descriptor，每个 Buffer Descriptor 通过起始地址和长度描述了一段存储 PCM 格式音频数据的 Buffer，如下图所示：

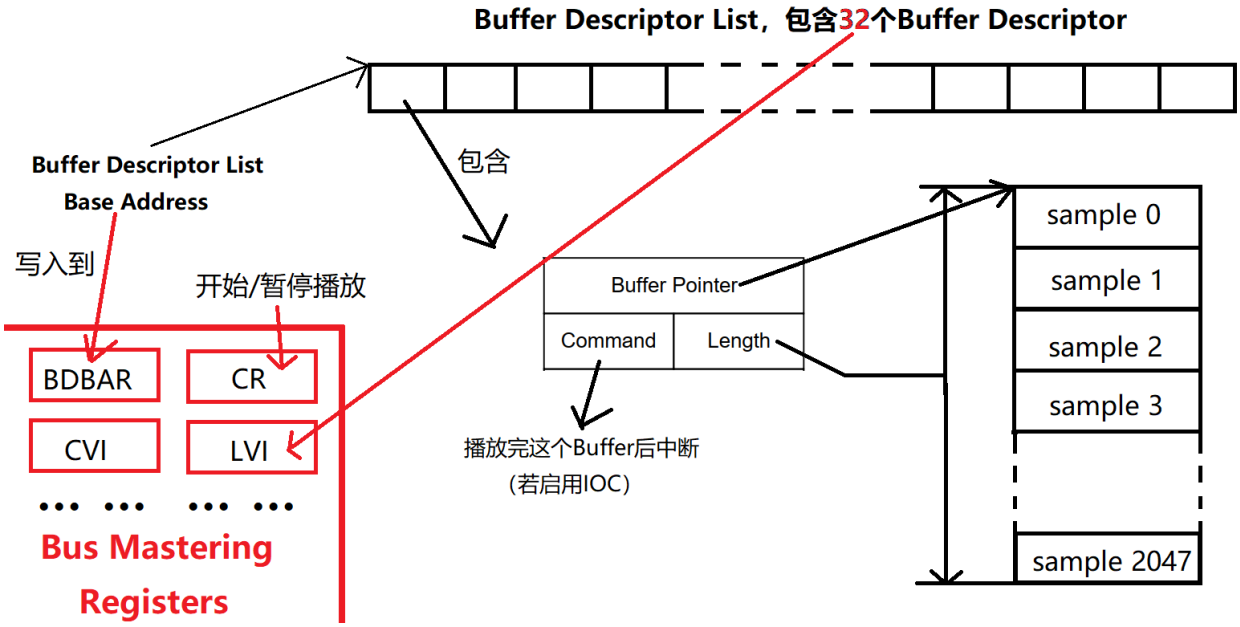


图 2. DMA 引擎示意图

创建了 Buffer 和 Buffer Descriptor List 后，将 Buffer Descriptor List 的地址写入主控寄存器中的 BDBAR，并设置 LVI=31（Last Valid Index，由于有 32 个 Buffer Descriptor 所以最后一个合法的下标为 31）。之后只需要写入 CR 寄存器即可让 DMA 引擎启动，令声卡开始播放 Buffer Descriptor List 中描述的音频数据。CR 寄存器还可以配置 Buffer 播完后中断，便于内核 I/O 软件处理用户提供的 PCM 音频数据。

3.3. 内核 I/O 软件

内核 I/O 软件向用户空间暴露若干设备无关的系统调用接口：

指令	效果
`int setSampleRate(int sampleRate);`	设置 PCM 音频采样率
`int kwrite(void* buf, int length);`	从 buf 开始，长度为 length 的 PCM 数据加入播放队列
`int pause();`	暂停播放（可恢复）
`int stop_wav();`	停止播放并重置缓冲区状态（不可恢复）
`int set_volume(int);`	设置音量（默认 50，范围 0 到 100）

内核 I/O 软件使用 2 个缓冲区，分别储存用户输入的音频数据，以及将送往声卡的音频数据缓冲队列。因此用户不需要关心具体的实现细节，只需要传入 P

CM 格式的音频数据即可。

内核会以两种方式将缓冲队列中的 PCM 数据发送到声卡：当声卡空闲时，用户传入音频后音频被送往声卡并启动 DMA 引擎，以及当声卡繁忙时，用户传入音频后音频加入缓冲队列，在收到 Buffer Descriptor List 空闲的中断后，改写 Buffer Descriptor List，将队列中的音频数据传输到声卡。

