



数字媒体实验二

任务一：对音频文件进行 mp3 编码和解码

(1) 输入一段 PCM 的音频数据，然后采用 MP3 编码器对这段音频文件进行 mp3 压缩和解压缩统计压缩前后文件大小、压缩倍数、压缩时间
代码如下：

```
>ffmpeg -y -f s16be -ac 2 -ar 16000 -acodec pcm_s16le -i 1.pcm 1.mp3|
```

```
C:\Users\86139\Desktop\数字媒体技术\Lab2>ffmpeg -y -f s16be -ac 2 -ar 16000 -acodec pcm_s16le -i 1.pcm 1.mp3
ffmpeg version 2023-09-07-git-9c9f48e7f2-essentials_build-www.gyan.dev Copyright (c) 2000-2023 the FFmpeg developers
  built with gcc 12.2.0 (Rev10), Built by MSYS2 project)
  configuration: --enable-gpl --enable-version3 --enable-static --disable-w32threads --disable-autodetect --enable-fontconfig --enable-iconv --enable-gnutls
--enable-libxml2 --enable-gmp --enable-bzlib --enable-lzma --enable-zlib --enable-libtst --enable-libssh --enable-libzmq --enable-avisynth --enable-d12 --
enable-libwebp --enable-libx264 --enable-libx265 --enable-libxvid --enable-libaom --enable-libopenjpeg --enable-libvpx --enable-mediasfoundation --enable-lib
ass --enable-libfreetype --enable-libfribidi --enable-libharfbuzz --enable-libvidstab --enable-libvmaf --enable-libzimg --enable-amf --enable-cuda-llvm --en
able-cuvid --enable-ffnvcodec --enable-nvdec --enable-nvenc --enable-dxva2 --enable-d3d11va --enable-libvpl --enable-libgme --enable-libopenmpt --enable-lib
opencore-amrwb --enable-libmp3lame --enable-libtheora --enable-libvo-amrwbenc --enable-libgsm --enable-libopencore-amrnb --enable-libopus --enable-lspspeex
--enable-libvorbis --enable-librubberband
  libavutil      58. 19.100 / 58. 19.100
  libavcodec     60. 26.100 / 60. 26.100
  libavformat    60. 11.100 / 60. 11.100
  libavdevice    60.  2.101 / 60.  2.101
  libavfilter     9. 11.100 /  9. 11.100
  libswscale     7.  3.100 /  7.  3.100
  libswresample  4. 11.100 /  4. 11.100
  libpostproc   57.  2.100 / 57.  2.100
[s16be @ 000001b2e04ab480] Estimating duration from bitrate, this may be inaccurate
[istat00/pcm_s16le @ 000001b2e04c3f00] Guessed Channel Layout: stereo
Input #0, s16be, from '1.pcm':
  Duration: 00:01:10.00, bitrate: 512 kb/s
  Stream #0:0: Audio: pcm_s16le, 16000 Hz, 2 channels, s16, 512 kb/s
Stream mapping:
  Stream #0:0 -> #0:0 (pcm_s16le (native) -> mp3 (libmp3lame))
Press [q] to stop, [?] for help
Output #0, mp3, to '1.mp3':
  Metadata:
    TSSE           : Lavf60.11.100
  Stream #0:0: Audio: mp3, 16000 Hz, stereo, s16p
  Metadata:
    encoder        : Lavc60.26.100 libmp3lame
[out#0/mp3 @ 000001b2e04be6c0] video:0kB audio:411kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.053501%
size=      411kB time=00:01:09.98 bitrate= 48.1kbits/s speed= 133x
```

