Victoria Routon

Task 4: ARM Assembly Programming

I begin by creating second file using nano command (Figure 1).

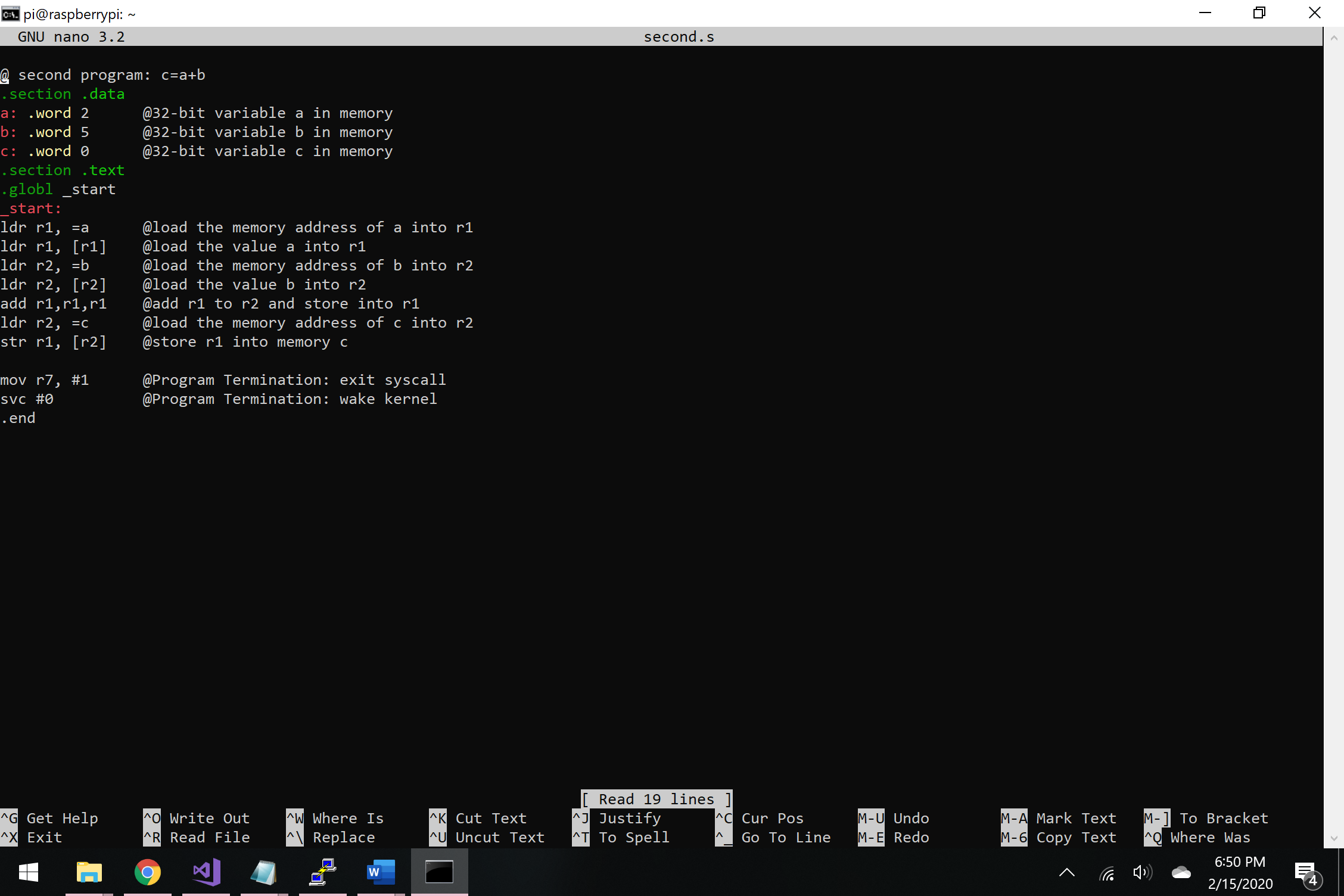


Figure 1 – second file

Then I compiled and ran the program. I noticed there was no output, which is to be expected because the program didn’t print anything. To see the change in registers, I began debugging the code (Figure 2).