描述这一过程的泳道图如下：

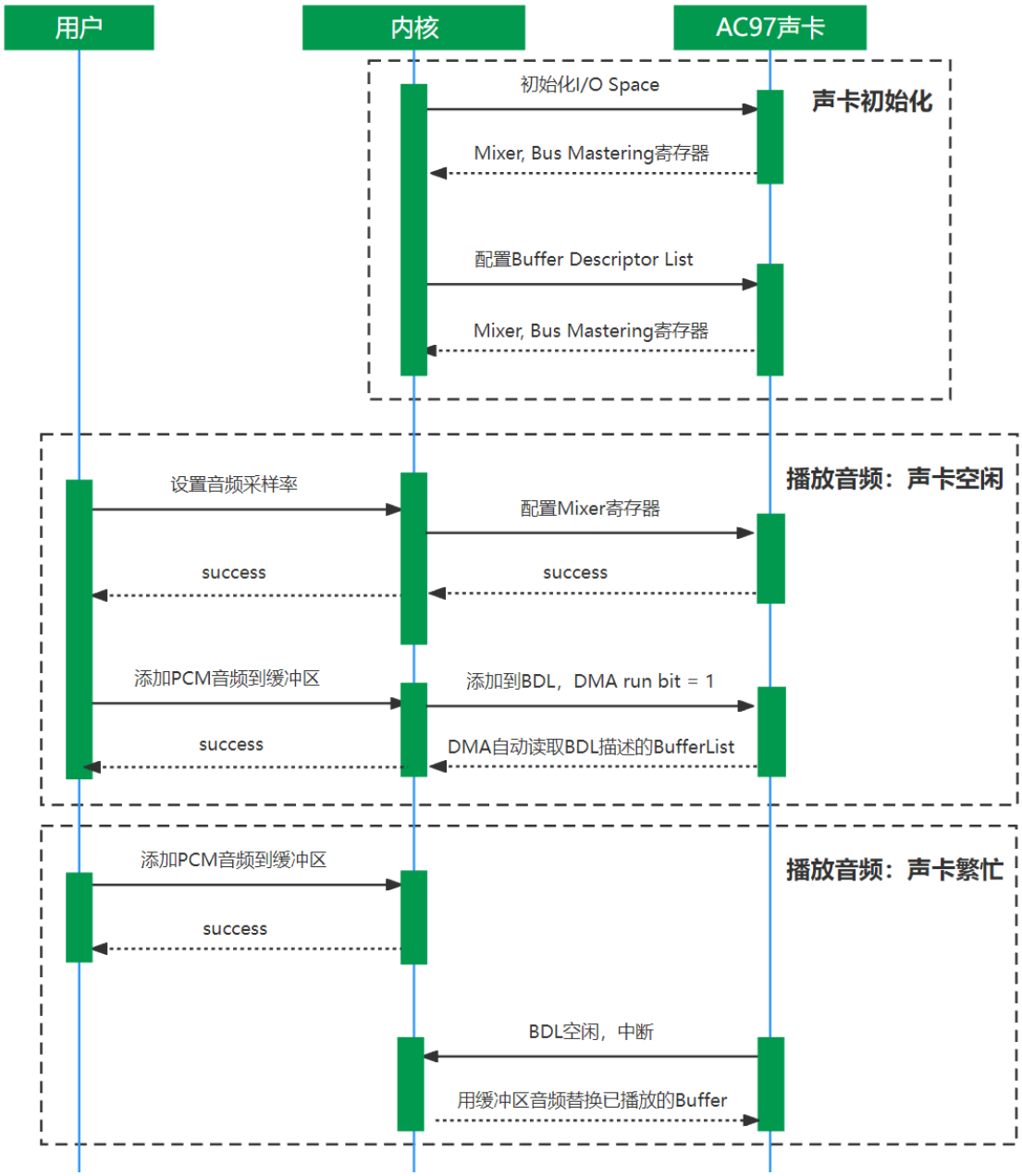
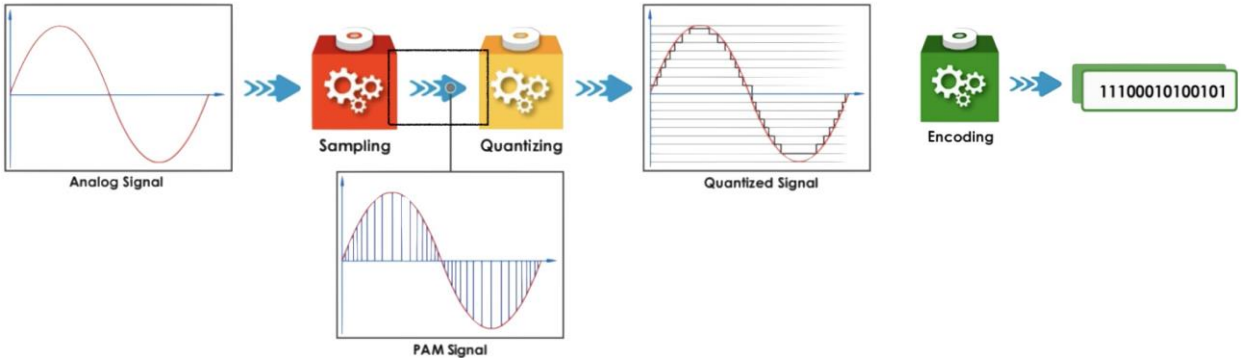