 1.mp3	11/8/2023 2:11 PM	MP3 文件	411 KB
 1.pcm	9/20/2023 5:28 PM	PCM 文件	4,375 KB

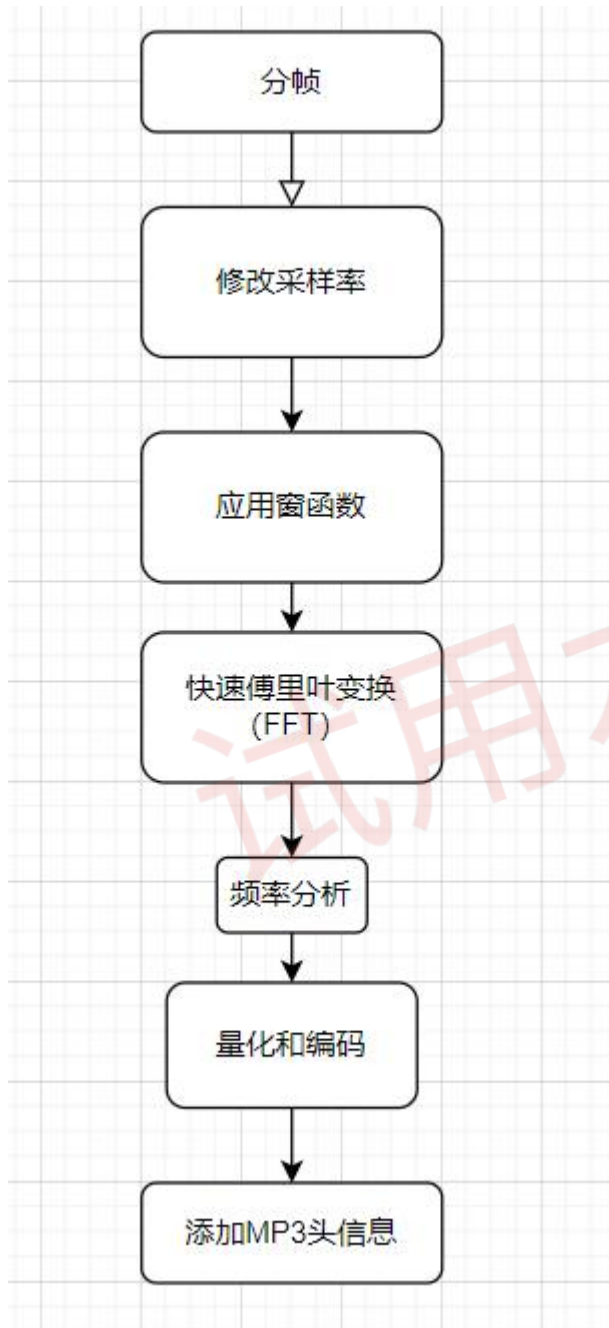
由图可知，压缩倍数略大于十倍，由 4375kb 压缩为 411kb，压缩时间为 00:01:09.98
解压缩命令行代码如下：

```
>ffmpeg -y -i 1.mp3 -acodec pcm_s16le -f s16le -ac 2 -ar 16000 11.pcm|
```

```
C:\Users\86139\Desktop\数字媒体技术\Lab2>ffmpeg -y -i 1.mp3 -acodec pcm_s16le -f s16le -ac 2 -ar 16000 11.pcm
ffmpeg version 2023-09-07-git-9c9f48e7f2-essentials_build-www.gyan.dev Copyright (c) 2000-2023 the FFmpeg developers
  built with gcc 12.2.0 (Rev10), Built by MSYS2 project)
  configuration: --enable-gpl --enable-version3 --enable-static --disable-w32threads --disable-autodetect --enable-fontconfig --enable-iconv --enable-gnutls
--enable-libxml2 --enable-gmp --enable-bzlib --enable-lzma --enable-zlib --enable-libtst --enable-libssh --enable-libzmq --enable-avisynth --enable-d12 --
enable-libwebp --enable-libx264 --enable-libx265 --enable-libxvid --enable-libaom --enable-libopenjpeg --enable-libvpx --enable-mediasfoundation --enable-lib
ass --enable-libfreetype --enable-libfribidi --enable-libharfbuzz --enable-libvidstab --enable-libvmaf --enable-libzimg --enable-amf --enable-cuda-llvm --en
able-cuvid --enable-ffnvcodec --enable-nvdec --enable-nvenc --enable-dxva2 --enable-d3d11va --enable-libvpl --enable-libgme --enable-libopenmpt --enable-lib