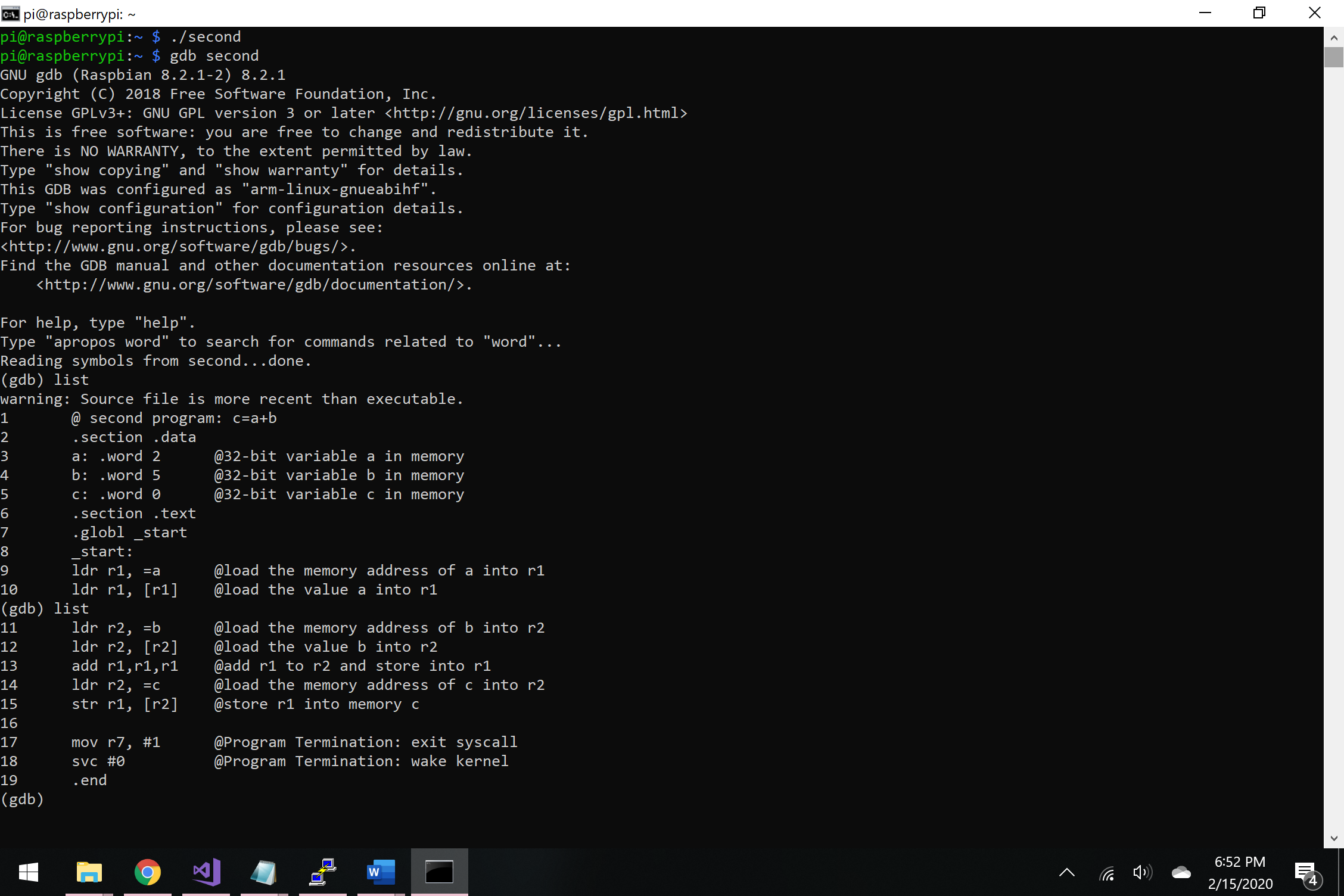


Figure 2 - Debugging the code

I set the breakpoint at line 10 and used the command “stepi” to see what the program does line by line. Then, I typed “info registers” command, which showed me all the registers and their addresses (Figure 3).