图 3. 内核 I/O 软件控制的 PCM 播放流程

3.4. 音频解码

3.4.1 PCM 基础知识

PCM 指 Pulse-Code Modulation，即用数字信号模拟声音脉冲信号存储数据的一种方式。简单而言，连续的声音信号通过固定频率采样，再经过离散化处理，成为了逐帧的离散数据，并且通过简单的二进制编码成为 PCM 数据。



一般来说，PCM 数据较为重要的信息有采样率、位深、声道数。其中采样率是指采样频率，即采样的间隔时间。位深是指离散化的等级：离散化时每一个采样节点对应的幅值被标记为若干连续等级，从而才可以编码成为二进制数据；一般我们多采用 8、16、24 或 32 位二进制数来记录离散等级，其中采用的二进制位数就是位深。

单声道的 PCM 通过将数据简单串列组织而成。多声道 PCM 存储组织方式通常采用左声道右声道交叉排列的方式，即一个左声道数据串接一个右声道数据，再串接一个左声道数据，依次交叉进行。声卡可以直接读取这样组织的 PCM 数据并播放。

3.4.2 WAV 文件解码

一般常见的 WAV 格式实际上是对 PCM 数据进行的简单封装[1]，以文件头来描述 PCM 数据的关键属性。总的来说，分为 RIFF、Format、数据区三个部分。RIFF 区是文件标识，Format 记录 PCM 的重要参数，如采样率、声道数等；数据区直接排列原始 PCM 数据。因此 WAV 格式的文件是可以通过直接阅读数据区获得任意帧数的 PCM 数据，并直接传递给声卡。

具体编码格式可查阅 Information about the Multimedia file types that Windows Media Player supports (Microsoft Help and Support. Microsoft Corporation).

在 xv6 上实现 WAV 文件播放，只需要先定位到数据区，然后一块块读取 PC

[1] WAV 格式仍可细分为无压缩 PCM 以及 Microsoft ADPCM、IMA ADPCM 等压缩 PCM 格式。常见的.wav 文件一般指无压缩 PCM.

M 数据；在这里，我们每读取 512 字节数据，就通过 kwrite 系统调用向缓冲区写入 PCM 数据。

3.4.3 MP3 文件解码

MP3 是一种有损压缩格式，全称为 MPEG Layer3，于 2017 年失去版权保护进入公有领域。除去文件头等信息，MP3 按照逐帧(frame)的方式排布数据，且每一帧的编码可能都有所不同。具体说来，以单声道为例，MP3 对 PCM 压缩的方法如下：提取 1152 个原始 PCM 数据，通过滤波分子频带、人耳效应优化、频域傅里叶变换、缩放、哈夫曼编码等一系列操作，转化为 MP3 的 1 帧(frame)。因此，实际上 MP3 的 1 帧对应若干原始 PCM 数据，可以通过解压缩，将一小段连续 MP3 数据转化为对应段的 PCM 数据。具体解码过程可以参照

具体而言，我们先通过文件头获取信息，然后通过调用开源的 lieff/minimp3 库，逐帧对 mp3 进行解压缩（单通道 1152 字节数据，双通道 1152*2 字节数据），然后通过 kwrite 系统调用，即时将该帧数据传入缓冲区。因此，我们便实现了**边解压边播放**的 MP3 播放功能。

