Matthew Maloof
Task 4: ARM Assembly Programming

First step is to go into the terminal and type nano second.s to create a file named second.s and edit the contents. Figure 1 shows the contents of this file. Then you press control+x, y, then enter to save the file and exit the editor.

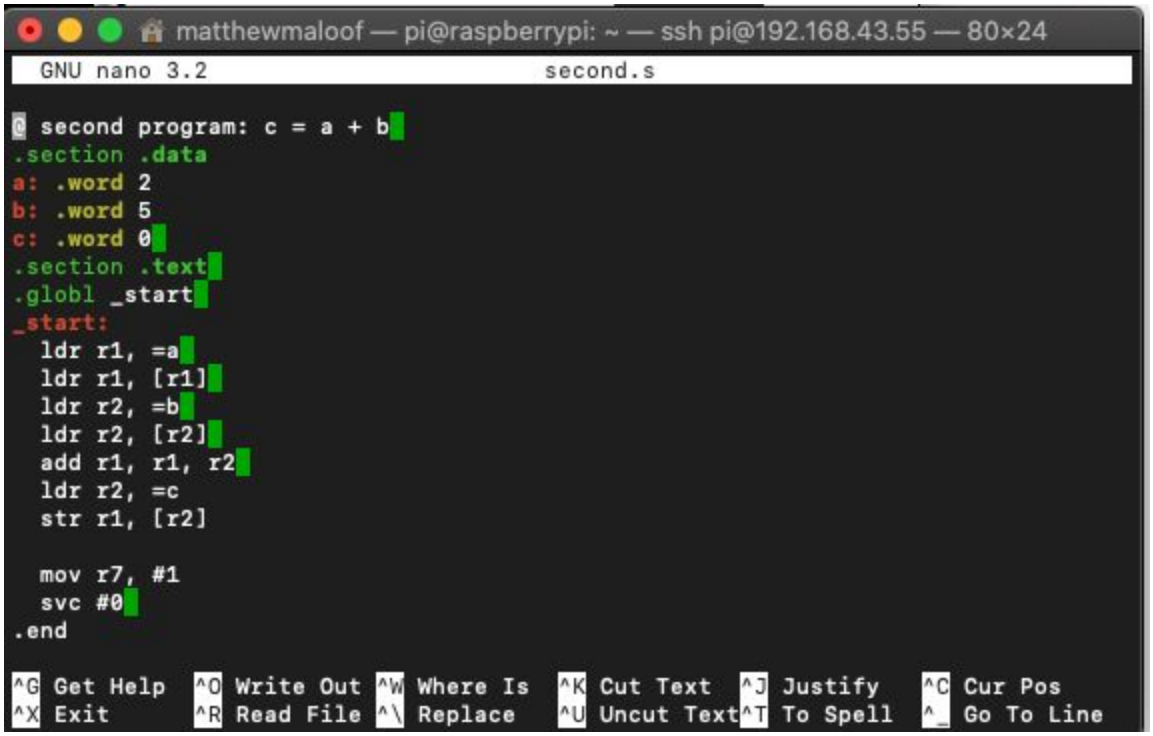

Figure 1: Code for second.s file

We then type as -o second.o second.s to assemble the file, ld -o second second.o to link the file and ./second to run the file that is now an executable. As seen in figure 2 below, there is no output for this program which is expected because no output was set in the code, it is just running a program. It'd be the equivalent of having a calculator with no screen, it would still do the calculation but no output would be given/shown for the user.