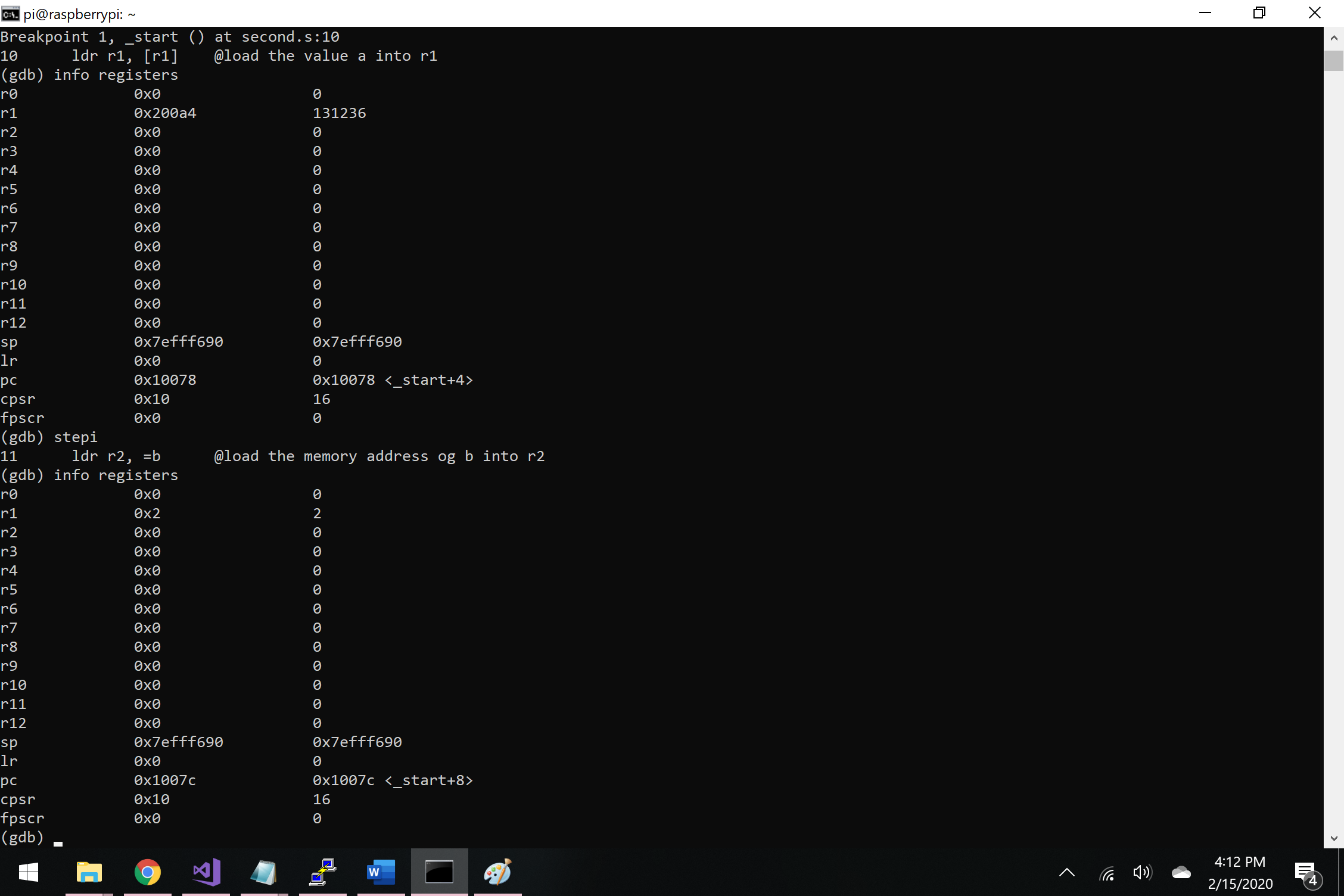


Figure 3 – registers after line 10 and 11

The first couple of lines of code load the memory and then values to the registers. To see the actual values of the registers, I can use the command “x/nfs address” that helped me further examine the memory content. From Figure 3, I can see that the address of r1 equals to “0x200a4”, while Figure 4, shows the address of r2 which equals to “0x200a8”. By typing this command during the debugging mode, I can see that register 1 holds the value of a=2 and the register 2 holds the values of b=5 (Figure 4).