opencore-amrwb --enable-libmp3lame --enable-libtheora --enable-libvo-amrwbenc --enable-libgsm --enable-libopencore-amrnb --enable-libopus --enable-lspspeex
--enable-libvorbis --enable-librubberband
  libavutil      58. 19.100 / 58. 19.100
  libavcodec     60. 26.100 / 60. 26.100
  libavformat    60. 11.100 / 60. 11.100
  libavdevice    60.  2.101 / 60.  2.101
  libavfilter     9. 11.100 /  9. 11.100
  libswscale     7.  3.100 /  7.  3.100
  libswresample  4. 11.100 /  4. 11.100
  libpostproc   57.  2.100 / 57.  2.100
Input #0, mp3, from '1.mp3':
  Metadata:
    encoder        : Lavf60.11.100
  Duration: 00:01:10.09, start: 0.069063, bitrate: 48 kb/s
  Stream #0:0: Audio: mp3, 16000 Hz, stereo, fltp, 48 kb/s
Stream mapping:
  Stream #0:0 -> #0:0 (mp3 (mp3float) -> pcm_s16le (native))
Press [q] to stop, [?] for help
Output #0, s16le, to '11.pcm':
  Metadata:
    encoder        : Lavf60.11.100
  Stream #0:0: Audio: pcm_s16le, 16000 Hz, stereo, s16, 512 kb/s
  Metadata:
    encoder        : Lavc60.26.100 pcm_s16le
[out#0/s16le @ 000001da96e525c0] video:0kB audio:4375kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.000000%
size=    4375kB time=00:01:09.98 bitrate= 512.1kbits/s speed= 509x
```

