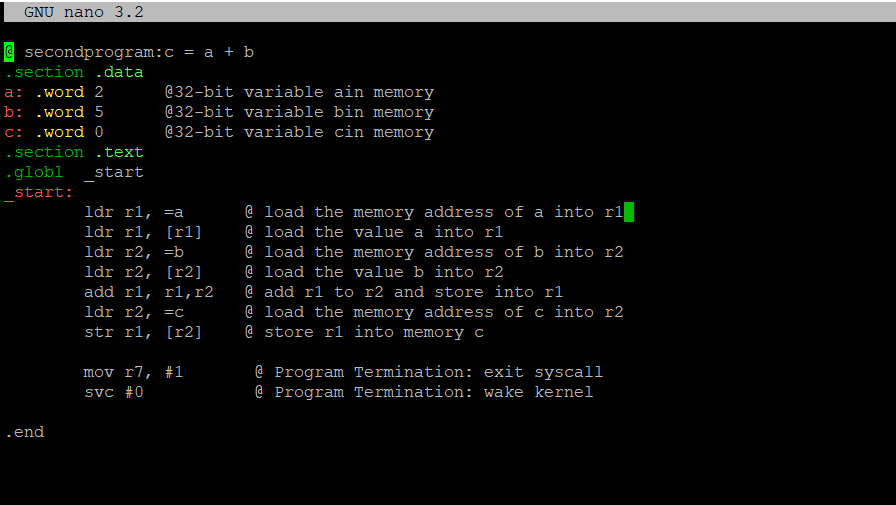
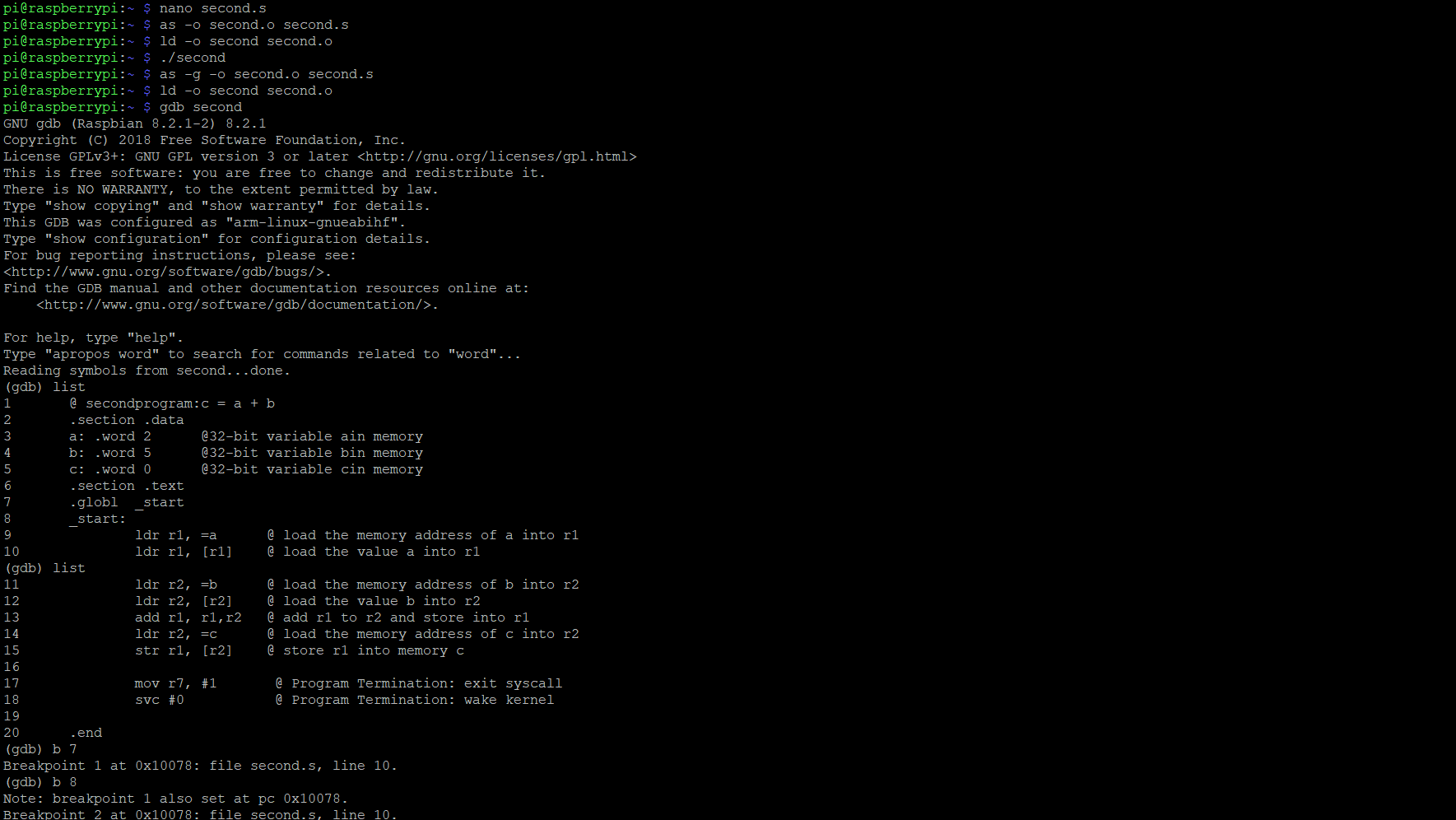
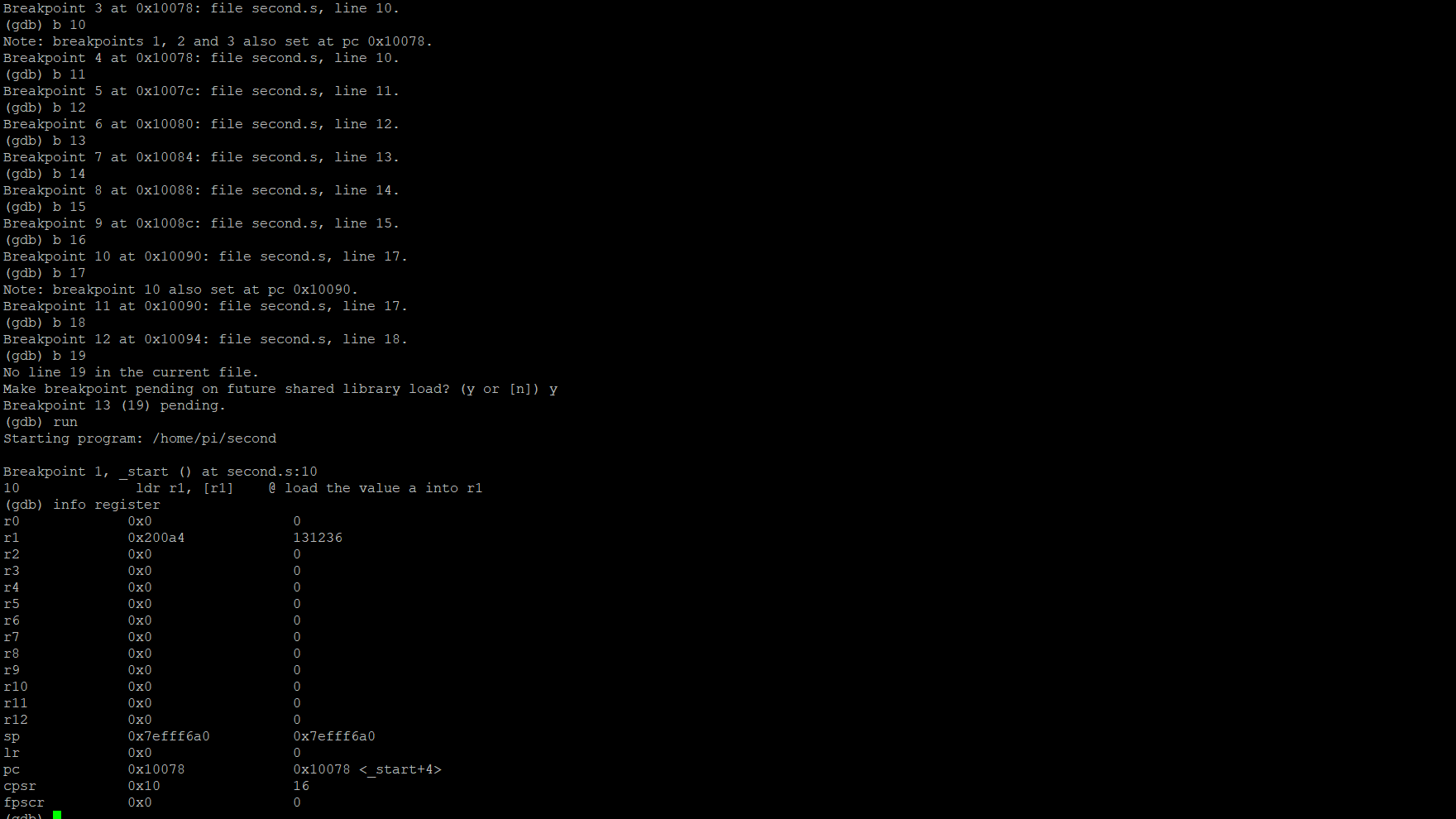
**ARM Assembly Programing**