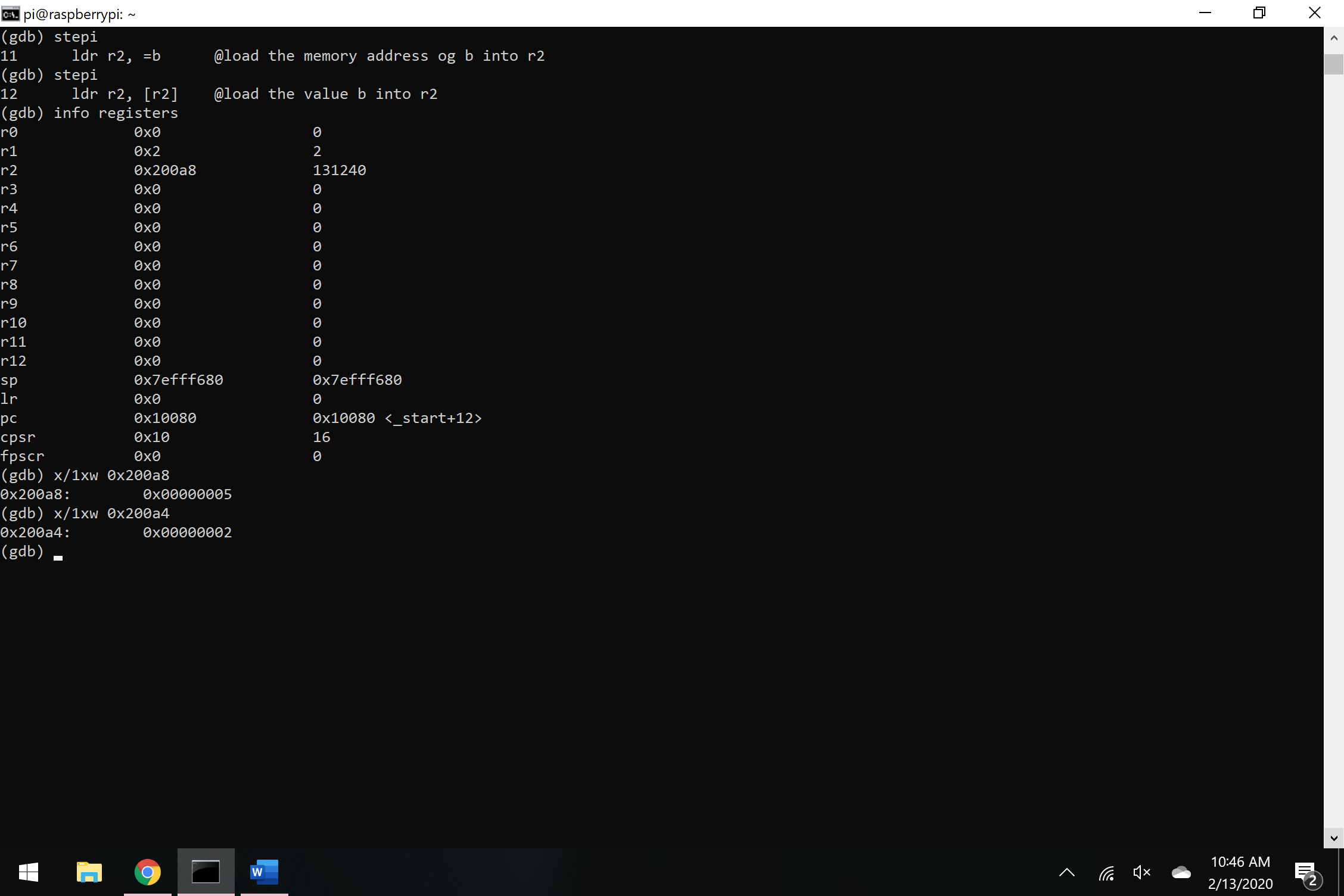


Figure 4 – examining the memory

**Part b:**

By using the template from part a, I wrote the arithmetic2 program (Figure 5).

