Design document

Name: Yixi Rao
UID: u6826541

Design overview: In this project, I created an advanced sequencer which can play a sequence of notes and therefore plays a song, and the user can custom their notes in four dimensions which are the frequency of the note, the amplitude of the note, the duration of the note and also the waveform of the note, by the way, the MIDI format is also supported when user want to type it on this format. And the MIDI format starts at 0 to 127.

Implementation: The program structure and the organization I used in this program is dividing the whole program and the codes into three different parts. The first part is the data part which can store the information of the note which is described above, and it is stored in an array data structure in assembly, the four kind of information is stored line by line (one line just stores one kind of information). The main part of the program is the sequencer part, within the sequencer part, lots of functions that are written in the function part are used to generate different synthesizer by using the information in the data part. At last, the function part is programmed to implement two kinds of functionality, the first functionality is the generating the sound of four waveforms and the silence effect and output it, the first functionality is achieved by four functions each will create a specific waveform sound, and second functionality is to transform the MIDI code to the correct frequency.

The data part is like an array data structure, and it is separated by the array name (label in assembly), each array can store four kinds of information and it is arranged by four-line, each line only stored one kind of information. The first one is the frequency, and the unit is Hz, the range is 0 to 24000, by the way, MIDI format is also supported, and the format is 0x200000xx, the last byte is the MIDI code. The second one is the amplitude, which dynamic range is 0x7fff to 0x8000. The third one is the waveform, 0xa stands for a triangle wave, 0xb is a square wave, 0xc is a sawtooth wave. The last one is the duration of which unit is Ms.

The implementation of the main part (or the sequencer part) is attained by using a function with parameters of the number of notes wanted to play and the base memory address of the sequence. The functionality of this function is to play the sequence until reach the end. Inside the function, there is a "playNote" while loop which can play one note each iteration until the index reaches the end of the sequence, the index will increase each iteration. Now within the "playNote" loop, it also has multiple parts, which is divided by labels, and each label will transfer the program to different function which can generate different waveform or just a silence effect. How does this "playNote" loop determines which label should the program jump to? The solution is the frequency can be a judgement criterion, I defined the 0 frequency as the silence effect and 0x20000000 or greater frequency is the MIDI format that the last byte of the format is the MIDI code, If the program is going to audio part then we can use the waveform information to determine which label should go to and then finally generate the specific sound. Moreover, the frequency, the duration, the waveform, and the amplitude information are extracted from the special memory address by using the base address and the given parameter of the number of the notes and the calculated memory offset.

The function part has four functions, the first three function are designed to generate to form three kinds of the waveform, their structure is mostly the same and they have already described in the last design document. The biggest modification is that I made it functional and the duration is added, here

I will explain about the duration part in this function, which just simply adds a while loop inside the synthesizer loop and the while loop will stop when a special number goes to 0, the special number is the time(s) multiplies the frequency.

Analysis: The reason why I chose the divided part program structure is that it can make the code more distinct and intelligible, also the reader can easily understand the whole program by just skimming through it, the other reason is that one part of code will not confuse the other part of the code and the same time the different section can be connected like a chain to work together to achieve the objective.

The data structure I chose is the static array, and it is very appropriate for this program because the sequence is just a list of number, so we can store the value in the array, and the sequence can be modified easily, furthermore, it is a static variable, because the objective of this sequencer is to play a sequence not create a sequence, therefore the size and the content of the sequence are been decided before the run-time, so there is no need to use the dynamic variable. the other aspect of the array I think it is very appropriate and correct for this program is the arrangement of the array, each line only stores one kind of information, not like other approaches such as store all the information on just one line or using multiple labels to store different information my approach is more concise and clear and it is also space-saving, since in each column of the array, it lists all information of one note and it is easy to modify, and it can also deduce the number of parameters the sequencer function when compared to the approach of using multiple label storing information, because the next line address can be inferred by the memory offset.

The algorithm of the sequencer is also proper, because the while loop in it will iterate through all the notes, and the labels in the loop can promise that all the situations can be considered and solved, it will not omit any other situation of the waveform, in other ways, the while loop is a good way to save the space because it can delete a lot of repetitive code. The other decision I made in this part is the order of the judgement criterion, my order is first determining whether it is a silence effect or it is a sound wave, and then it will determine the waveform of the note, the reason behind that is it can save space and increase the efficiency, and the principle is that it can reduce the label and the depth of the function.

In the implementation of the duration of the function part, I use the method of the duration time(s) multiplies the frequency to attain the effect of the duration, which is correct because it is the principle of the relation between time and frequency, as long as the frequency is correct, then the duration can be guaranteed to be correct. Also, this will not generate some kind of semi-finished wave since float is not allowed in assembly. My implementation of the MIDI translate function is proper but not very precise, in that function, which just tries to implement a math formula $(440 * 2\text{^}((m - 69)/12))$, the most difficult part is how to deal with the logarithm, and the 2 to the power of float number, my solution is that enumerating the frequency between 57 and 69, 69 to 81 as well. It can perfectly solve the logarithm with prefect precision, however, the solution of 2 to the power of float number is still needed to improve, my solution is just using the common division, but assembly will round it to an integer, that is the cause of the imprecision, with a deviation about 0 t0 200hz.

Lastly, my sequence design is also correct and fully demonstrate the sequencer functionality, which is comprise of a song and in this song the waveform is changing, frequency and amplitude are changing as well and also some MIDI code format notes.