**Second Program**

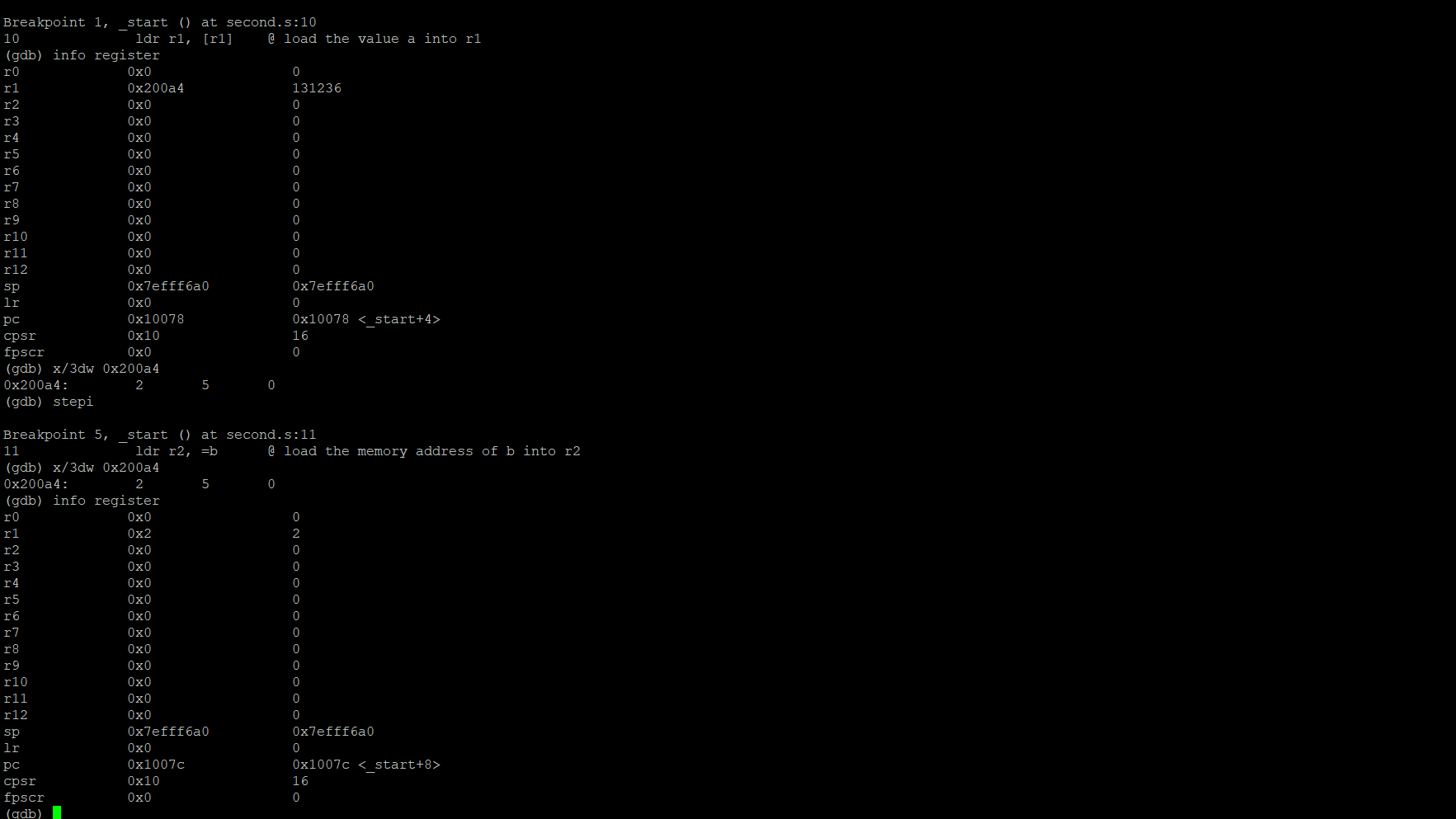




Within these two screenshots, I create second.s file using “nano second.s”. After that I assembled the file by using “as -o second.o second.s” and used “ld -o second second.o” to link the file. After this I used “./second” to run the code. Since no errors came up, I decided to run the program through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