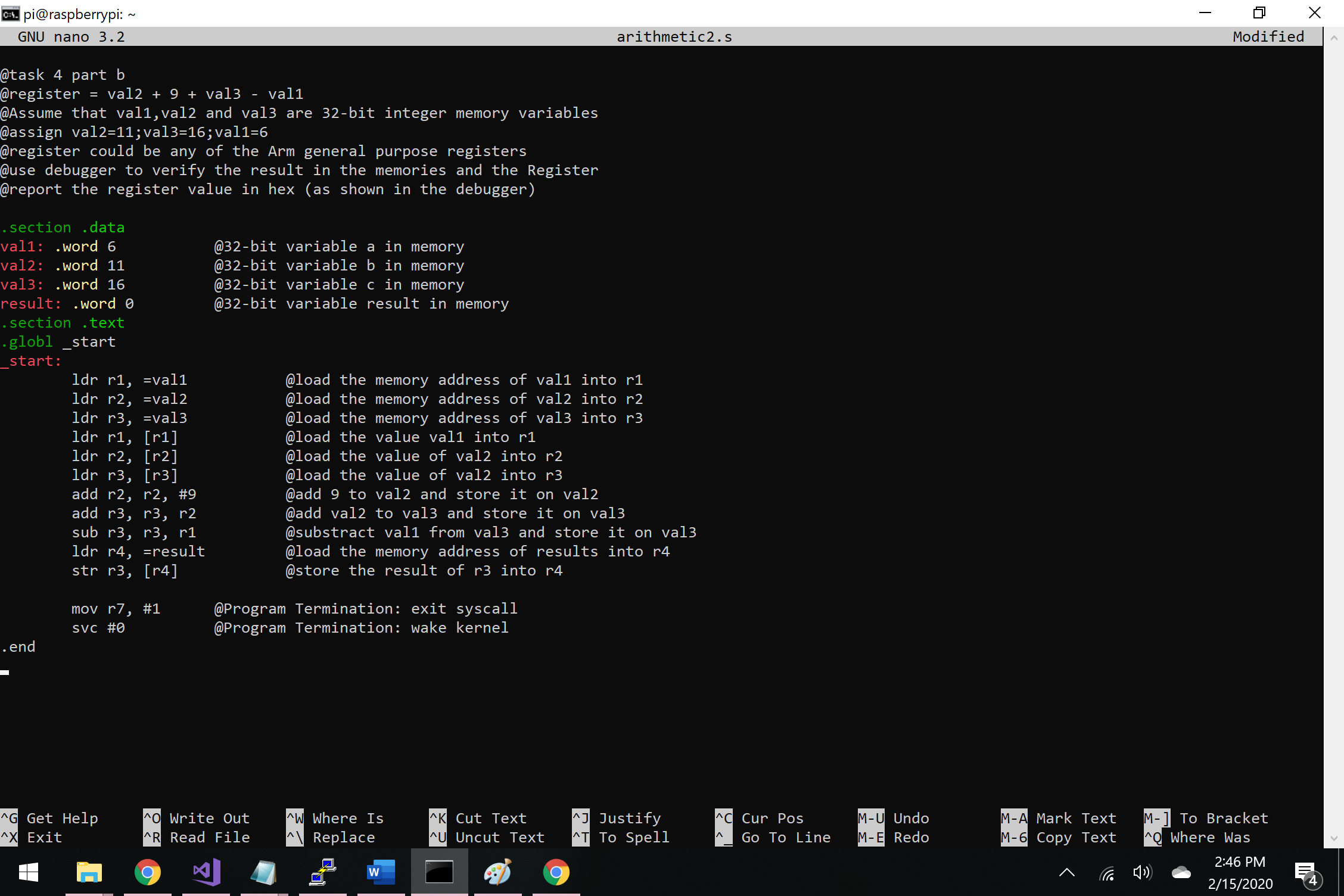


Figure 5 – arithmetic2 program

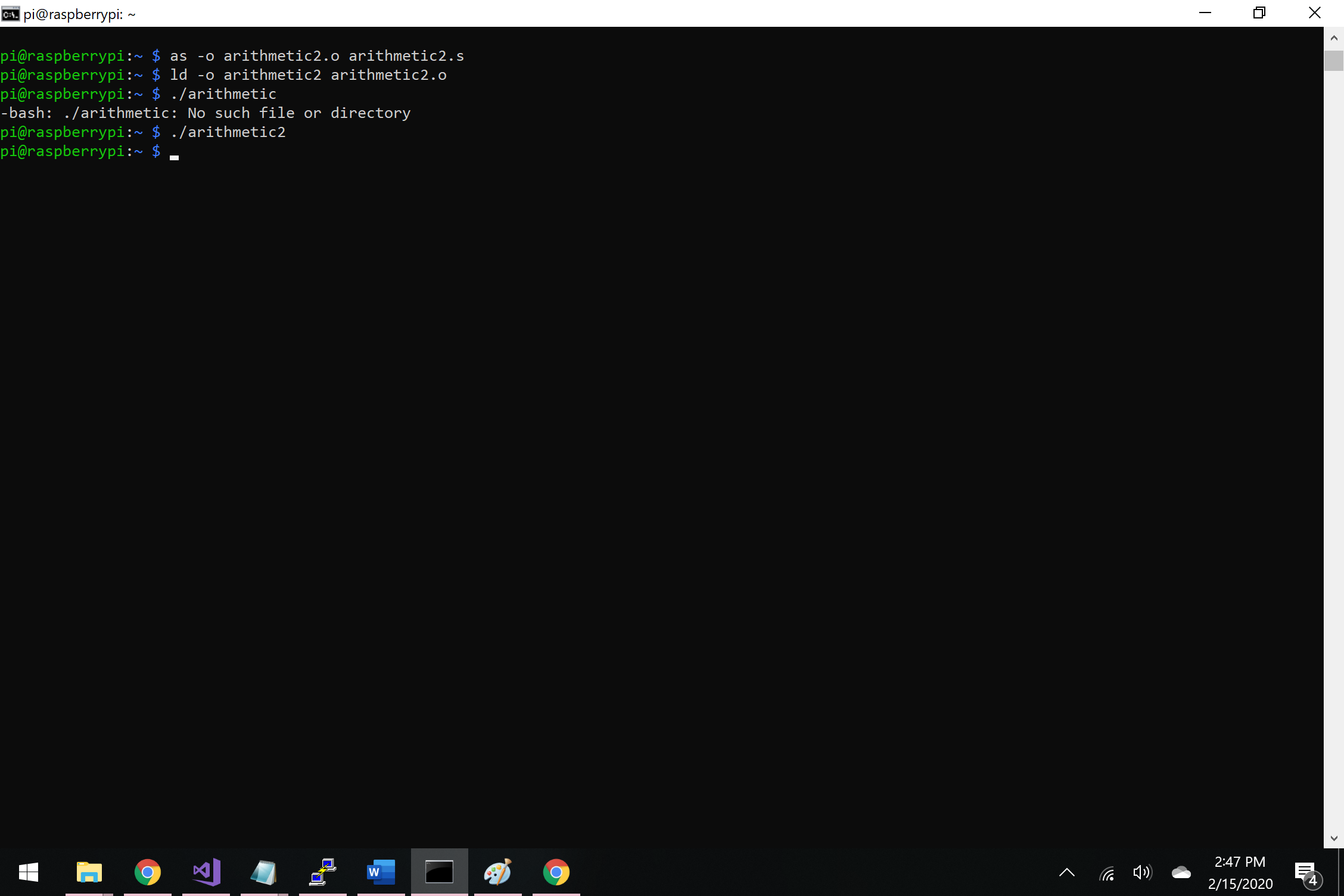