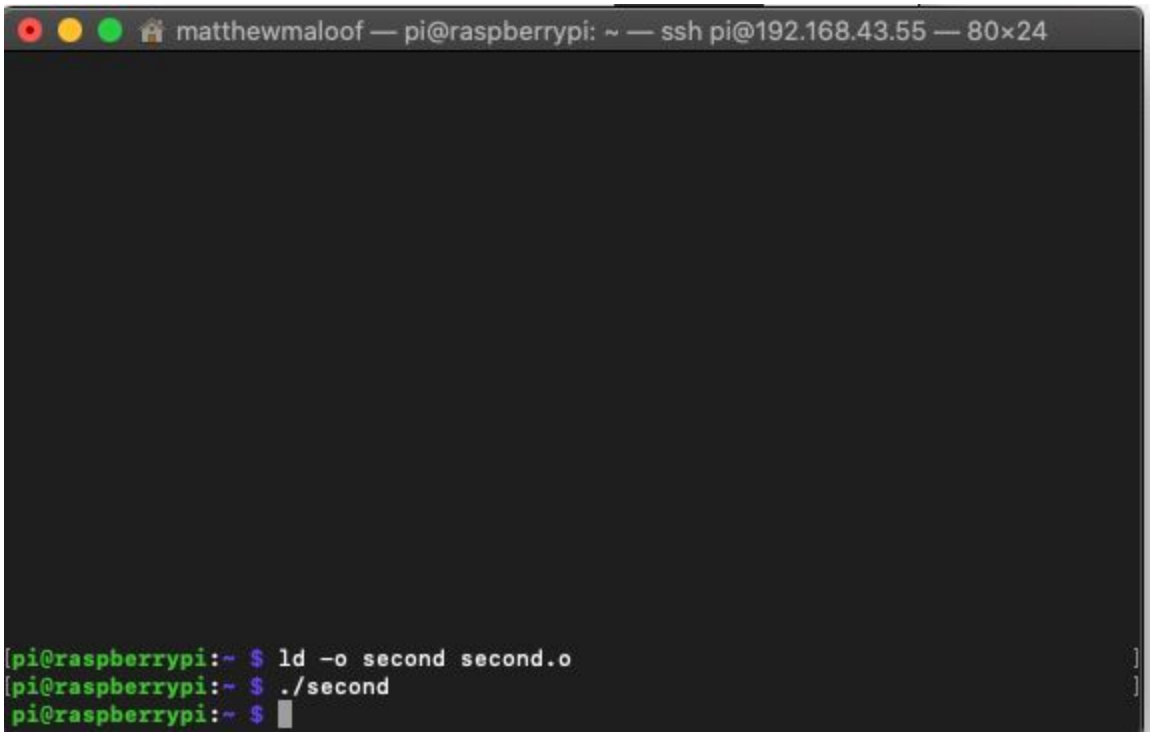

Figure 2: Turn into executable no output

We then type gdb second to enter the GNU debugger.

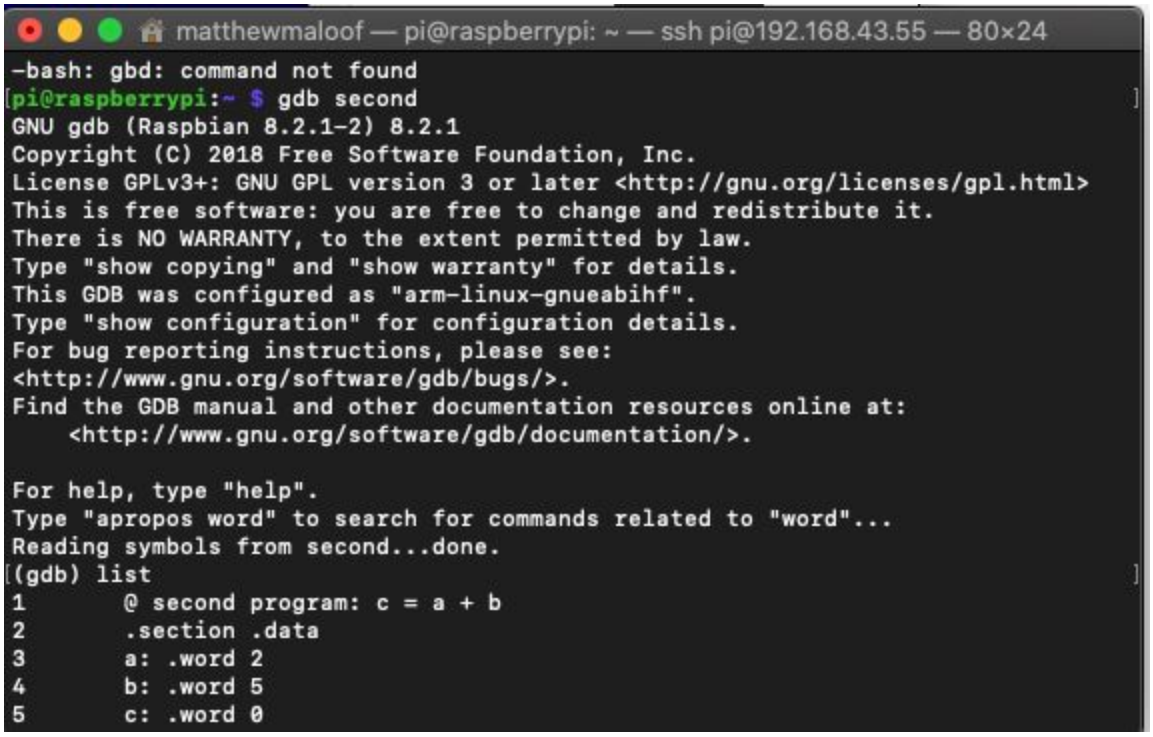

Figure 3: GDB into GNU debugger

We then set a breakpoint at line 15 by typing b 15 then type run.