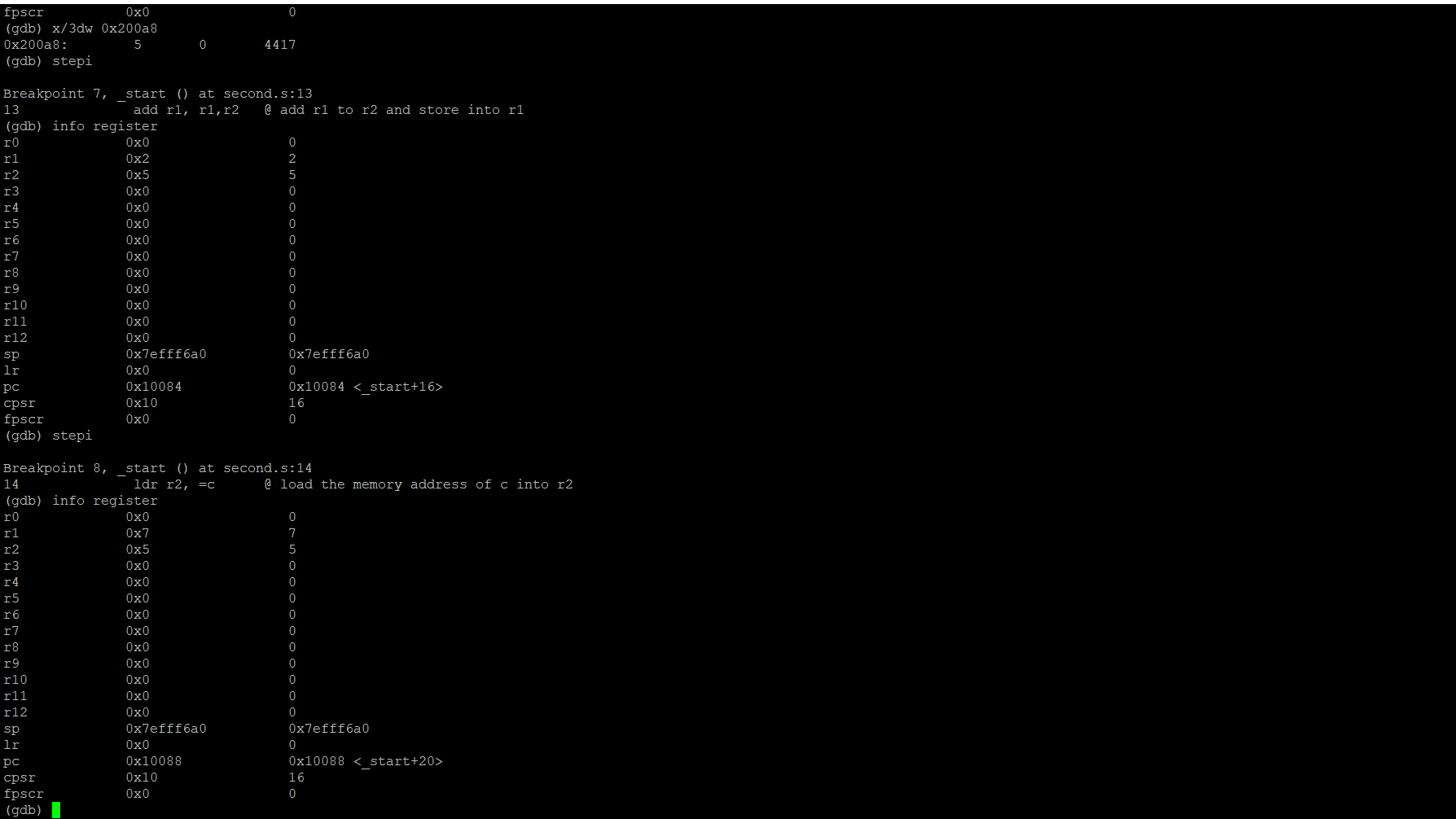
After loading into the debugger, I set the breakpoints from line 10 to line 18. To make sure all lines of code were executed, I put a breakpoint at “b 19” or line 19 to make sure I was at the end of the line.

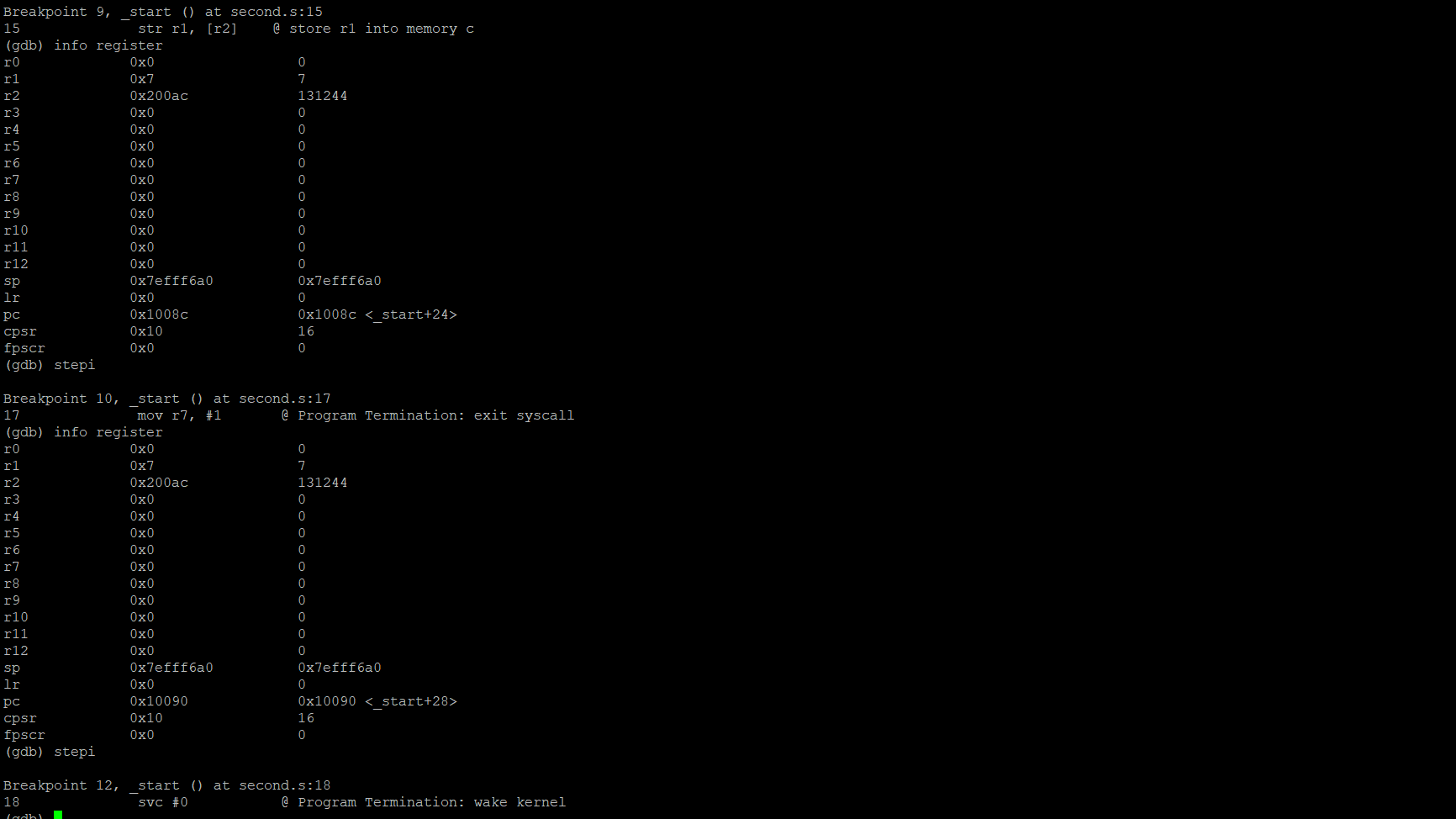


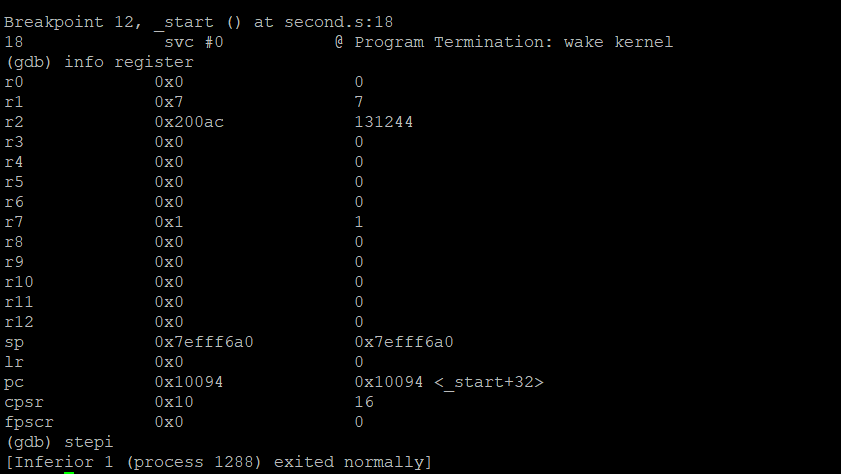
The debugger ran the program and stopped at the first breakpoint. From the first breakpoint at line 10 we can see the value the memory address of a (0x200a4) was loaded into Register 1.



