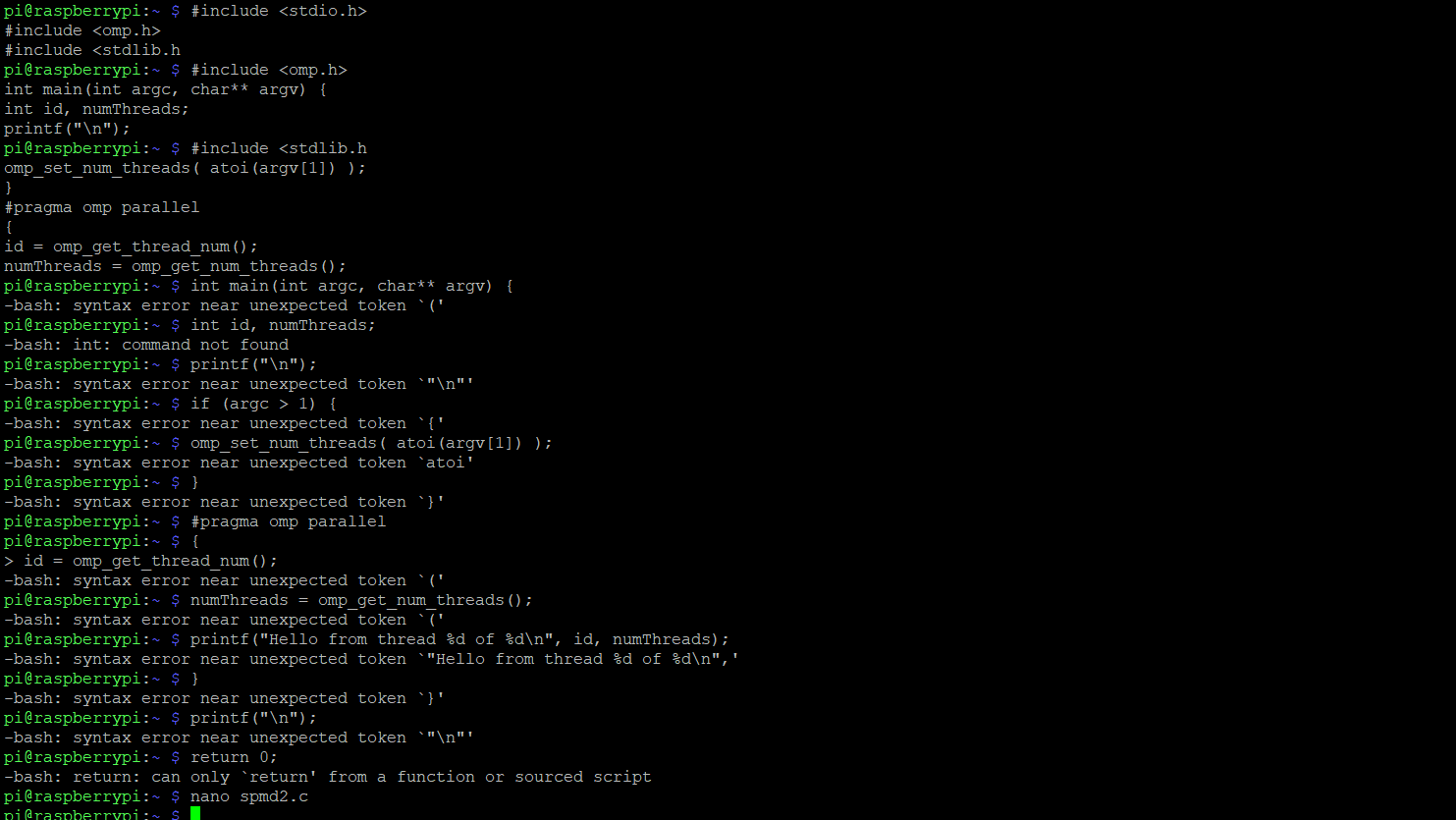
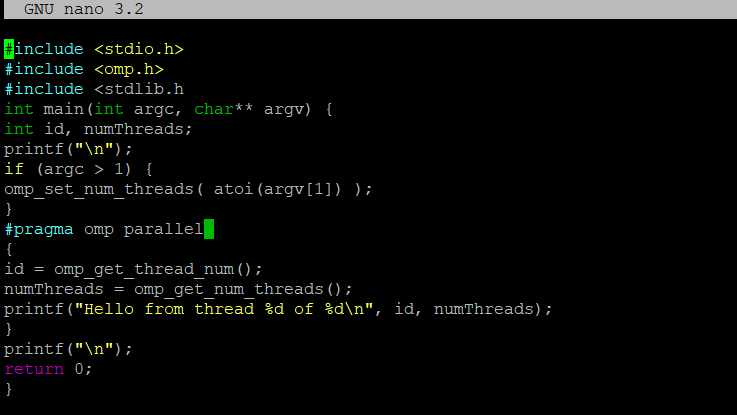
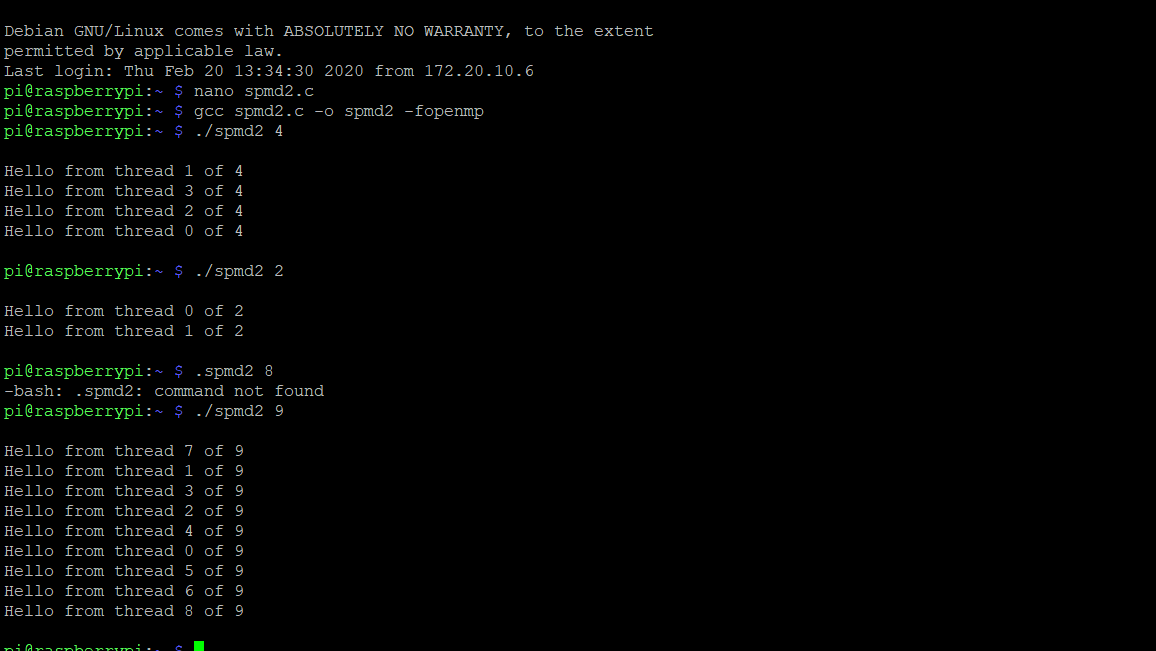
Getting Started with Raspberry Pi B+