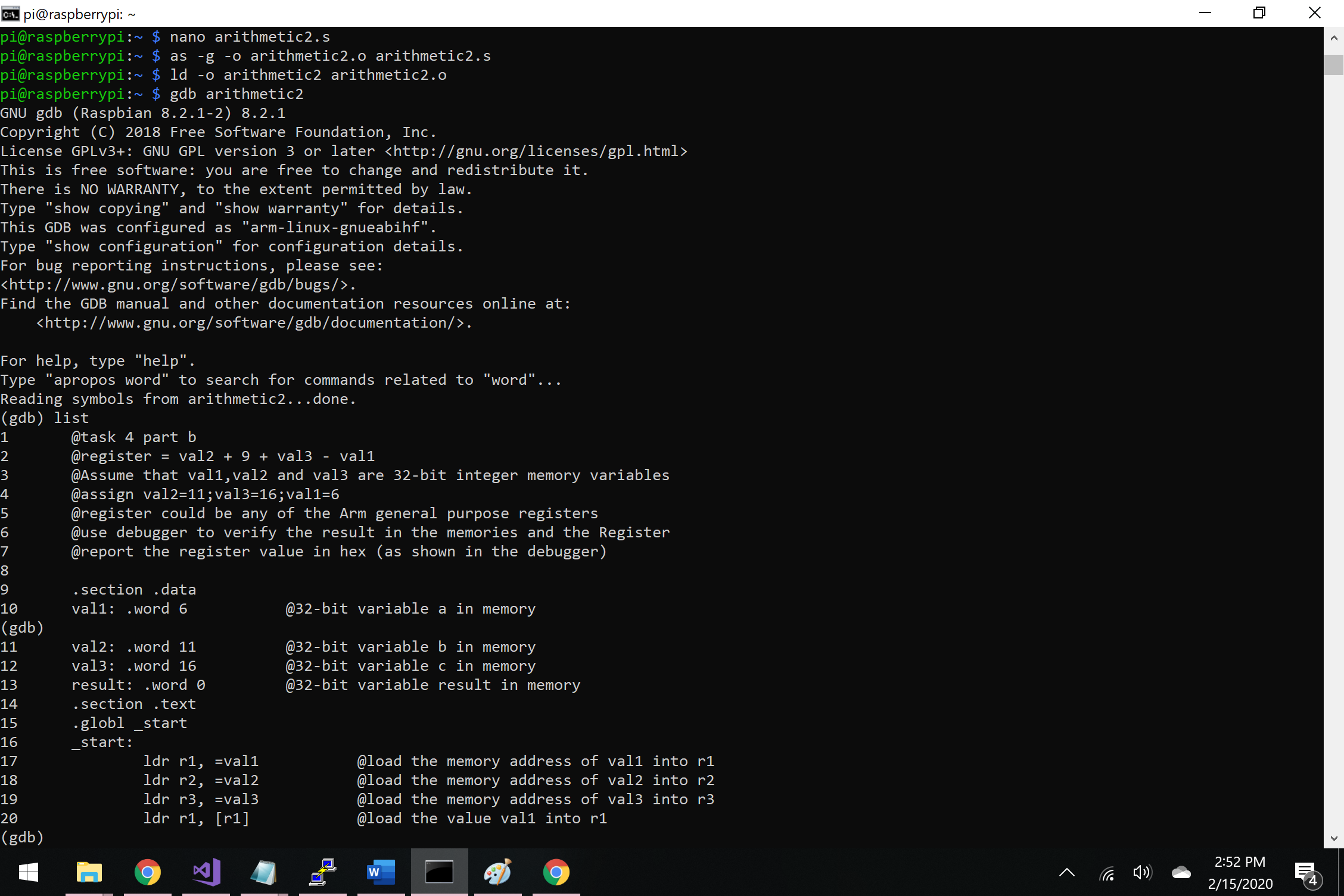
After compiling and running the code, just like in the previous example no output was shown, which is exactly what I expected (Figure 6). 