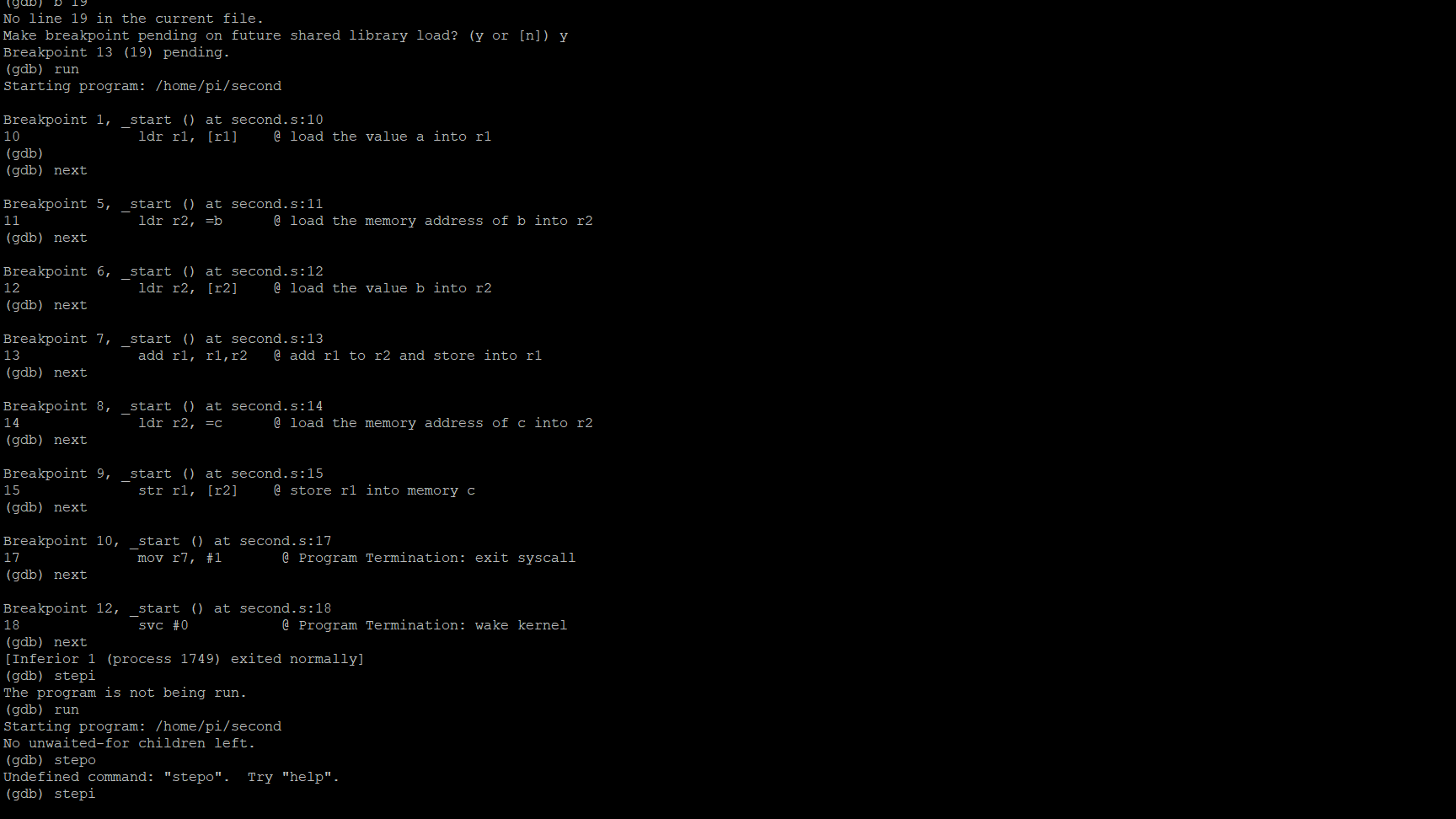
I checked the address to make sure the right value was put into the right address to make sure there was no fault in my code and as you can see the value of 2 in Register 1 stands true. The next breakpoint the memory address of b (0x200a8) was loaded into Register 2.

We can see the value of 5 was place in Register 2. In Breakpoint 8 on Line 14 we can see the value of the Register 1 and the value of Register 2 were added together and placed back into Register 1.



