COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGES FINAL PROJECT

PIANO SOUNDS

Content

- **Outline 1**
- **Motivation 1**
- **How To Use 1**
- **Implementation 2**
- **Conclusion 4**
- **Acknowledgement 4**
- **References 4**
- **Appendix ------ About DOSBox 5**

資工二 **b95902065** 楊瑩綺 **b95902104** 林君轂

Outline

Use the computer keyboard as the piano keyboard to produce simple notes and rhythm. We can even observe the whole procedure through the keyboard image which changes instantly.

Motivation

Computer keyboard and piano keyboard have something really alike. Thus we try to make the specific sounds through the computer speaker implementing by Assembly Language. There may be some kinds of similar programs to do this interesting task, and we act it with the great language we had learned. Now just enjoy it!

How To Use

1. **Once** users enter the executive screen, they can see the keyboard image in the bottom of the screen.

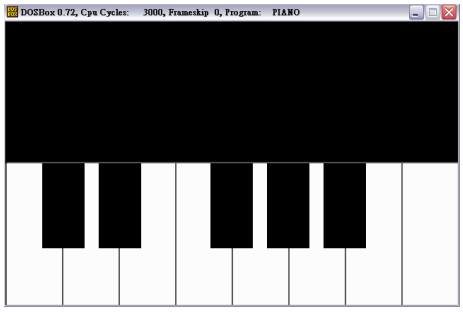

Figure 1. Initial Interface

2. Users now can hit one of the following keys

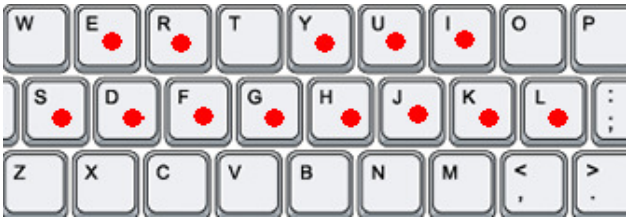

Figure 2. Keys

Key	S	E	d	R	f	g	Y	H	u	j	i	k	l
Tone	Do	Do#	Re	Re#	Mi	Fa	Fa#	So	So#	La	La#	Si	Do

Table 1. Keys and correspondent notes

3. Then users can hear the correspondent note sound and the note which is designated by the users will show a red dot on the correspondent keyboard location instantaneously.

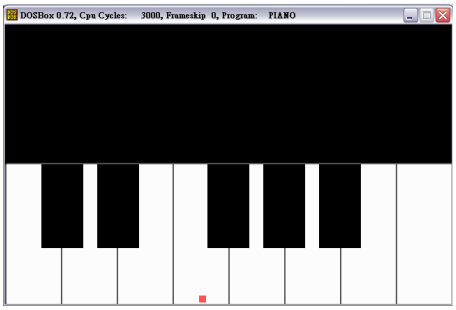

Figure 3. After pressing the key

4. And users are allowed to repeat the1.-2.steps to play a length of melody at user's will.
5. If users want to exit the program, just press the space key. That is to say, the space key is designed to end it.

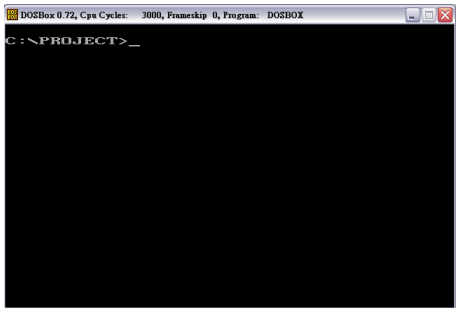

Figure 4. Exit the program

Implementation

We use the DOSBox to implement the whole program with assembly, including sound and image aspects. We want to touch BIOS which is protected by the Windows indirectly. So, we use the emulator---DOSBox to simulate the behavior.

And we implement this project as two parts---sound and image. Following steps may describe detailed state to let us know more about the project.

The first part is about sound. There are three important functions in this file which are main() keyword() and sound().

First, we define each note with the authorized digits hertz unit respectively in an array.

Note	Do	Do#	Re	Re#	Mi	Fa
f(Hertz)	261.6	277.2	293.7	311.1	329.6	349.2

Fa#	So	So#	La	La#	Si	Do
370.09	392.0	415.3	440.0	466.2	523.3	523.3

Table 2. Notes and frequencies

After user press a key, the function keyword() will compare the user's input and the element in the defined array. If the input keyword can be found in the array, the main() will turn the speaker on, else it will not do. As the speaker turns on, call the function sound() to produce sound. And if users press the space key, the program will exit and turn off the speaker and then end main().

The second part is about image. There are three small functions to assist whole drawing work---gray() white() and black() in this file, which mean "draw a gray point" "draw a white point" and "draw a black point".