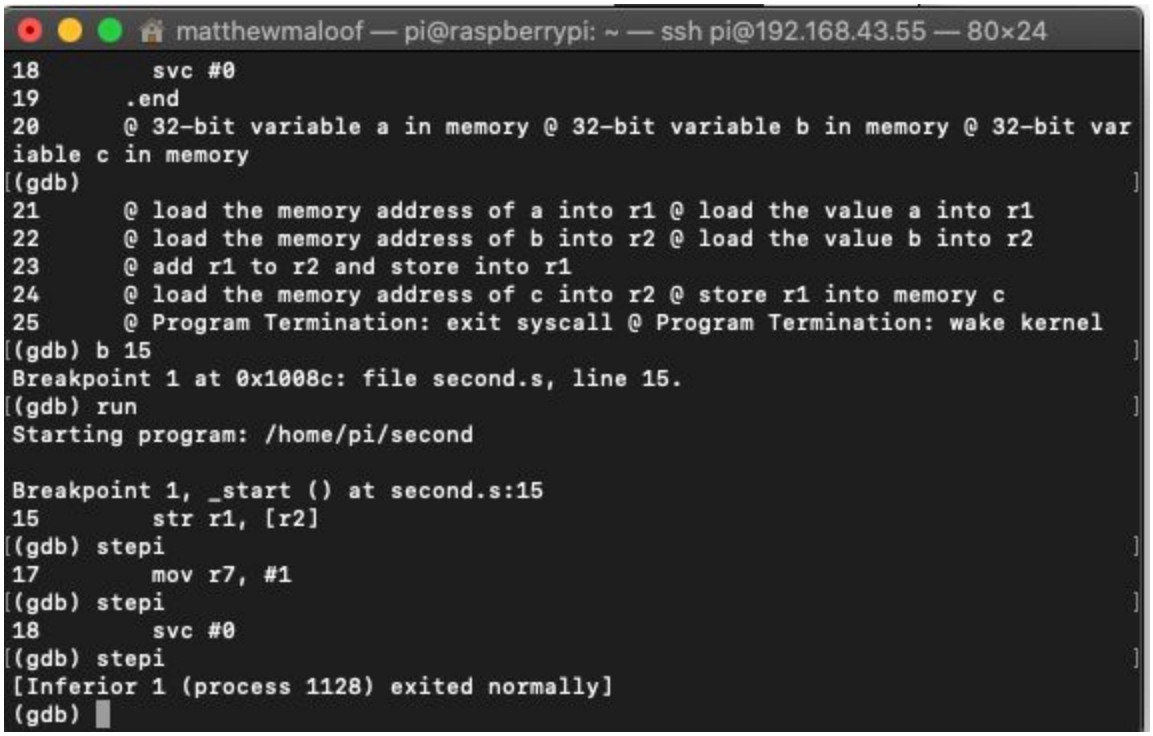

Figure 4: Breakpoint + run debugging

We type stepi to step through the instructions one at a time to see what the program is doing.