3.4.4 FLAC 文件解码

FLAC 是一种无损压缩格式，也是一种无版权音频格式。与 MP3 类似，它也采用逐帧(frame)压缩的算法对原始 PCM 数据进行压缩。唯一区别在于 FLAC 的每帧包含的 PCM 数是可变的，在文件头给出。

具体而言，与 MP3 解码类似，我们采用开源的 jprjr/miniflac 库进行解压缩，并且逐帧调用 kwrite 将解码数据写入缓冲区，实现边解压边播放的 FLAC 播放功能。

3.5. 音频播放器

我们封装了各个格式的音频播放，完成了命令行音频播放器。播放器可以循环接收指令，以实现播放、暂停、恢复、展示列表、调节音量等功能。

player 支持的命令如下：

指令	效果
`play filename`	播放音乐
`pause`	暂停播放（可恢复）
`resume`	恢复播放
`stop`	停止播放（不可恢复）

`volume {int 0~100}`	调节音量（默认值 50）
`list`	显示可以播放的音频列表
`exit`	退出音频播放器

通过识别不同文件拓展名来启用相应的播放函数；暂停和恢复功能通过系统调用 `pause()`并设置 `is_paused` 这一全局变量来暂停和恢复向声卡传输数据；停止功能需要先终止播放进程，并通过系统调用 `stop_wav()`来重置缓冲区状态；音量调节通过系统调用 `set_volume()`实现；展示可播放音频的列表通过类似 `ls` 这一用户程序的方法实现，即查询根目录的目录项，寻找其中拓展名为`.wav`、`.mp3`、`.flac` 的文件并打印出来；最后的退出功能则先终止掉 `player` 程序，然后在调用 `stop_wav()`来重置缓冲区状态。

4. 困难与挑战

4.1. 如何访问 I/O 独立编址的 AC97 寄存器

AC97 的主控寄存器和 Mixer 寄存器采用 I/O 独立编址，而 RISC-V 指令集不像 X86 可以使用 IN, OUT 指令访问 I/O 端口，其对 I/O 空间的访问需要通过内存映射实现。

我们查阅 QEMU VIRT-IO 主板的代码，发现其将 I/O Space 映射到 VIRT_PCIE_PIO（0x3000000）开始，长度为 0x10000 bytes 的物理内存上，可以通过读写这段内存，访问 AC97 I/O 独立编址的 Mixer 寄存器和主控寄存器。

```
static const MemMapEntry virt_memmap[] = {
    [VIRT_DEBUG] =        {        0x0,         0x100 },
    [VIRT_MROM] =         {     0x1000,        0xf000 },
    [VIRT_TEST] =         {   0x100000,        0x1000 },
    [VIRT_RTC] =          {   0x101000,        0x1000 },
    [VIRT_CLINT] =        {  0x2000000,       0x10000 },
    [VIRT_ACLINT_SSWI] =  {  0x2F00000,        0x4000 },
    [VIRT_PCIE_PIO] =     {  0x3000000,       0x10000 },
    [VIRT_PLATFORM_BUS] = {  0x4000000,     0x2000000 },
    [VIRT_PLIC] =         {  0xc000000, VIRT_PLIC_SIZE(VIRT_CPUS_MAX * 2) },
    [VIRT_APLIC_M] =      {  0xc000000, APLIC_SIZE(VIRT_CPUS_MAX) },
    [VIRT_APLIC_S] =      {  0xd000000, APLIC_SIZE(VIRT_CPUS_MAX) },
    [VIRT_UART0] =        { 0x10000000,         0x100 },
    [VIRT_VIRTIO] =       { 0x10001000,        0x1000 },
    [VIRT_FW_CFG] =       { 0x10100000,          0x18 },
    [VIRT_FLASH] =        { 0x20000000,     0x4000000 },
    [VIRT_IMSIC_M] =      { 0x24000000, VIRT_IMSIC_MAX_SIZE },
    [VIRT_IMSIC_S] =      { 0x28000000, VIRT_IMSIC_MAX_SIZE },
    [VIRT_PCIE_ECAM] =    { 0x30000000,    0x10000000 },
    [VIRT_PCIE_MMIO] =    { 0x40000000,    0x40000000 },
    [VIRT_DRAM] =         { 0x80000000,           0x0 },
};
```