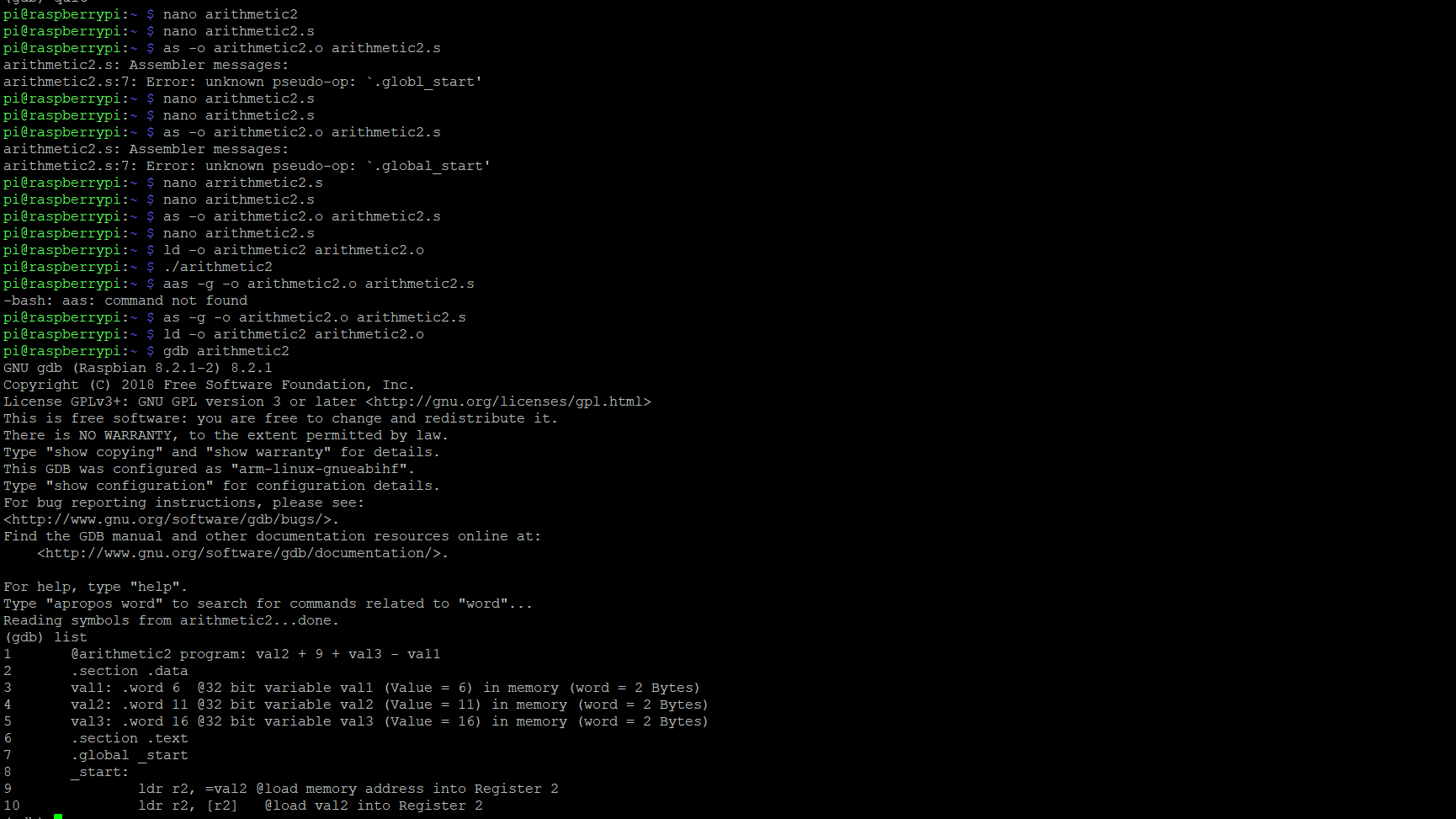
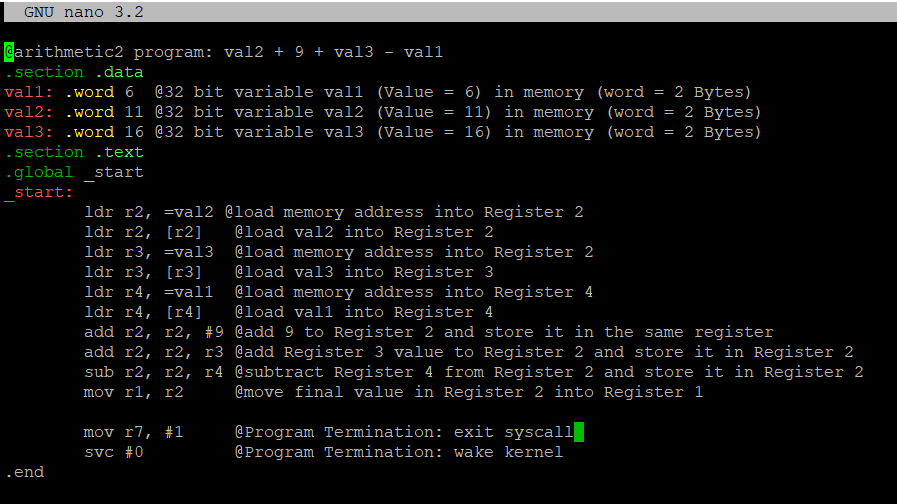
From the next breakpoint we can see the value from Register 1 (Previously the sum of Register 1 and Register 2), which is 7 was stored in memory c. As we can see above the memory address is seen above in Register 2 (0x200ac). 

Lastly, we can see the value of one (coded as #1) was placed in Register 7.

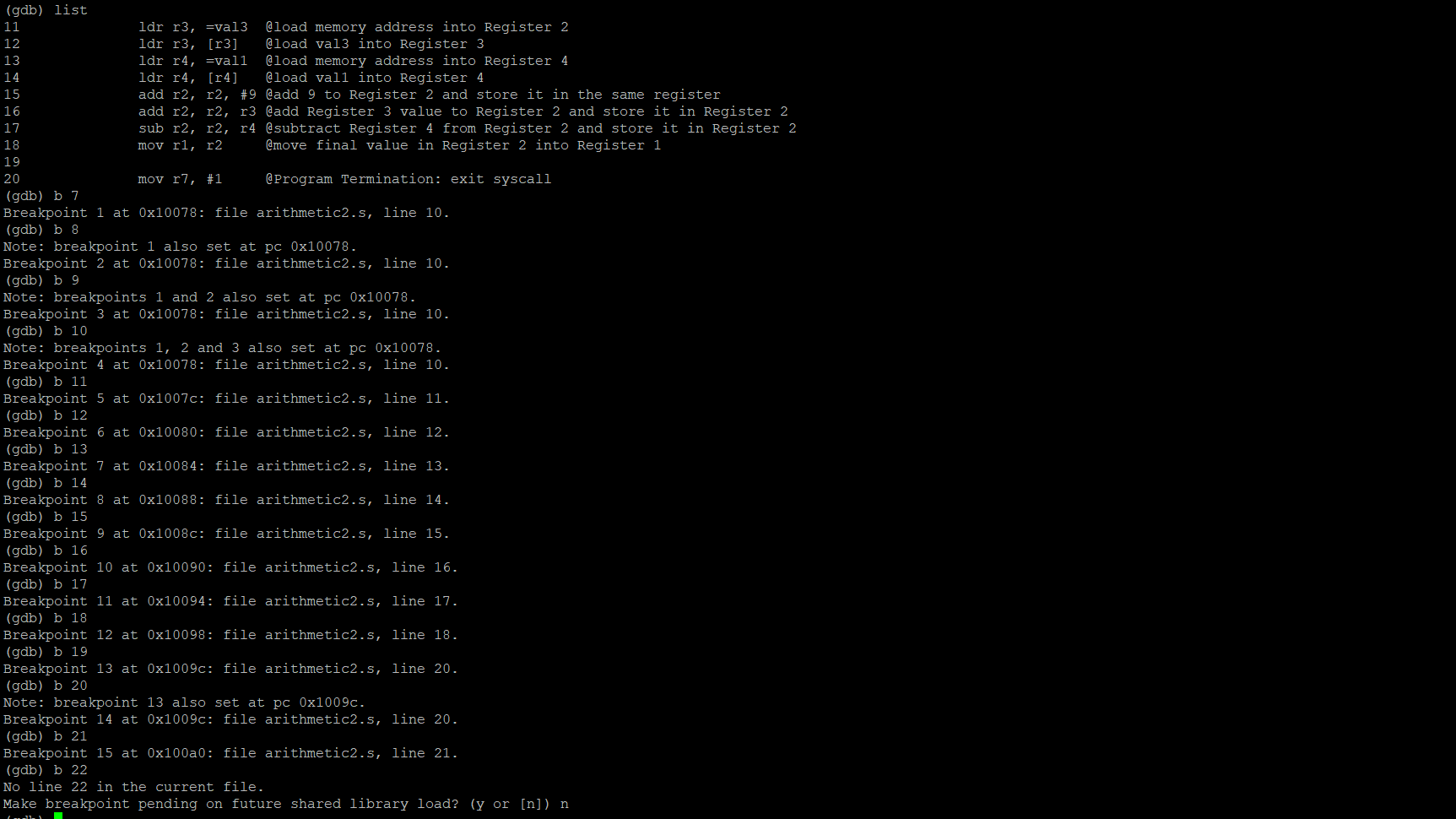


Initially I ran the code and did not check the info register, I just checked if the code ran. But as you can see, I corrected it above.