 11.pcm	PCM 文件	4,375 KB
 1.mp3	MP3 文件	411 KB

(2) 理解 mp3 压缩的基本流程，查资料阅读文献画出 mp3 的处理流程图
以下是个人理解的 pcm 转 mp3 的处理流程图：



任务二：对 BMP 图像进行 JPEG 压缩

(1) 针对任意一幅 BMP 图像，采用 JPEG，JBIG 压缩和解压缩代码对其进行压缩和解压缩，然后阅读文献，写出 JPEG 压缩的基本流程

 **1.bmp**

BMP 文件

249 KB

 **1.jpeg**

JPEG 文件

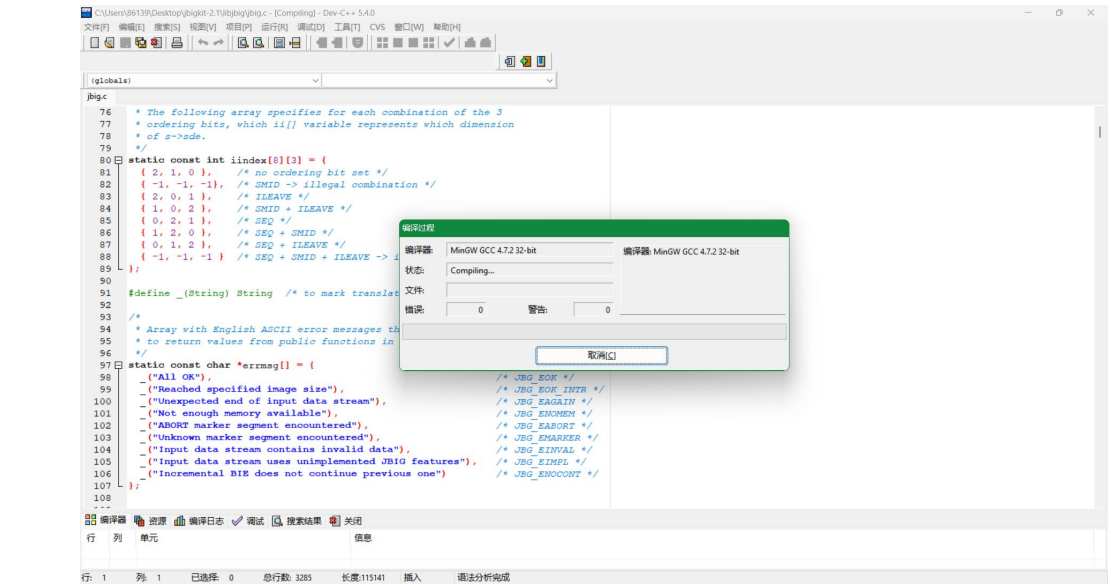
52 KB

```
C:\Users\86139\Desktop\数字媒体技术\Lab2>ffmpeg -i 1.bmp 1.jpeg
ffmpeg version 2023-09-07-git-9c9f48e7f2-essentials.build-www.gyan.dev Copyright (c) 2000-2023 the FFmpeg developers
built with gcc 12.2.0 (Rev10, Built by MSYS2 project)
configuration: --enable-gpl --enable-version3 --enable-static --disable-w32threads --disable-autodetect --enable-fontconfig --enable-iconv --enable-gnutls
--enable-libxml2 --enable-gmp --enable-lzma --enable-zlib --enable-lsrt --enable-libssh --enable-libzmq --enable-avisynth --enable-sdl2 --
enable-libwebp --enable-libx264 --enable-libx265 --enable-libxvid --enable-libaom --enable-libopenjpeg --enable-libvpx --enable-mediafoundation --enable-lib
ass --enable-libfreetype --enable-libfribidi --enable-libharfbuzz --enable-libvidstab --enable-libvmaf --enable-libzimg --enable-amf --enable-cuda-llvm --en
able-cuvid --enable-ffnvcodec --enable-nvdec --enable-nvenc --enable-dxva2 --enable-d3d11va --enable-libvpl --enable-libgme --enable-libopenmpt --enable-lib
opencore-amrnb --enable-libopenmpt --enable-libtheora --enable-libvo-amrwbenc --enable-libgsm --enable-libopencore-amrnb --enable-libopus --enable-lspspeex
--enable-libvorbis --enable-libvorbis --enable-libvorbis
libavutil 58. 19.100 / 58. 19.100
libavcodec 60. 26.100 / 60. 26.100
libavformat 60. 11.100 / 60. 11.100
libavdevice 60. 2.101 / 60. 2.101
libavfilter 9. 11.100 / 9. 11.100
libswscale 7. 3.100 / 7. 3.100
libswresample 4. 11.100 / 4. 11.100
libpostproc 57. 2.100 / 57. 2.100
Input #0, bmp_pipe, from '1.bmp':
Duration: N/A, bitrate: N/A
Stream #0:0: Video: bmp, pal8, 501x502, 25 fps, 25 tbr, 25 tbn
Stream mapping:
Stream #0:0 -> #0:0 (bmp (native) -> mjpeg (native))
Press [q] to stop, [?] for help
[swscale @ 000001a6a117a00] deprecated pixel format used, make sure you did set range correctly
Output #0, image2, to '1.jpeg':
Metadata:
encoder      : Lavf60.11.100
Stream #0:0: Video: mjpeg, yuvj444p(pc, progressive), 501x502, q=2-31, 200 kb/s, 25 fps, 25 tbn
Metadata:
encoder      : Lavc60.26.100 mjpeg
Side data:
cpb: bitrate max/min/avg: 0/0/200000 buffer size: 0 vbv_delay: N/A
[image2 @ 000001a6a117a00] The specified filename '1.jpeg' does not contain an image sequence pattern or a pattern is invalid.
[image2 @ 000001a6a117a00] Use a pattern such as %03d for an image sequence or use the -update option (with -frames:v 1 if needed) to write a single image.
[output#0/image2 @ 000001a6a117a00] video:51kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: unknown
frame= 1 fps=0.0 q=5.4 Lsize=N/A time=00:00:00.00 bitrate=N/A speed= 0x
```