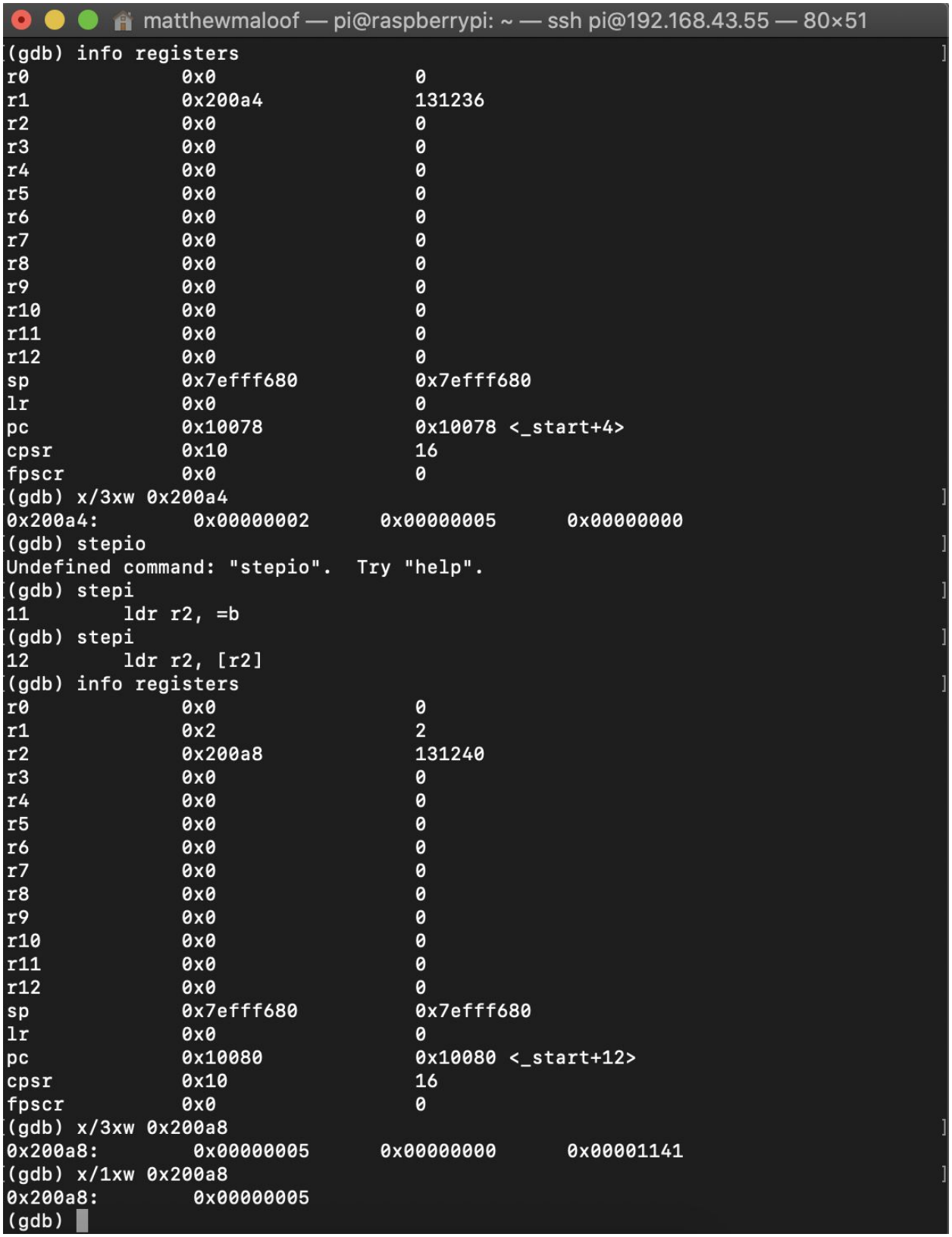

Figure 5: Info registers and stepi and x/1xw 0x200a8 command

In figure 5, we can see the contents of the registers by typing stepi then info registers. Register 2 has the value it should with 0x200a8 and 131240 in the right column. To display in hexadecimal, we can use the command x/1xw to display one hexadecimal followed by the address in the register, which in this case is 0x200a8, so x/1xw 0x200a8. The output of this is shown above which is 0x00000005 which is correct.

Part B:

Using the template from Part A above, we create the code for arithmetic2.s with the instructions given in the slides.