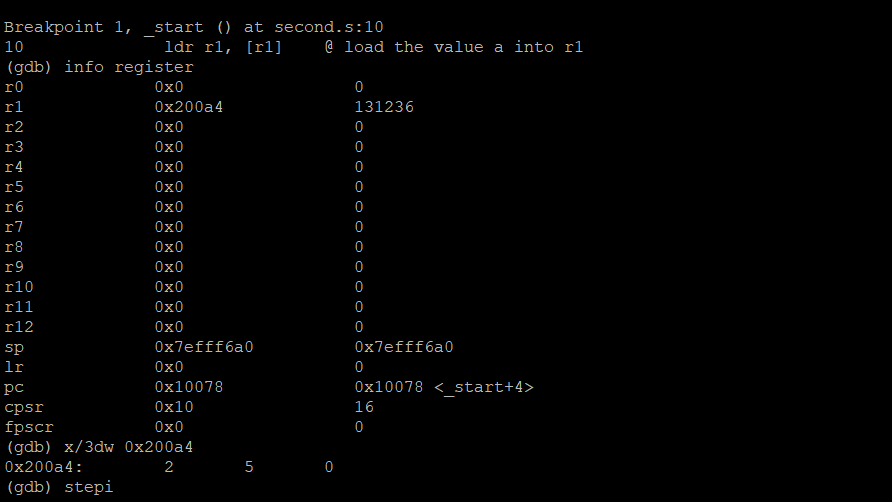
**Arithmetic2 Program**



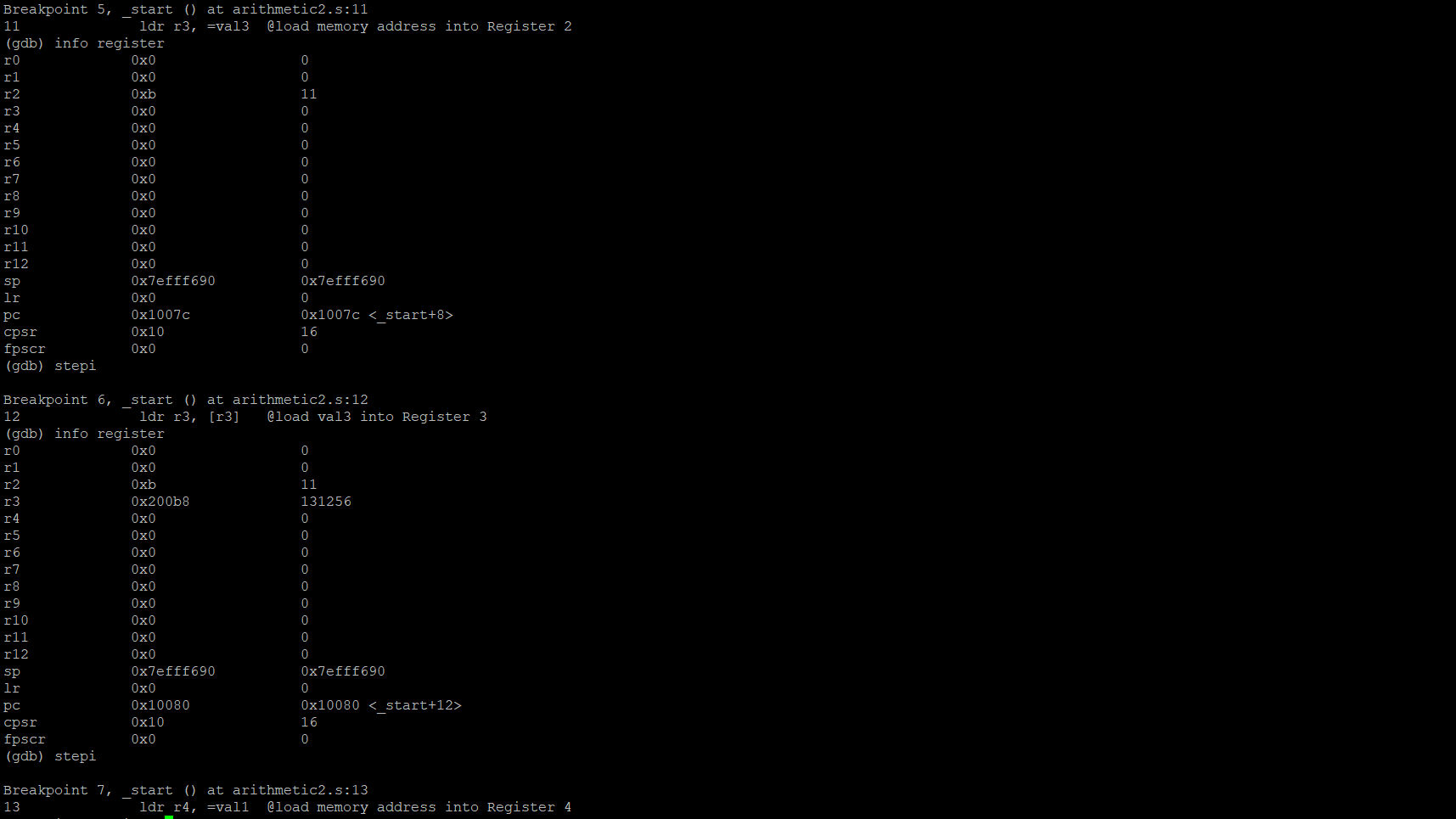
I created the file arithmetic2.s file using nano. After that I assembled the file using “as -o arithmetic2.o arithmetic2.s”. I initially got an error with the code because I did not put a space between .globl [and] \_start, but when I fixed that the file was assembled. After that, I linked arithmetic2 file by typing in “ld -o arithmetic2 arithmetic2.o”. Since no errors came up, I ran the program. The program did not run as you can see above so I prepared the program to go through the debugger using “as -g -o arithmetic2.o arithmetic2.s and ld -o arithmetic2 arithmetic2.o to assemble and link the program to run in the debugger respectively and ran the program through the debugger.



