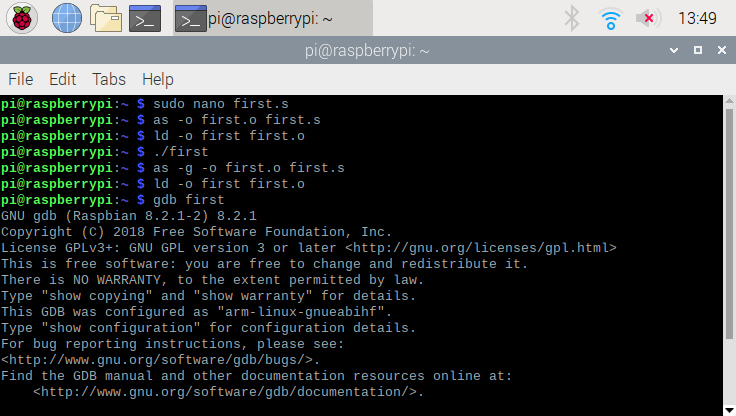
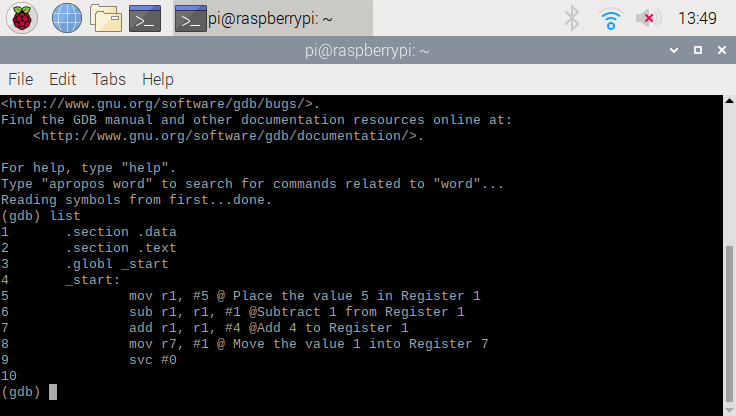
ARM Assembly Program Report

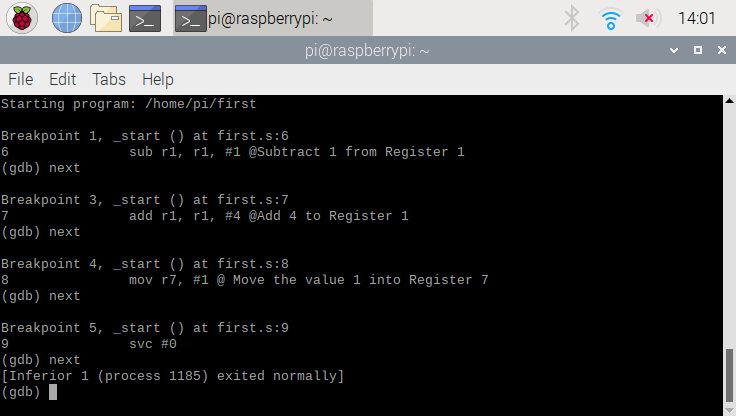
Part 1: First Code



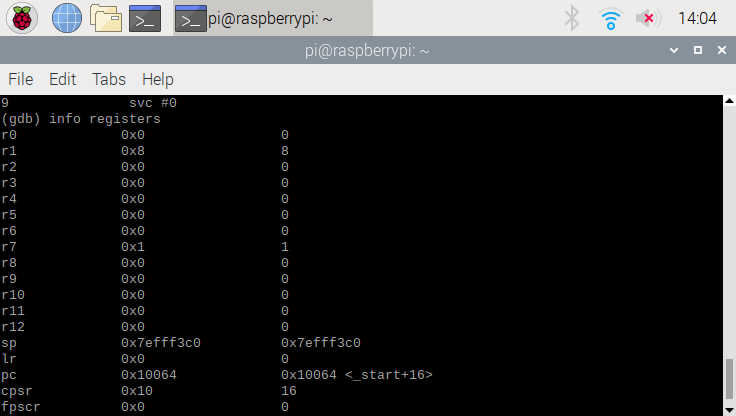
To preface, I used sudo before entering the code into the GNU because I was not able to run the code, but this allowed it to gain access. Also, this is what I wrote before entering the GDB for debugging. I started with “sudo nano first” which allowed me to input the code and allowing me to fun the first program. After I saved the first.s file, I assembled the program and made the first.o program. After that, I linked the first file to first.o. Once this was complete, I ran the code “first”. Once no errors showed up I decided to assemble (using as -g -o first,o first.s) and link the program (ld -o first first.o) once again, but this time to be able to be used in the debugger. Then I ran the code in the debugger using “gdb first”.