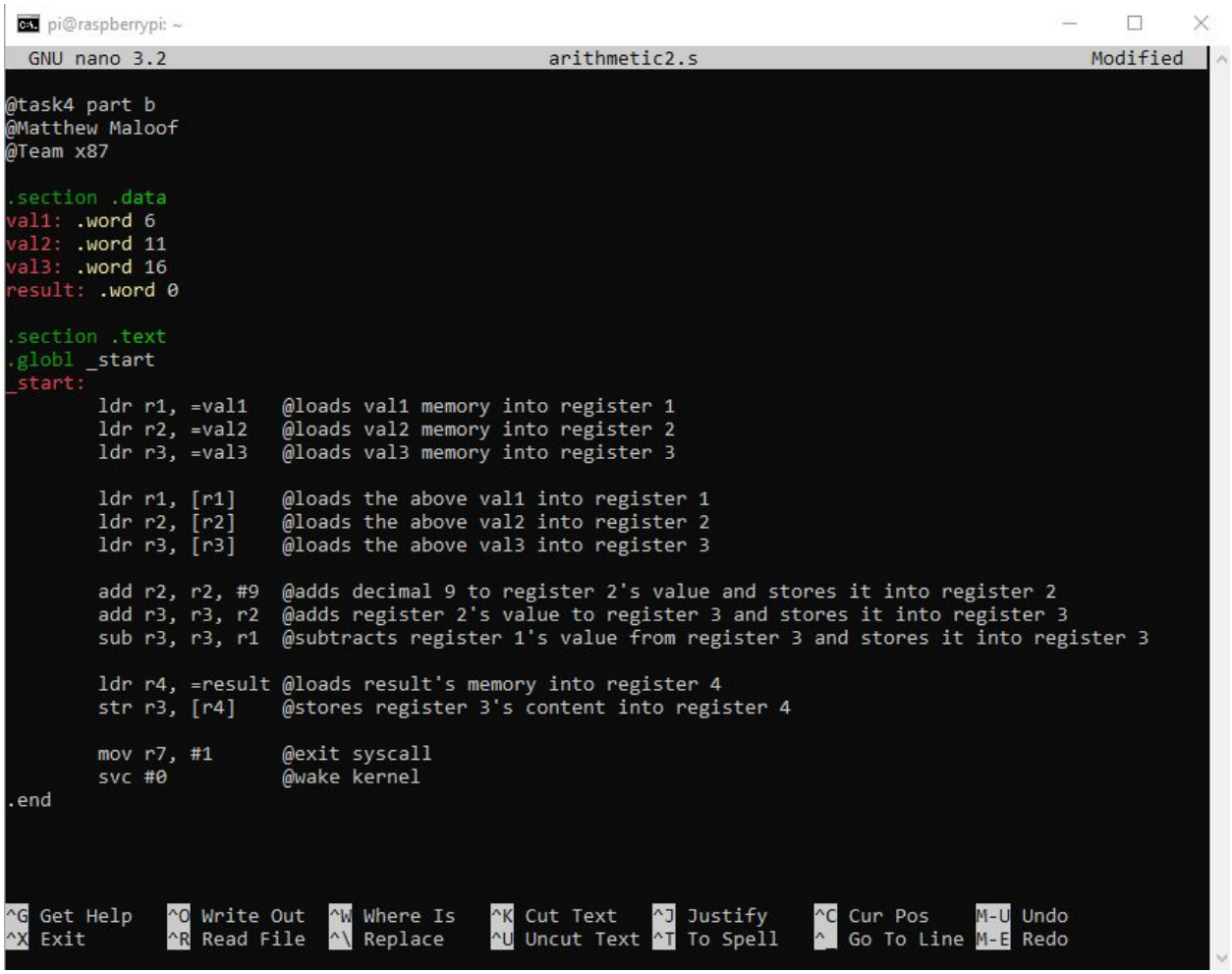

Figure 6: arithmetic2.s code

We can then compile and turn the file into an executable. After running this code, no output is shown just like in Part A. Next, we are going to debug the program so we can see what's being run/stored in the registers.

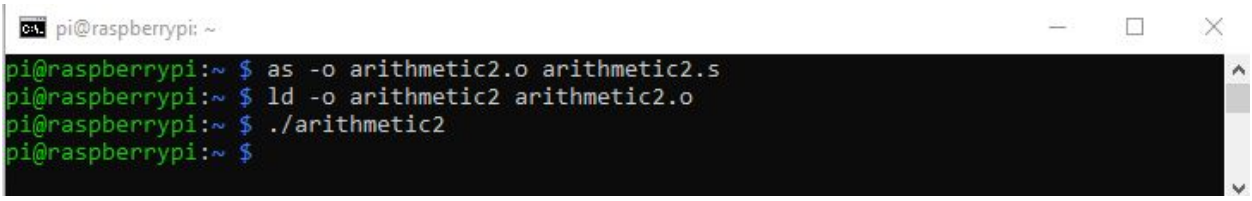

Figure 7: arithmetic2 code not showing any output

Figure 8: debugging continued

We enter gdb GNU debugging mode and set a breakpoint at line 27, then use stepi and info registers to view what's in the registers at each line of code as we progress through the file shown in figure 8 (these are the final values in the registers) Finally, we use the command in Part A to view the memory address in hex. x/1xw 0x200c4 which gives the hex value 0x0000001e.