First, we draw the keyboard image with the coordinate system and the color is consulted from the following table.

Value(2)	example	Value(2)	example
0000	black	1000	Gray
0001	blue	1001	light blue
0010	green	1010	light green
0011	cyan	1000	light cyan
0100	red	1100	light red
0101	magenta	1101	light magenta
0110	brown	1110	Yellow
0111	light gray	1111	White

Table 3. Four digits color code

Here we just use the three simple colors--- gray, black and white to sketch the keyboard image. The gray color is used to depict the edge of the keyboard, and the black and white colors are just like the piano keyboard colors.

Then, divide and conquer! We divide the keyboard into two parts---upper and lower ones. And then use a series of operations to calculate each point's exact

coordinate location and tone the needed color to the point. Note that use combination is also a smart way to carry out efficiently. Thus, the keyboard image is finished.

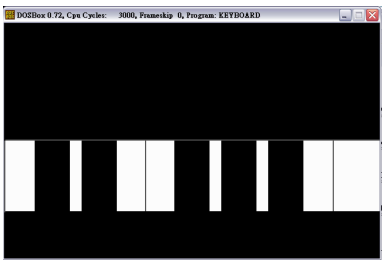

Figure 5. Upper keyboard

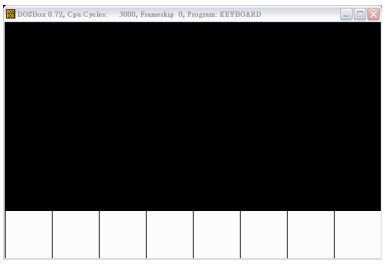

Figure6. Lower keyboard

Conclusion

Using this project program can play simple songs with single note definitely. This is already a nice performance and also an interesting experience for us. There may be some improvements to let it be more powerful to play the chords (to have multiple notes at the same time) and draw the staves with correspondent notes. And that is what we are struggle for the next step. Through this project, we learn the skill to simulate it and do all the things under this basic language. We firmly believe it can be improved further more. Advice is the power motivating us to advance it. Thank you for watching this project!

Acknowledgements

李根逸 TA　　許晉東 同學　　黃禮俊 學長

References

<<Assembly Language for Intel-based Computers>> KIP R. IRVINE
http://home.educities.edu.tw/wanker742126/asm/ap07.html
http://www.protw.net/index.php/content/view/77/91.html
http://en.wikipedia.org/wiki/Dosbox
http://ftp.isu.edu.tw/pub/Windows/Chinese/txt/hard/0339.txt
http://delphi.ktop.com.tw/board.php?cid=173&fid=1167&tid=92219
http://www.phy.ntnu.edu.tw/demolab/phpBB/viewtopic.php?topic=296
http://www.seoconsultants.com/windows/keyboard/images/keyboard4.png

Appendix ------ About DOSBox

DOSBox is an x86 emulator which mimics a DOS command-line environment intended for running MS-DOS-based IBM PC compatible programs, especially computer games, which may not run properly on newer PCs and may not run at all on non-IBM PC compatibles (e.g. PowerPC Macintosh). DOSBox is open source and available for many operating systems, such as Linux, OpenBSD, FreeBSD, Windows 9x, Windows NT 4.0, Windows 2000, Windows XP (32-bit and 64-bit), Windows 2003, Windows Vista, Mac OS X, OS/2, Palm OS, RISC OS, and BeOS. It has also been ported to PlayStation Portable and the GP2X device.

DOSBox has been used by software companies to redistribute MS-DOS-native games in a format which can be run on Windows NT-based operating systems. Most recently, Steam interactive has teamed up with id Software to distribute various DOS games such as Wolfenstein 3D and Commander Keen bundled with DOSBox.

As with most emulators, DOSBox requires substantially more computing (particularly processor) power than the original systems, and is greatly affected by what software the user is running in the emulated system at the time. This weakness can be sometimes greatly reduced by using an open source substitute of the most common protected mode memory extender DOS/4GW (the much smaller and faster DOS32a). Moreover, DOS programs that run in protected mode, which include most games released after 1995, may not perform as well as in other emulators such as VMware or Virtual PC, since those programs mostly virtualize the processor instead of emulating it like DOSBox.

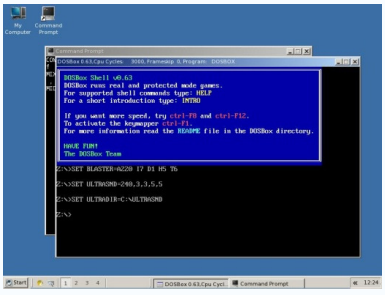

Figure 7. Dos Mode

Figure 8.9. classic DOSBox games