This is the code that was inputted in the beginning and this was where I was able to set breakpoints to be able to run each line of code and see the result.

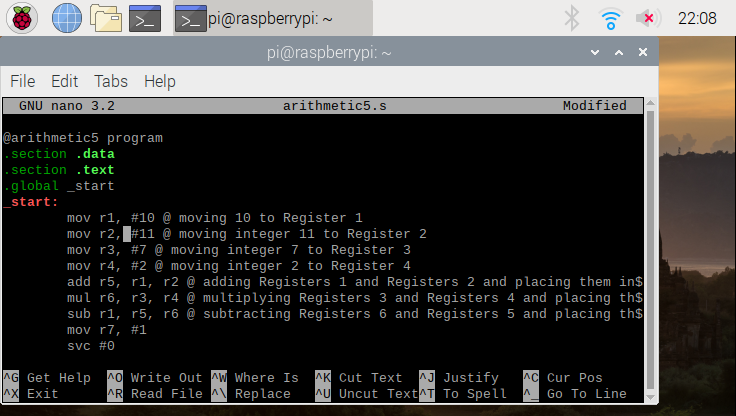


These are the breakpoints I set up and made sure each one ran before looking at the result of the registers.

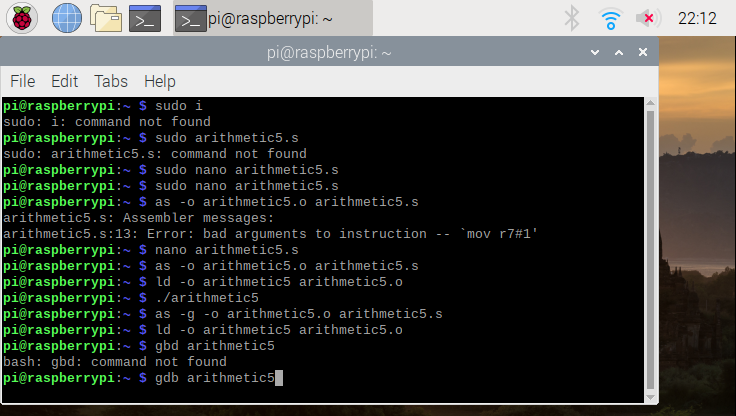


This was the result of the registers and was seen by typing in “info registers” in the debugger. As you can see in the code above, 5 was placed in Register 1, 5 was subtracted by 1 equalling 4 and finally 4 was added into Register 1 resulting in the value of 8 being stored in the register. Finally, the value 1 was put into Register 7 and we can see above that register is shows the same information.

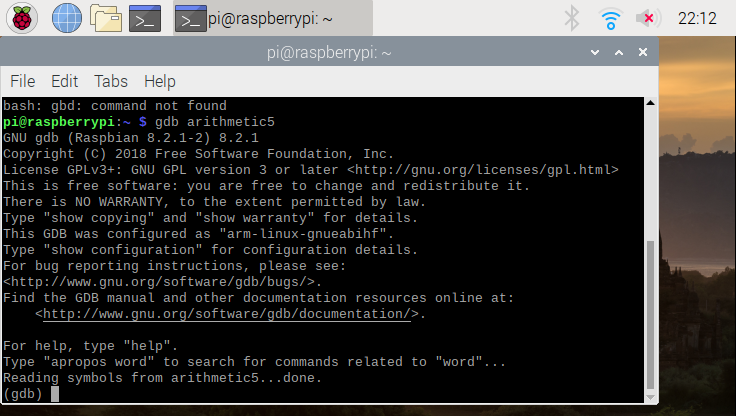
Part 2 : Arithmetic Code



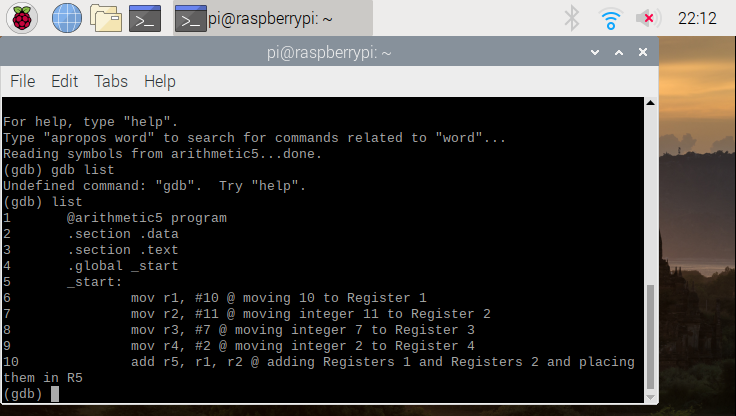
This is the starter code for the file “arithmetic5.s” that was inserted in the GNU nano. As you can see, I also put comments to describe each line of code.