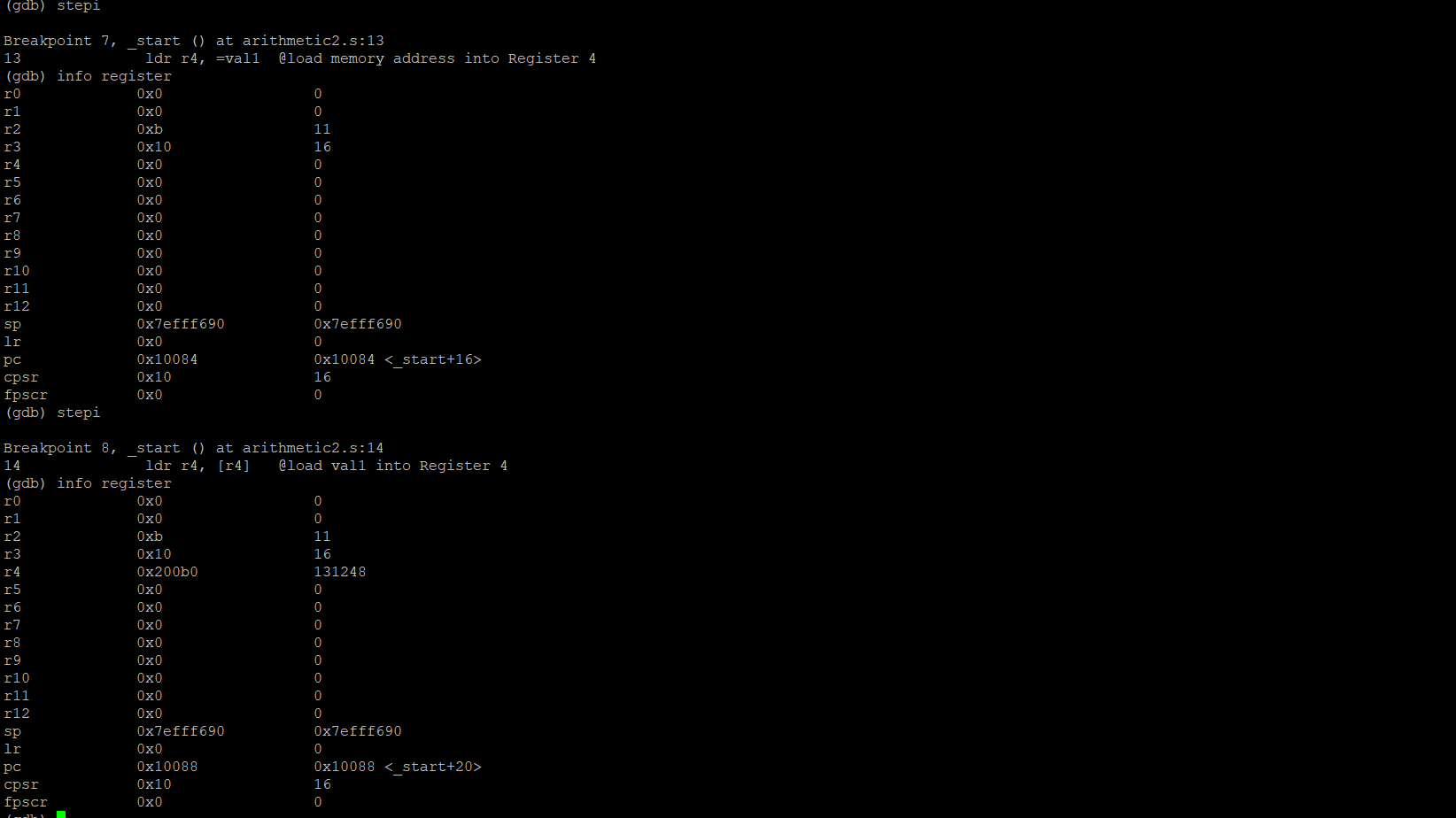
I placed breakpoints from B 10 through B 21. As you can see, I started my breakpoints before 10 and after 21 to make sure I did not miss any code through the program.



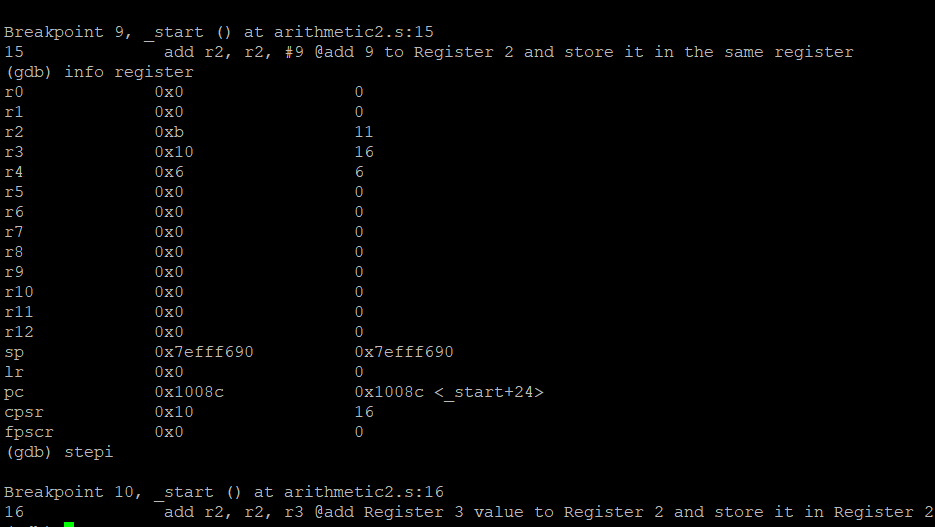
After running the program, we can see the address (0x200a4) of “a” is loaded into Register 1.



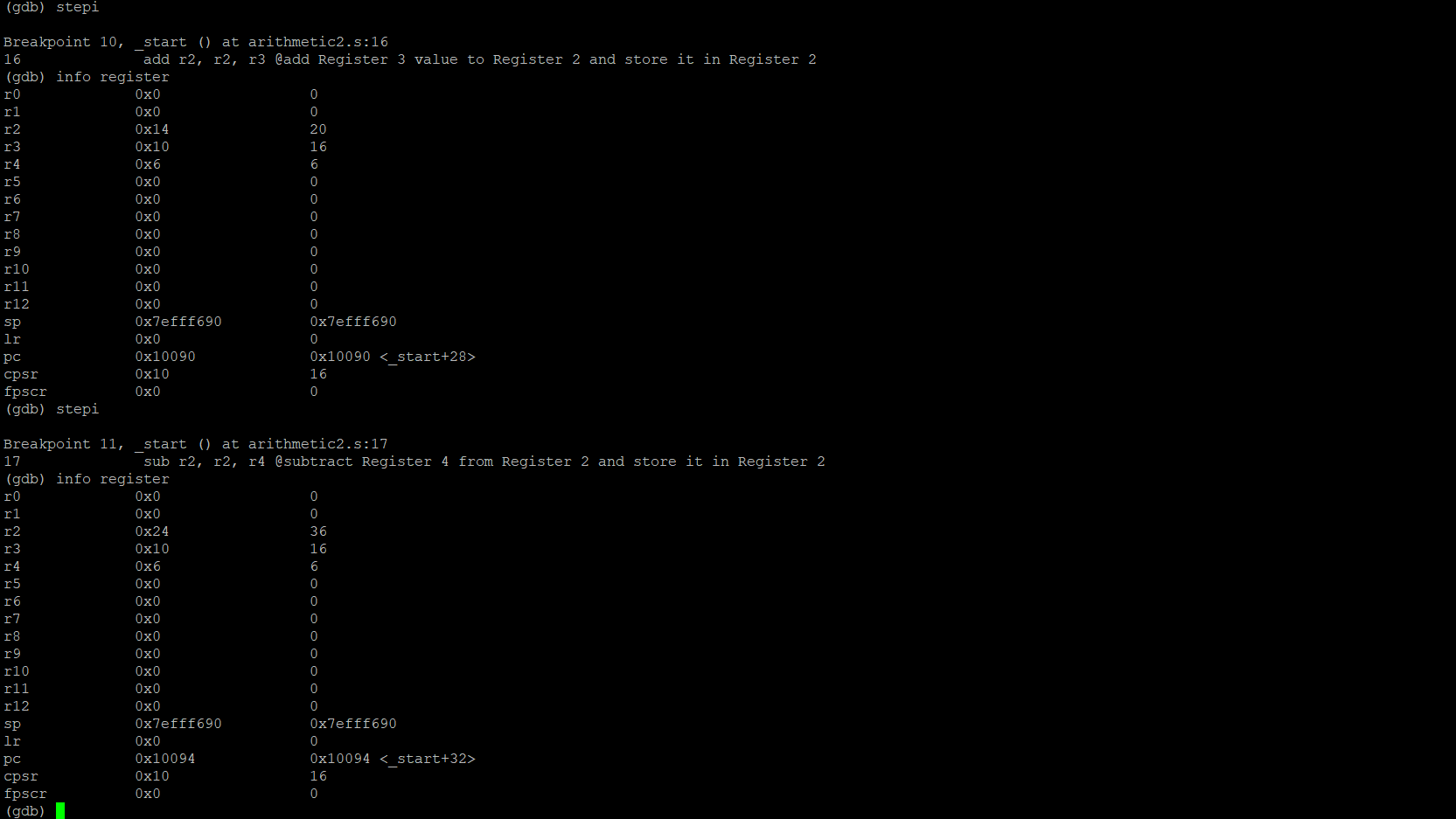
After we loaded the address of “a” into Register 1, we place the value of 11 into Register 1

