University of Cape Town, Computer Science

CSC2002S, Computer Architecture Assignment 1

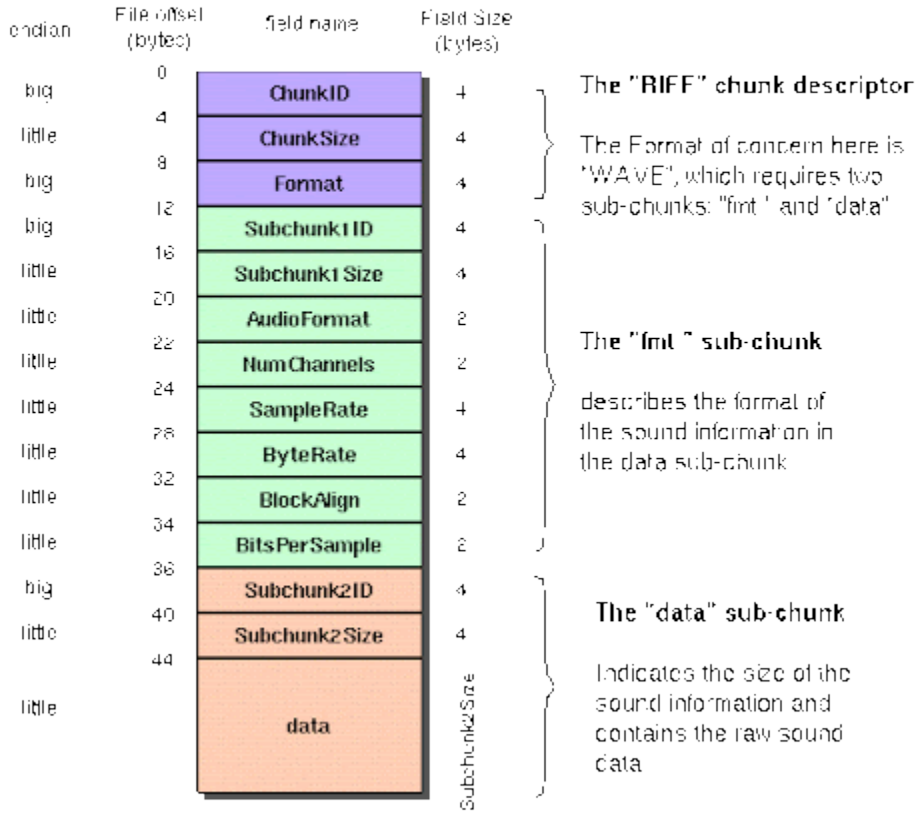
100 marks

Sound files come in many forms. There are different codecs for storing audio data and there are different compression algorithms used for minimising the size of files. In this assignment, we will explore WAVE files.

WAVE (Waveform Audio File Format) files were developed in 1991 for Microsoft PCs. They store uncompressed audio data using linear pulse-code modulation.

The format of a wave file is as follows:

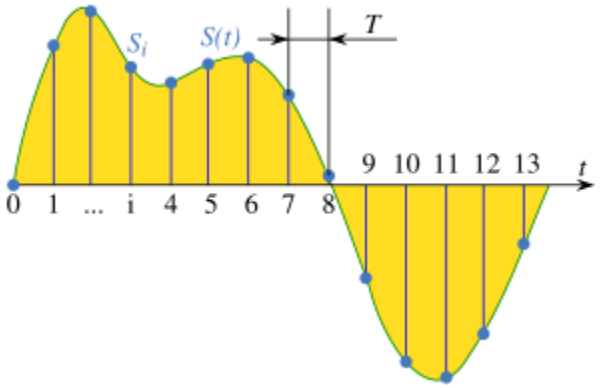
The Canonical WAVE file format



From above, we see that the first 44 bytes of a wave file is the header. This provides information about the wave file, which is required by an audio player to play the file. All the bytes after the header, is the actual audio data.

The actual audio data is represented through linear pulse code modulation. This is a way of digitally representing an audio signal.

Raw audio signals themselves represent sound waves. Below, we see an example of an audio wave:



The green line represents the amplitude of the sound, or pressure variation caused by the sound, which we see changes over time. The variation of this is what creates the sounds we hear.