JPEG 压缩的基本流程如下：

- 1.颜色空间转换：将 RGB 图像转换为 YCbCr 颜色空间。Y 是亮度分量，Cb 和 Cr 是色度分量。
- 2.下采样：在人类视觉中，我们对亮度的敏感度高于色度。因此，JPEG 压缩通常会对色度分量进行下采样。
- 3.分块：将图像分割为 8x8 的块。
- 4.离散余弦变换（DCT）：对每个 8x8 的块进行 DCT，将图像数据从空间域转换到频率域。
- 5.量化：使用量化表对 DCT 系数进行量化。量化表通常会对高频分量进行更大的量化，因为人眼对高频细节的敏感度较低。
- 6.Zigzag 扫描：将 8x8 的块转换为 64 维的向量，以便进行后续的编码。
- 7.熵编码：使用如哈夫曼编码等熵编码方法对数据进行编码。

Jbig 编译：



编译过程

MinGW GCC 4.7.2 32-bit

MinGW GCC 4.7.2 32-bit

状态: Compiling...

文件:

错误: 0 警告: 0

取消(C)

任务三：对视频进行 AVS、H.264 和 H.265(HEVC)，H.266 压缩（选一个即可）

本次实验采用 HEVC 压缩，可以看到，文件大小有明显降低。

```
ffmpeg -benchmark -i 1.mp4 -c:v libx265 -preset ultrafast -crf 20 -c:a copy crf20.mp4
```

 crf20.mp4	媒体文件(.mp4)	7,168 KB
 1.mp4	媒体文件(.mp4)	18,560 KB

```
Output #0, mp4, to 'crf20.mp4':
  Metadata:
    major_brand      : isom
    minor_version    : 512
    compatible_brands: isomiso2avc1mp41
    te_is_reencode   : 1
    Hw               : 1
    bitrate          : 16000000
    maxrate          : 0
    encoder          : Lavf60.11.100
  Stream #0:0(und): Video: hevc (hev1 / 0x31766560), yuv420p(tv, bt709, progressive), 1920x1080 [SAR 1:1 DAR 16:9], q=2-31, 30 fps, 15360 tbn (default)
    Metadata:
      creation_time   : 2023-10-14T02:55:54.000000Z
      handler_name     : VideoHandler
      vendor_id        : [0][0][0][0]
      encoder          : Lavc60.26.100 libx265
    Side data:
      cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: N/A
  Stream #0:1(und): Audio: aac (LC) (mp4a / 0x61347060), 44100 Hz, stereo, fltp, 193 kb/s (default)
    Metadata:
      creation_time   : 2023-10-14T02:55:54.000000Z
      handler_name     : SoundHandler
      vendor_id        : [0][0][0][0]
[out#0/mp4 @ 0000022635b6fe80] video:6760kB audio:386kB subtitle:0kB other streams:0kB global headers:2kB muxing overhead: 0.302270%
frame= 489 fps= 20 q=28.0 Lsize=      7168kB time=00:00:16.27 bitrate=3607.4kbits/s speed=0.678x
bench:  time=130.922s stime=1.547s rtime=24.026s
bench:  maxrss=832920kB
x265 [info]: frame I:      2, Avg QP:23.87 kb/s: 7276.20
x265 [info]: frame P:    132, Avg QP:25.38 kb/s: 5348.27
x265 [info]: frame B:    355, Avg QP:27.66 kb/s: 2649.05
encoded 489 frames in 23.99s (20.38 fps), 3396.60 kb/s, Avg QP:27.01
```