This is when I copied the code into the terminal. I proceeded to get a few errors and copied the code into the GNU.

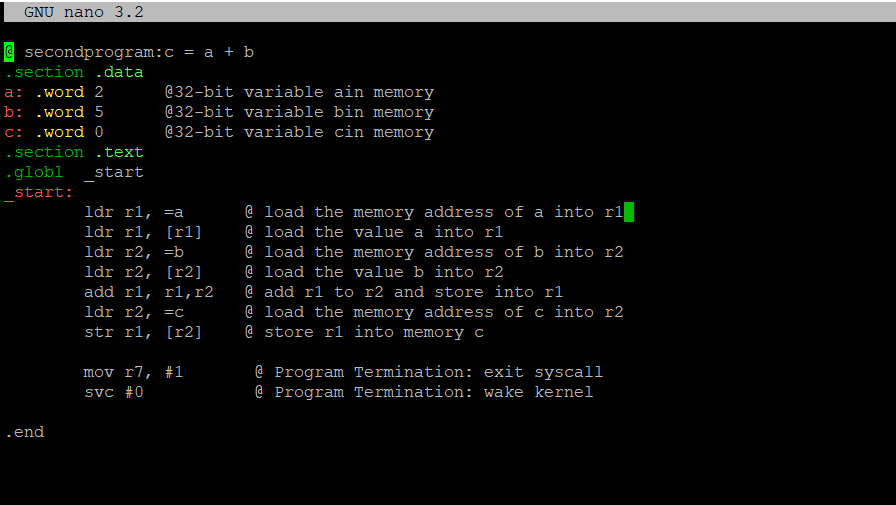


The code was put into the GNU. It was not able to run because of two things: 1. Line 3 needs a > after the “h” in the code. 2. Line 12 and 13 has id = class and numThreads = class respectively, but haven’t been initialized so putting int in front of id and numThreads will allow the code to finally run (int id = cmp\_get\_thread\_num(); and int numThreads = cmp\_get\_num\_threads();)