This signal is, however, analogue. We need a way to discretise this signal in order to store it using values in a file. This is known as sampling. For a point in time, we measure the sound signal and store this value obtained – this is then repeated, represented by the blue dots above.

There are 2 factors to consider when sampling a file:

1. The **sample rate** – how many times per second the sound signal is being measured (in Hz). In the WAVE file header, you can find this information at address 24.
2. The **sample bit depth** – how many bits is used to store the value obtained from the sample. In the WAVE file header, you can find this information at address 34.

Additionally, we can store 2 or more audio forms in a single file. A common use case for this is when we need to have separate audio for each ear – each side, left and right, will have its own audio form in its own **channel**. The number of channels in a WAVE file is declared in the header as well.

The byte rate (also present in the header) tells us how much information is being processed a second and is defined as:

$$byte\ rate = sample\ rate * number\ of\ channels * bytes\ per\ sample$$

In this assignment, we will explore interpreting WAVE file headers, generating sound effects and synthesising WAVE files using MIPS assembly.

Question 1: WAVE Header Interpretation [20 marks]

Write a MIPS program, called **question1.asm**, that will tell us information about a wave file by reading its header. Your program should prompt the user to enter the **full file name** (the entire path) and the **file size** (in bytes) as input. The program will then output information about the file's number of channels, sample rate, byte rate and bits per sample.

Sample IO:

```
Enter a wave file name:
/home/lynn/Documents/CSC2002S/assign1/q1_t1_in.wav
Enter the file size (in bytes):
132344
Information about the wave file:
================================
Number of channels: 1
Sample rate: 22050
Byte rate: 44100
Bits per sample: 16
```

Question 2: Audio Data Analysis [20 marks]

Write a MIPS program, called **question2.asm**, that will read in a wave file, and then find the maximum and minimum amplitude (highest and lowest values) in the audio data. You will need to prompt the user to enter the **full file name** (the entire path) as input to the program, and the **file size** (in bytes). You may assume that all audio files inputted will only have 1 channel and will have a bit depth of 16.

Sample IO:

```
Enter a wave file name:
/home/lynn/Documents/CSC2002S/assign1/q2_t1_in.wav
Enter the file size (in bytes):
64
Information about the wave file:
================================
Maximum amplitude: 16
Minimum amplitude: 1
```

Question 3: Sound Effects [30 marks]

Many different effects can be used to transform audio. You are required to implement a simple reversing algorithm. Write a MIPS program, called **question3.asm**, that will read in a WAVE file, and will then reverse the audio data in the file, and output this to a new file. You will need to

obtain, from the user, the **full input file name** (the entire path), the **full output file name** (the entire path) and the **file size** (in bytes). You may assume that all audio files inputted will only have 1 channel and will have a bit depth of 16.

Note: for this question, the Automarker will not check your console output (and there should not be console output, only input). It will only check your output file. When it prints your output file, it will convert your binary file to a textual hexadecimal form.

Sample Console **Input**:

```
/home/lynn/Documents/CSC2002S/assign1/q3_t1_in.wav
/home/lynn/Documents/CSC2002S/assign1/q3_t1_out.wav
64
```

Sample File **Input** (represented as text using xxd):

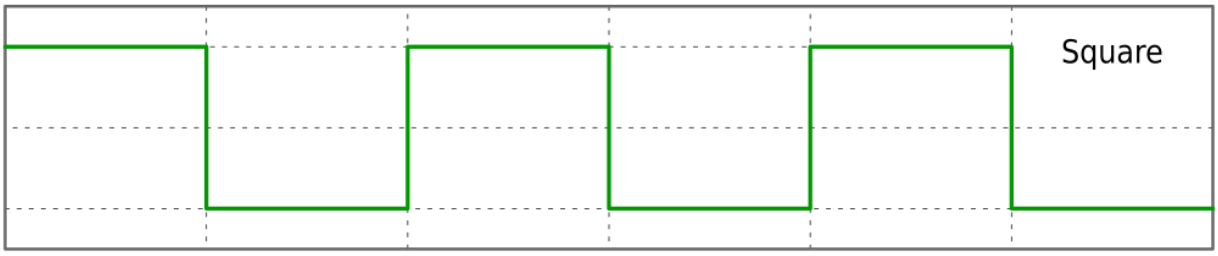
```
00000000: 52 49 46 46 f0 04 02 00 57 41 56 45 66 6d 74 20
00000010: 10 00 00 00 01 00 01 00 22 56 00 00 44 ac 00 00
00000020: 02 00 10 00 64 61 74 61 cc 04 02 00 00 00 01 00
00000030: 02 00 03 00 04 00 05 00 06 00 07 00 08 00 09 00
```