图 4. QEMU VIRT-IO 主板的内存布局

4.2. 修改 xv6 程序的栈空间大小

MP3 解码进程需要较大的栈空间。尽管可以用 sbrk 系统调用增加进程内存，但栈空间却是固定的 4096B（一个页的大小）。因此，我们修改了 kernel/exec.c 中将程序装载到内存的过程，给栈空间分配了 10 个页。虽然对代码的修改很小但十分有效，解决了 MP3 解码进程爆栈的问题。

需要注意的是，栈空间不足有时并不会给予显式提示，寄存器 scause 会给予 0x2 的错误值(Illegal instruction)。

4.3. 允许更大的音频文件

由于 xv6 采用一级索引的存储管理方式储存文件，且块大小固定为 1024B，这就导致文件大小的上限很小，甚至不足以支持存储很短的音频文件。为此，我们一方面将一级索引拓展为二级索引，另一方面扩大了块大小，从而解决了音频文件大小上限这个问题。

4.4. 解码库的兼容性问题

直接调用开源解码库的接口存在一个很大的问题，即 xv6 并不支持许多常见的 C 标准库。为此，我们一方面通过细致寻找，找到了依赖弱的开源项目；另一方面，对于这些项目的依赖项，我们利用 xv6 已有库重构，并删去不需要的内容——这一系列操作既可以直接删改，也可以借助宏定义等方法直接作出修改。

4.5. xv6 目录项的长度

由于 xv6 的文件系统默认的文件名长度为 14 个字符，所以如果写入系统的音频文件名超过的这个长度，会使得在展示可播放的音频列表时，部分文件名被截断，导致 `list` 功能不能正确识别、显示所有音频文件。因此，我们在写入音频文件时控制了文件名的长度，保证其小于 14 个字符。当然，更好的方法是修改 xv6 的文件系统，使其更加通用。

5. 未来展望

5.1. 优化内存管理和文件系统

在大作业的完成过程中，xv6 原生内存管理方式和文件系统给我们造成了较大困扰：如用户栈只有一页，支持最大文件为 267KB 等，我们计划在之后可以借鉴其他小组大作业成果，将较为先进的内存管理方式和文件系统迁移到我们现有 xv6 系统中，一方面彻底解决我们在播放音频文件时遇到的底层困难，另一方面使用户使用 xv6 系统时更加高效便捷。

5.2. 支持更多格式音频文件

我们在本次大作业中完成了 MP3、WAV、FLAC 三种文件格式的解码播放，但常见音频文件格式还有 ALAC、WMA、OGG、AAC 等，这无疑对用户很不方便。因此，我们计划在之后可以将更多 Linux 下开源解码库迁移到 xv6 系统中，以支持对更多音频文件格式的解码。

5.3. 支持 MP4 视频文件

在实现音频播放的基础上，我们希望更进一步，在 xv6 系统中完成 MP4 视频文件的解码、播放。这一方面需要我们实现类似“图片浏览器”的功能，在 xv6 上进行图像信息的显示；另一方面对于图像和音频两个进程，为了避免音画不同步的情况，我们计划完成进程间通讯的相关算法，如信号量和 PV 原语等，保障视频文件的流畅播放。

5.4. 制作音乐播放器 GUI

现行主流音乐播放器均以图形界面为基础，良好的 GUI 设计可以让用户更好地掌握音乐播放器的相关功能。因此，我们可以以图形界面的方式取代命令行式控制，对用户更加友好。

6. 人员分工

人员	AC97 驱动	系统调用	MP3 音频解码	音频播放器
和嘉晅	√	√	√	√
何承昱	√			√
徐浩博		√	√	
马越洲			√	√

附录 1　　个人信息

姓　名	学　号	手机号码	电子邮箱
和嘉晅	2019010297	18049428151	jx-he19@mails.tsinghua.edu.cn
何承昱	2019010300	15109165493	hecy19@mails.tsinghua.edu.cn
徐浩博	2020010108	15029028230	xuhb20@mails.tsinghua.edu.cn
马越洲	2020011470	13488041102	mayz20@mails.tsinghua.edu.cn

附录 2　　参考资料

[1] Intel® 82801AA (ICH) and Intel® 82801AB (ICH0) I/O Controller Hub DataSheet

[2] Intel® 82801AA (ICH) & Intel® 82801AB (ICH0) I/O Controller Hub AC '97 Programmer's Reference Manual

[3] AC97 - OSDev Wiki

[4] THSS13/XV6 - Github

[5] NebulorDang/xv6-lab-2021 - GitHub

[6] lieff/minimp3 - GitHub

[7] jprjr/miniflac - GitHub