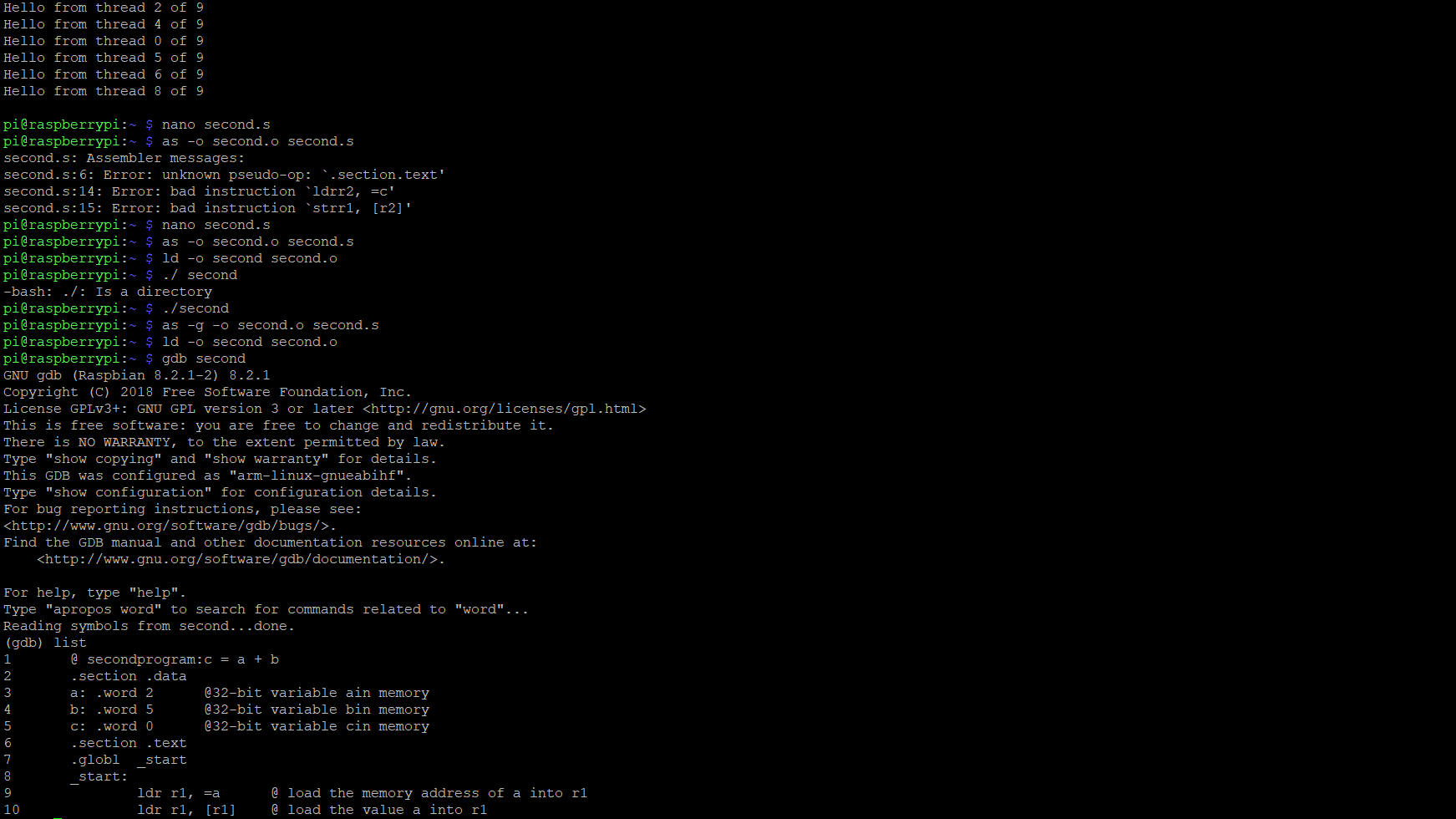


After the issues were fixed, I was able to run the code. After I copied to code into the nano as you can see above, I used the code “nano spmd2.c”. To be able to run the file, I used the code gcc spmd2.c -o spmd2 -fopenmp and after that, I ran the code using “./spmd2 and added 4 after that to execute 4 threads. After I ran 4 threads, I wanted to see how many tries it would take to get a sequence of threads in order with two threads and I got it in one try. I also tried to do 9 threads to see how the threads would be ordered and as you can see it was random and threads start from 0 and are out of 9 but go to 8.

