Contents

[04/11/2023 – 4 hours 2](#_Toc150706301)

[06/11/2023 – 3 hours 2](#_Toc150706302)

# 04/11/2023 – 4 hours

Today I started off by cloning the starting template for the assignment and getting qemu working on my personal PC. Once done, I then began attempting to use the int 16 bios interrupt. This is the interrupt used to retrieve a keyboard input from a user. At first I was having issues with this as I didn’t fully understand the documentation for the interrupt table. Figure 1 shows the documentation for get keystroke, in order to use this, I had to pass in the value `00` into the AH register. When trying this I got an error which stated that there was a junk “h” after the expression, this is because I had already prefixed the `00` with 0x which denotes that it is a hexadecimal value, so removing the H resolved the problems.

A close-up of a computer code

Description automatically generated

Figure – interrupt documentation for keyboard input

# 06/11/2023 – 3 hours

Figure 2 shows me being able to type the number of the sector to read, this is currently not being used but I am creating the functionality for it. The next stage is to also display the sector number in the message “Contents of sector”, to do this I will be temporarily storing the contents of al inside of the stack, this will allow me to reuse it after the other console logs have been ran.

A screenshot of a computer

Description automatically generated

Figure - keyboard input working

When attempting to use the pushw instruction I ran into an error **“Error: unsupported instruction `push'”,** this is because I was using the %al register which is not supported with this instruction. After reading the lecture slides for week 3 it stated that you can only push the whole register onto the stack, once I changed it to use %ax I no longer received the error.

**Another issue which I have ran into is with the order that I was pushing and popping. I was using the pushw instruction after I call cons\_write\_crlf so the value in %al was being overwritten by the special characters. To fix the problem I had to use the pushw instruction before I called cons\_write\_crlf in order to save the correct value (the keyboard input)**

**A screen shot of a computer

Description automatically generated**

Figure - bug showing the wrong value being popped onto the register

**Figure 4 displays the code used to output the contents of figure 5.**

**A screenshot of a computer program

Description automatically generated**

Figure - The final code for keyboard input and contents of sector message

Figure 5 displays the final output for the code above. The next stage is to start reading the contents off of the disk.

A black screen with white text

Description automatically generated

Figure - final output of keyboard input label

# 10/11/2023 –2h 14m

**Today I had a bug where if I had a label under my `contents\_of\_sector\_message` it would output that to screen when it shouldn’t. This can be seen in figure 6.**

**A screenshot of a computer

Description automatically generated**

Figure - bug showing extra label being print to screen

After much time debugging and trying different things, such as adding a terminator value to the end of the content of sector string (even though .string adds this automatically) I was able to fix the issue. However, I didn’t quite understand why this was happening, so I had a look into it some more and later found that because I was calling cons\_writeline without moving a value into %si there was no null terminator, this caused the function to loop go through the wrong memory and access strings which it was not supposed to, the code for this can be seen in figure 7.

A screenshot of a computer program

Description automatically generated

Figure – code

To fix the problem in figure 7 I modified the code to include an empty string, which would act as a null terminator to stop the cons\_writeline going into the wrong area of memory. I later realised that cons\_writeline was for outputting not just a new line, but also the value in %si, so it I could just call cons\_write\_crlf to output the newline instead of calling cons\_writeline, which was for a different purpose.

The next stage was to start reading the disk, the initial code for this gave me some problems as I was just calling cons\_write\_hex, without passing any information into the register. This can be seen in figure 8.

A screenshot of a computer

Description automatically generated

Figure - cons\_write\_hex not outputting any data

In order to fix this problem I had a look at the code for outputting hex values, the register which was being used was BX. To get some initial output I moved the bytes that were inside of the memory address “0xD000” into the BL register. I then got the output in figure 9.

A screen shot of a computer

Description automatically generated

Figure - values being read from disk

A computer screen with white text

Description automatically generated