After that, we load the memory address of val3 into Register 3

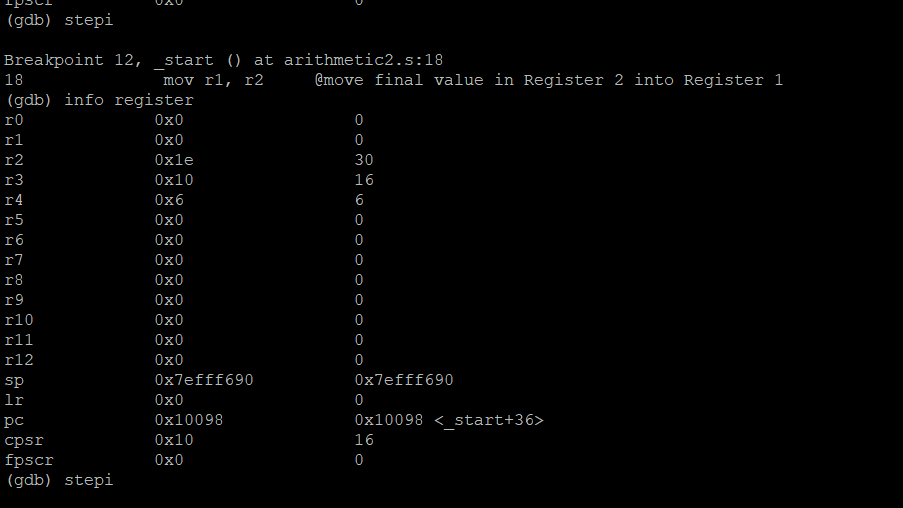
Once we run the step to the next breakpoint, we get the value of 16 placed in Register 3. During that breakpoint, we see the memory address of val1 is loaded into Register 4.



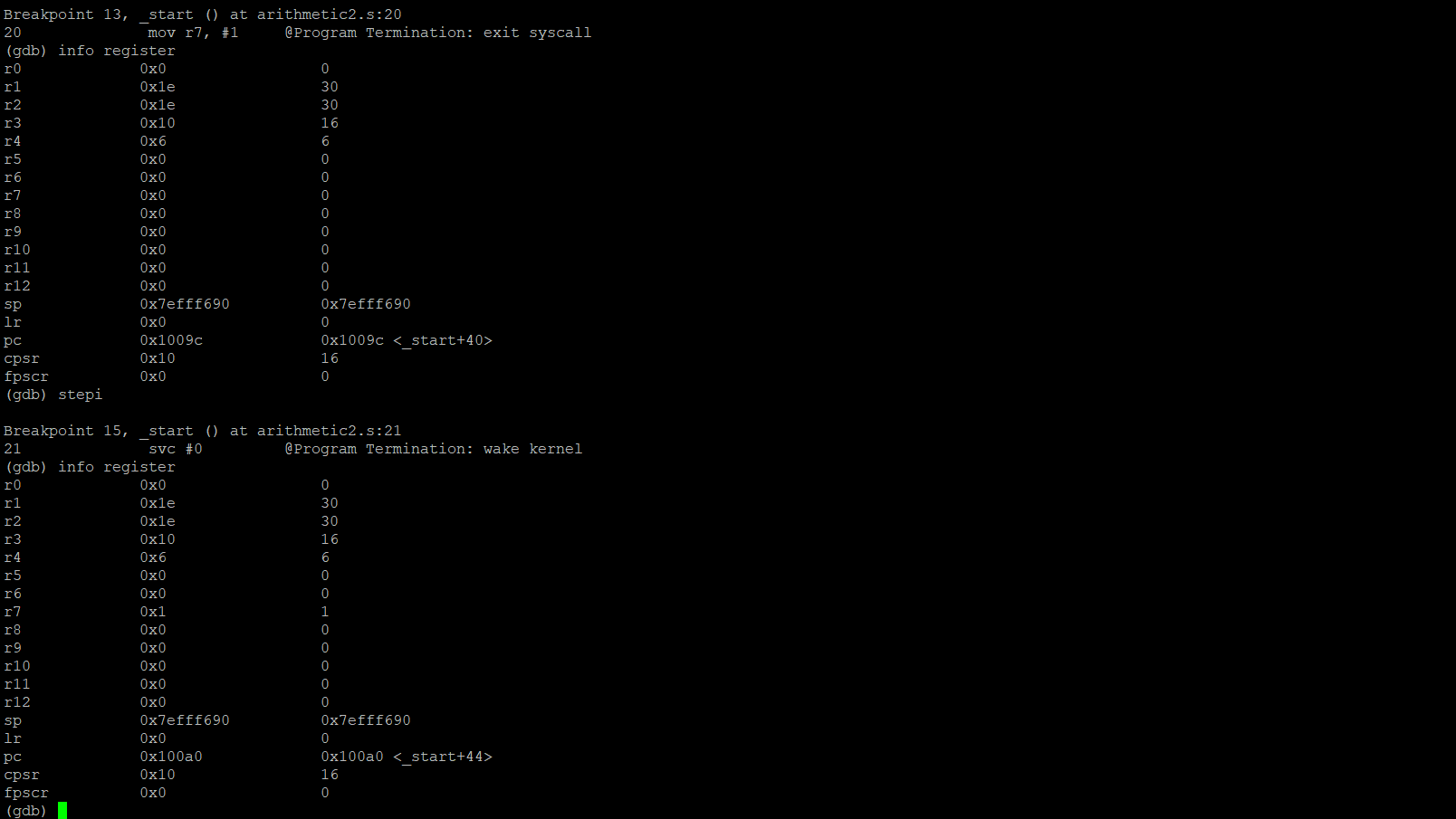
We can see the content of val1 is placed in Register 4 which is 6 as we can see above.



Next we add the value Register 3 which is 16 and the value of Register 2 which is 20 and place it back to Register 2, overriding the previous value of 20.



Now we move the final value which is stored in Register 2 and move (copy) it into Register 1.



We can see the result of 30 placed in Register 1. The last code to execute is to move number (#) 1 into Register 7. I tried to use “stepi” to see if we can execute an additional breakpoint but the program ended so I exited the debugger by typing in (gdb) “quit”.