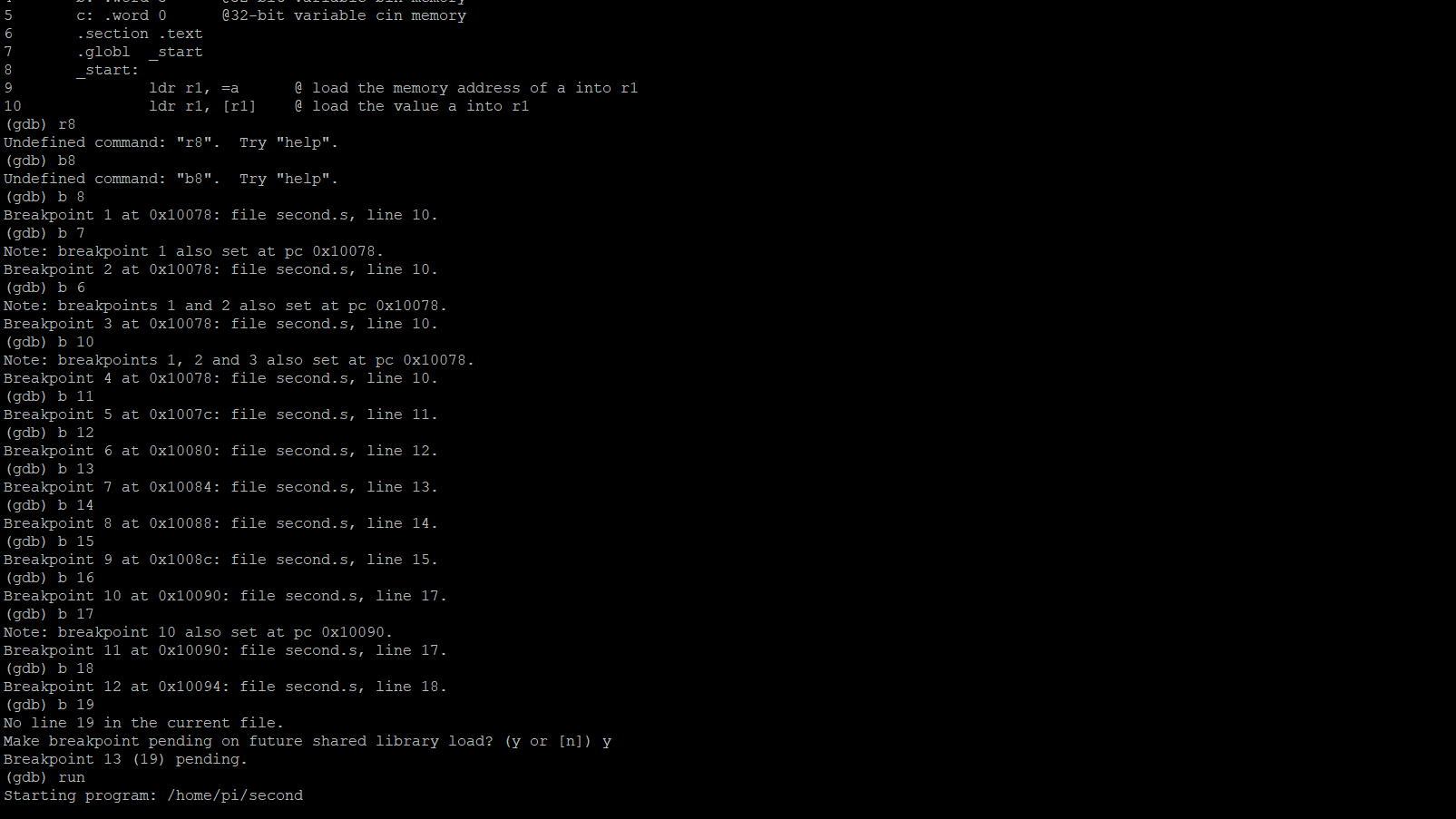
**Second Program**



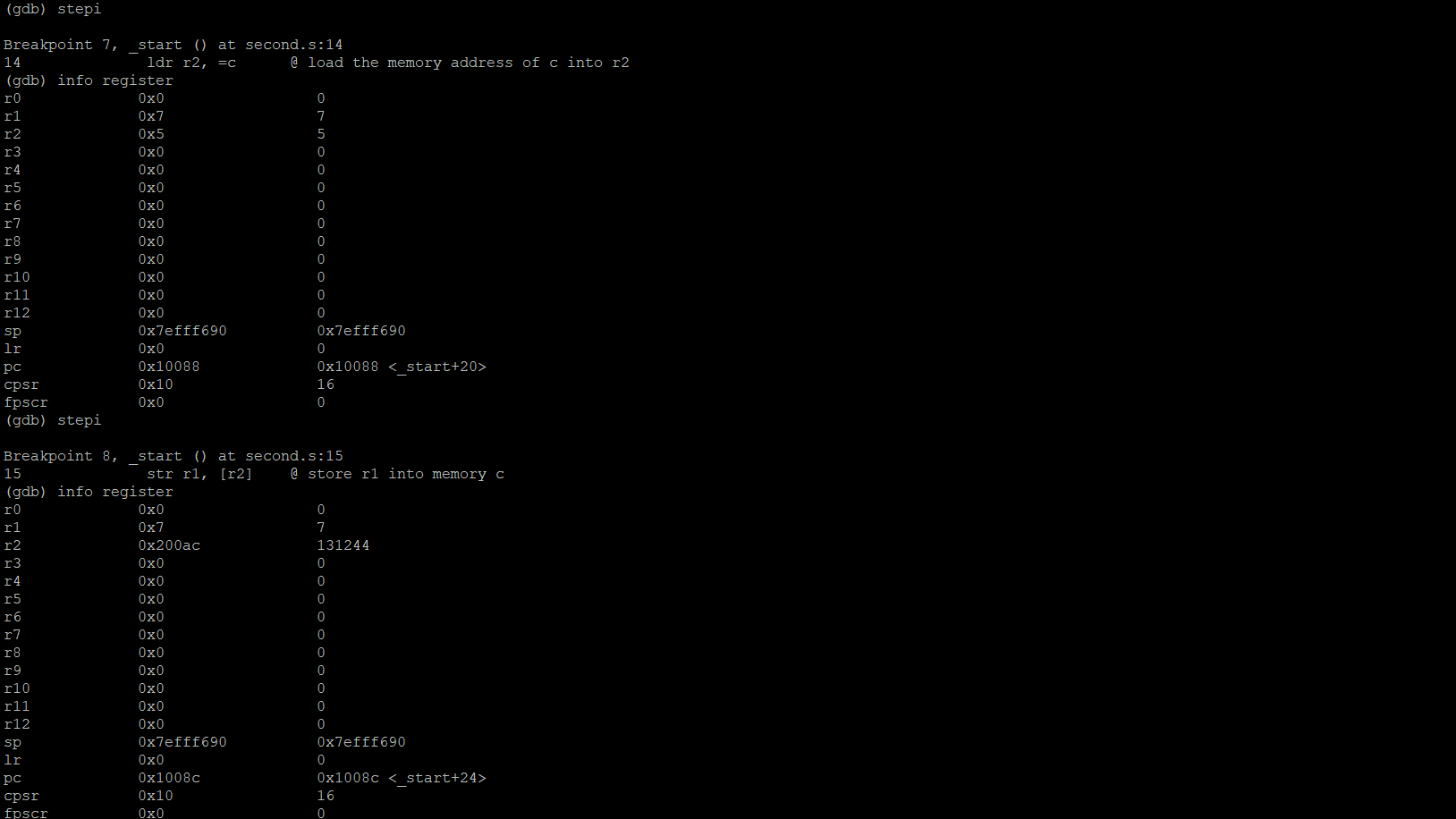
Here’s my program I called second.s and tried to add two values placed in a and b and equate it to c and store it into that memory.