Sample File Output (represented as text using xxd):

```
00000000: 52 49 46 46 f0 04 02 00 57 41 56 45 66 6d 74 20
00000010: 10 00 00 00 01 00 01 00 22 56 00 00 44 ac 00 00
00000020: 02 00 10 00 64 61 74 61 cc 04 02 00 09 00 08 00
00000030: 07 00 06 00 05 00 04 00 03 00 02 00 01 00 00 00
```

Question 4: Sound Synthesis [30 marks]

Square waves are particularly useful in electronics as they can represent low and high states (they only have 2 states). We can also create audio forms with square waves. Like any periodic waveform, square waves will have an amplitude and frequency. Below is an example of a square wave.



In WAVE files, this square wave can be represented using the minimum and maximum values of bit depth, i.e. in a WAVE file with a bit depth of 16, high states will be represented with the number 32767 and low states will be represented with the number -32768.

Write a MIPS program, called **question4.asm**, that will generate square waves. You will need to obtain, from the user (in this order), the **full output file name** (the entire path), the **tone frequency** (how many times a wave period will appear within a second, an integer), the **sample frequency** (how many sound samples will be measured in a second, an integer), and the **length of the tone** (an integer in seconds).

Your output file should have 1 channel only and a bit depth of 16. You should ignore the contents of the header: set all 44 bytes of the header to 0. Your tone should always start with the high state. You may assume that the tone frequency, sample frequency and length will be such that there will always be an even number of samples per wave period.

Note: for this question, the Automarker will not check your console output (and there should not be console output, only input). It will only check your output file. When it prints your output file, it will convert your binary file to a textual hexadecimal form.

Sample Console **Input** (do not place the comments in the program input):

```
/home/lynn/Documents/CSC2002S/assign1/q4_t1_out.wav
2  # tone frequency
20 # sample frequency
1  # length of the tone
```

Sample File Output (represented as text using xxd):

```
00000000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 ff 7f ff 7f
00000030: ff 7f ff 7f ff 7f 00 80 00 80 00 80 00 80 00 80
00000040: ff 7f ff 7f ff 7f ff 7f ff 7f 00 80 00 80 00 80
00000050: 00 80 00 80
```

Quest for the Curious [0 marks]

The WAVE files you have created in Question 4 may not play (or if they do, they may not play correctly) because they lack a WAVE header, which gives the audio player information about the type of audio in the file. Without the header, the audio player will not know how to interpret the audio data.

To resolve this, using the table above, which describes what values should be placed into the header, you could programmatically insert header information or manually type it into the WAVE file using a hex editor. However, do **NOT** submit this to the Automarker (it has been configured to only mark zeroed out headers as correct). Once you have done this, your synthesised square waves should play!

Submission and Marking

Compress all your question solutions into a **zip** file (there should be no subdirectories within the zip file), labelled with your student number, and submit this to the Automarker.

The Automarker will test your solutions with 10 trials, with differing input. You will be able to view the expected and actual output of your program for the first 5 trials in each question. The expected and actual output for the second 5 trials will be hidden.

You must not hardcode solutions.

Useful Information

NOTE: You **MUST** use a Linux PC to run your MIPS programs. File IO **WILL NOT** work as expected on other OSs.

You may use the Linux tool `xxd` to write a binary file as text (in hexadecimal form). Usage example for displaying a binary file as text in hexadecimal form onto the terminal:

```
xxd -g 1 <binary.file>
```

Usage example for converting a binary file to text in hexadecimal form and saving this in another file:

```
xxd -g 1 <binary.file> > <text.file>
```

You may also use the Hex Editor extension in Visual Studio Code (ID: ms-vscode.hexeditor), to display binary files in textual hexadecimal form.

Audacity[1] is an audio editing software and is useful for visualising audio forms.

[1] Audacity ® | Free Audio editor, recorder, music making and more! (audacityteam.org)