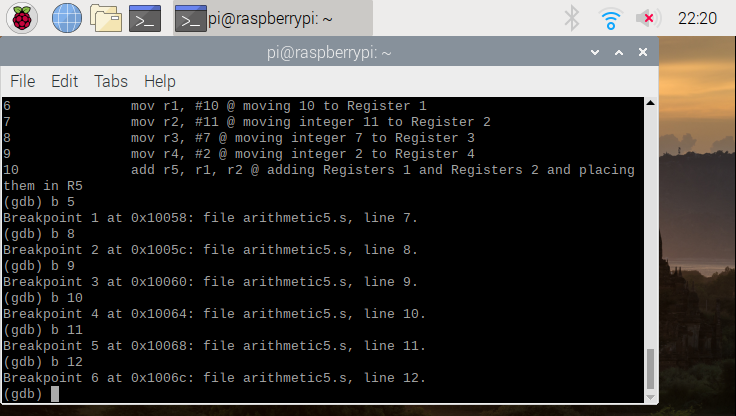
this is what I wrote before entering the GDB for debugging. I started with “sudo nano arithmetic5.s” which allowed me to input the code and allowing me to fun the first program. After I saved the arithmetic5.s file, I assembled the program and made the arithmetic5.o program. After that, I linked the first file to arithmetic5.o. Once this was complete, I ran the code “arithmetic5”. Once no errors showed up I decided to assemble (using as -g -o arithmetic5,o arithmetic5.s) and link the program (ld -o arithmetic5 arithmetic5.o) once again, but this time to be able to be used in the debugger. Then I ran the code in the debugger using “gdb arithmetic5”.



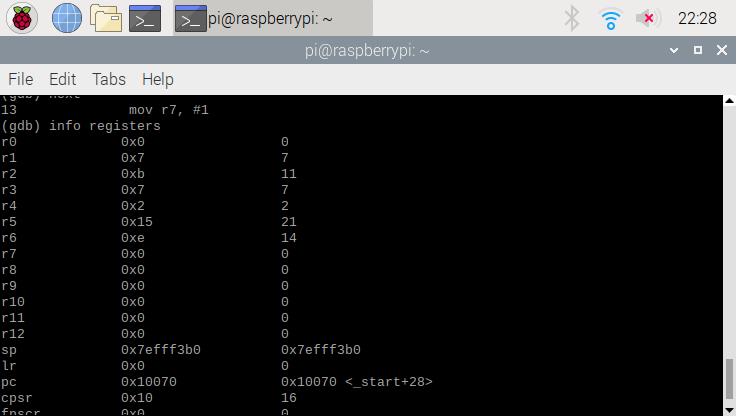
This is the screen when I typed in “gdb arithmetic5” and launched the debugger.



This is where the code is listed so we are able to add breakpoints for each line.



As you can see the breakpoints are being added so we can run each line of code.



This is the result of the code. Each one of the lines of code were executed where the value 10 was placed in Register 1, the value 11 was placed in Register 2, the value 7 was placed in Register 3, the value 2 was placed in Register 4, and adding the contents of Register 1 and Register 2 which is 21 and that value was placed in Register 5. Finally, Registers 2 and Register 4 were multiplied and the value of 14 was stored in Register 6.

Overview of Assignment and What I Learned:

Connecting my Raspberry Pi was difficult and since I have very little experience using the device and never using GitHub in general. With the help of my team, I was able to finally connect my Raspberry Pi to GitHub and access the repository. Lastly, when I was creating the code for the arithmetic part of the program, I ran into an issue being able to run the program, but with the help on my team members, who were able to assist me on what I needed to fix in my code in the nano.

Each of the pictures have a description of what processes occurred between each code. I was able to put values into assigned registers, perform operations between two register, and move the final value in the assigned register. To be able to do this, there is a somewhat of a long and lengthy process of inputting the code, assembling the file, linking the file, and finally running the file. After this to start debugging, you how to assemble the file again, link the file again, and load it into the debugger which was done by putting it in the GDB. After this, you are able to assign breakpoints and run through each line of code (or at least where the breakpoints were placed) and be able to see the out by typing “info registers”. After we made sure each of the values were in the proper register, we were able to leave the debugger.

I learned many things while completing the assignment. For instance, I was able to navigate through the Raspberry Pi, link GitHub through the terminal, execute different programs through the terminal, place files that were stored on the Raspberry Pi and upload them on GitHub. Lastly, I learned a different form of coding where you can add the values of two registers and place them in a different register all in one line of code.