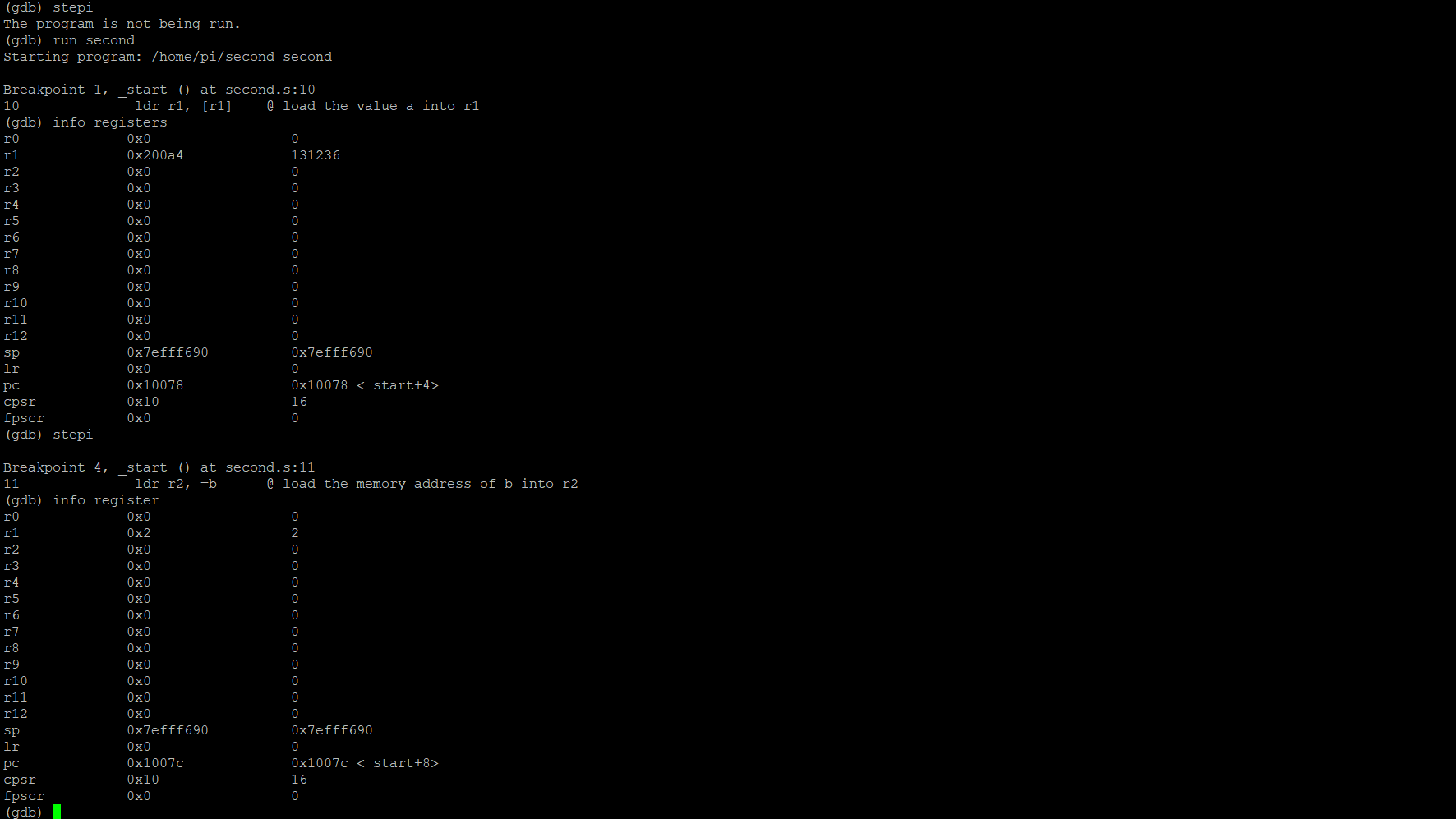


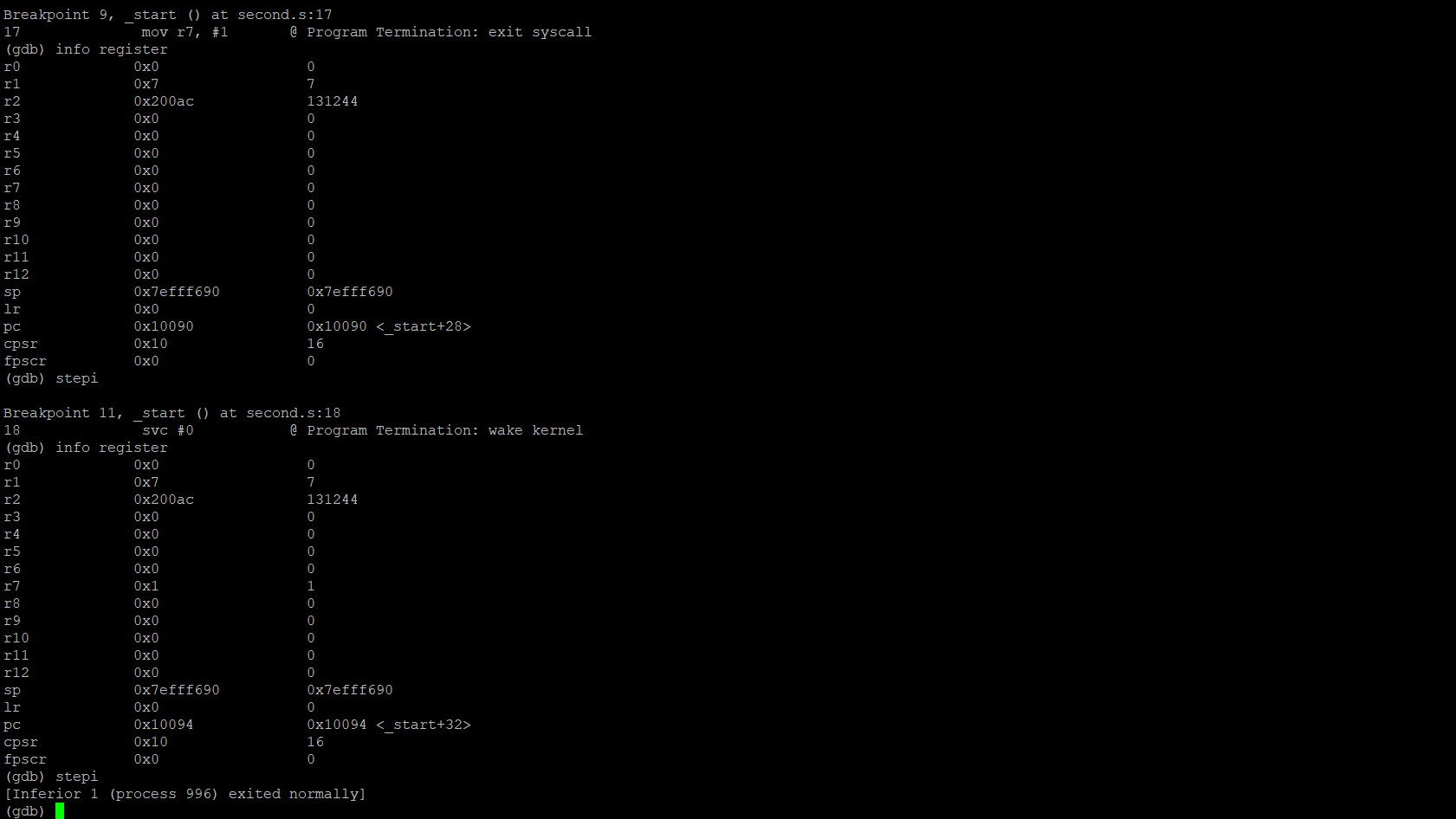
Within these two screenshots, I create second.s file using “nano second.s”. After that I assembled the file by using “as -o second.o second.s” and used “ld -o second second.o” to link the file. After this I used “./second” to run the code. Since no errors came up, I decided to run the program through the debugger to see the final result

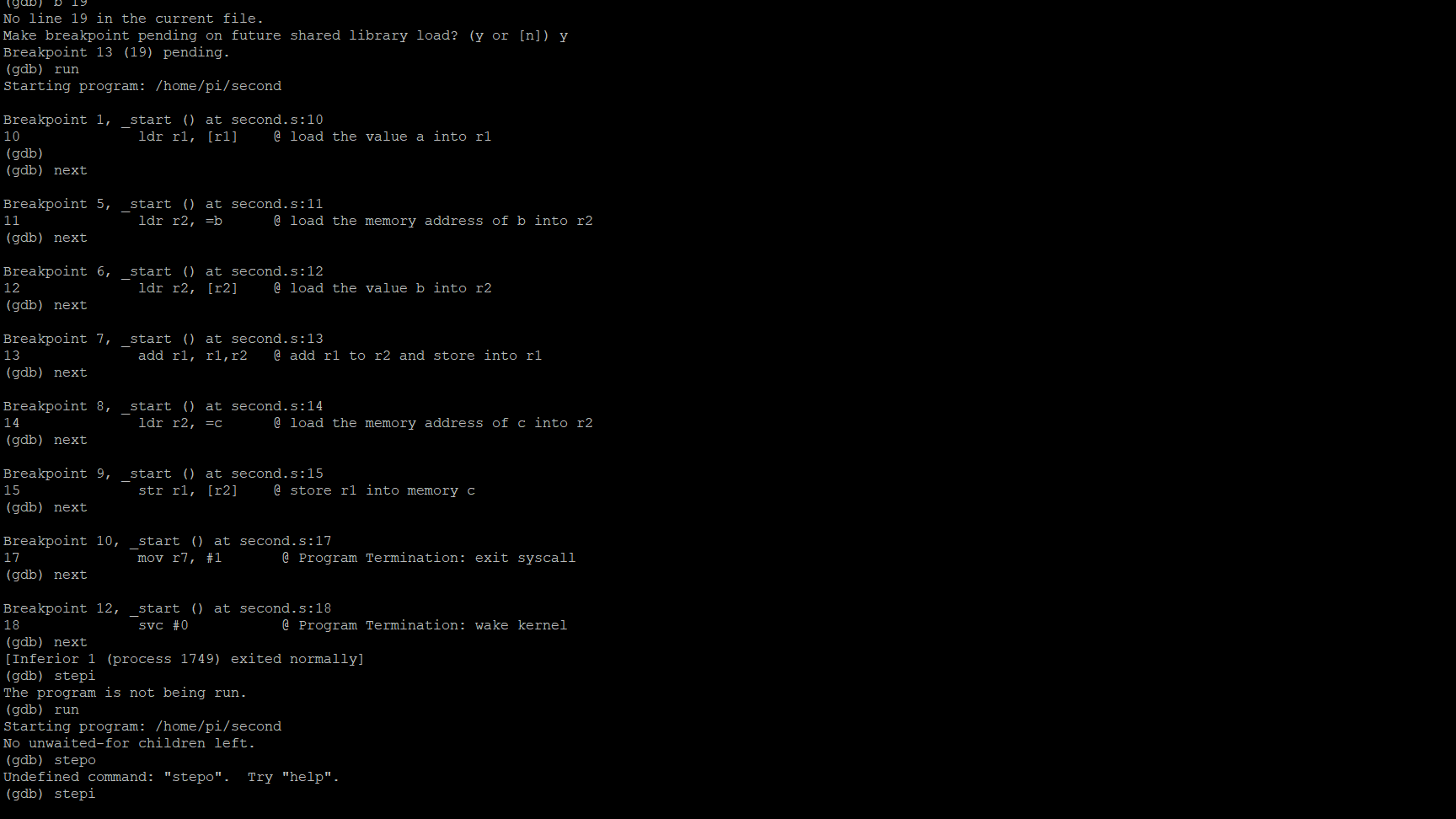


After loading into the debugger, I set the breakpoints from line 10 to line 18. To make sure all lines of code were executed, I put a breakpoint at “b 19” or line 19 to make sure I was at the end of the line.

 The debugger ran the program and stopped at the first breakpoint. From the first breakpoint at line 10 we can see the value the memory address of a (0x200a4) was loaded into Register 1.

 I checked the address to make sure the right value was put into the right address to make sure there was no fault in my code and as you can see the value of 2 in Register 1 stands true. The next breakpoint the memory address of b (0x200a8) was loaded into Register 2.

 From the next breakpoint we can see the value from Register 1 (Previously the sum of Register 1 and Register 2), which is 7 was stored in memory c. As we can see above the memory address is seen above in Register 2 (0x200ac).



Initially I ran the code and did not check the info register, I just checked if the code ran. But as you can see, I corrected it above.