Figure 6 – running arithmetic2 program

To see what the program does line by line, I began debugging my code by firstly linking and assembling the program (Figure 7). Then I used the command “list” to see all the lines of code so I can choose my debugger breakpoints (Figure 8).

Figure 7 – linking, assembling and debugging arithmetic2 program

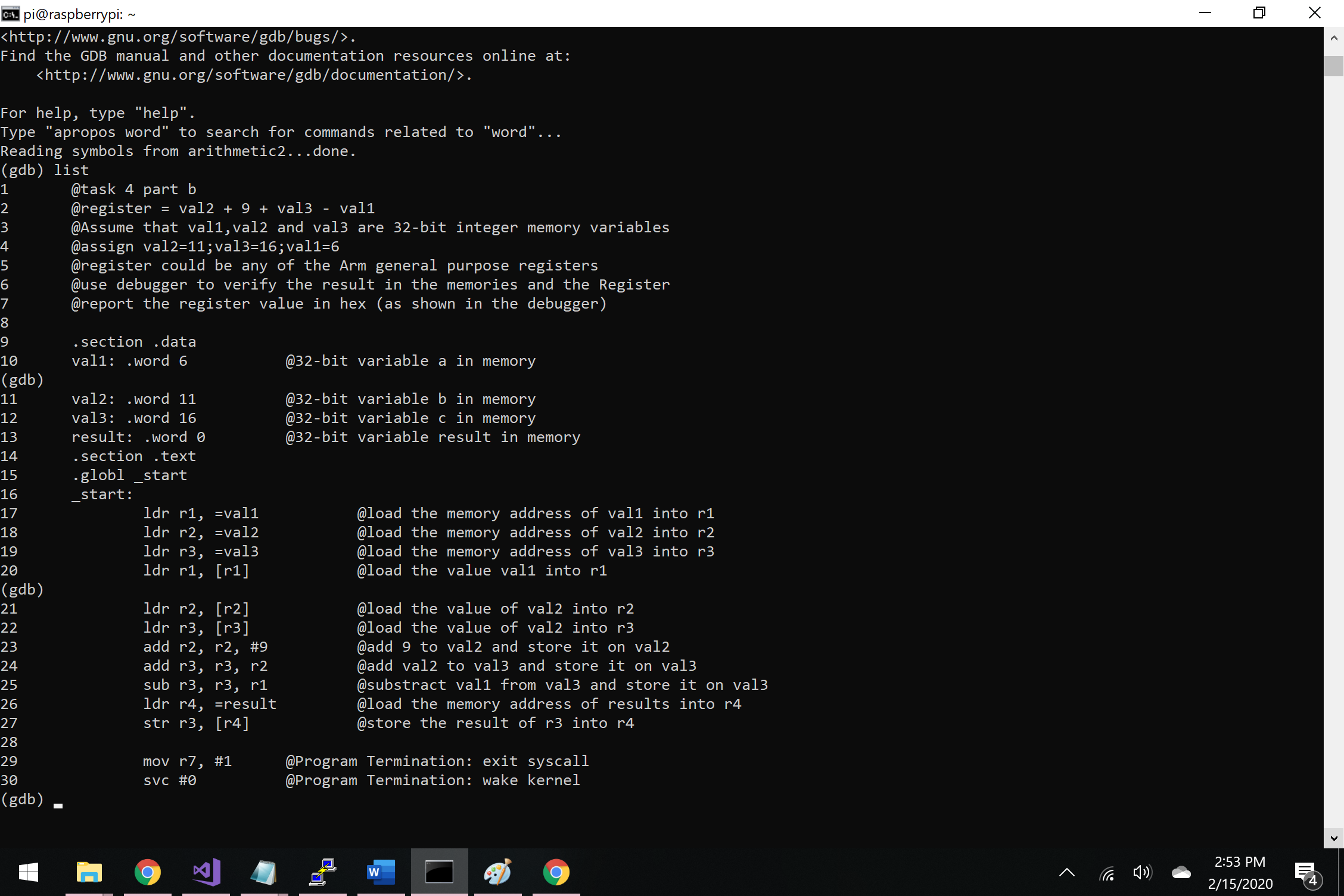


Figure 8 – listing all the lines of the arithmetic2 program

I decided to set up the first breakpoint at the start of my program, which is at line 17. Then, I used the command “info registers”, which told me the address of the register 2 – 0x200b8. By typing the command “x/4xw 0x200b8” I see the values that I set “00000006”, “0000000b”, “00000010”, which are hexadecimal values for 6, 11 and 16 respectively (Figure 9).

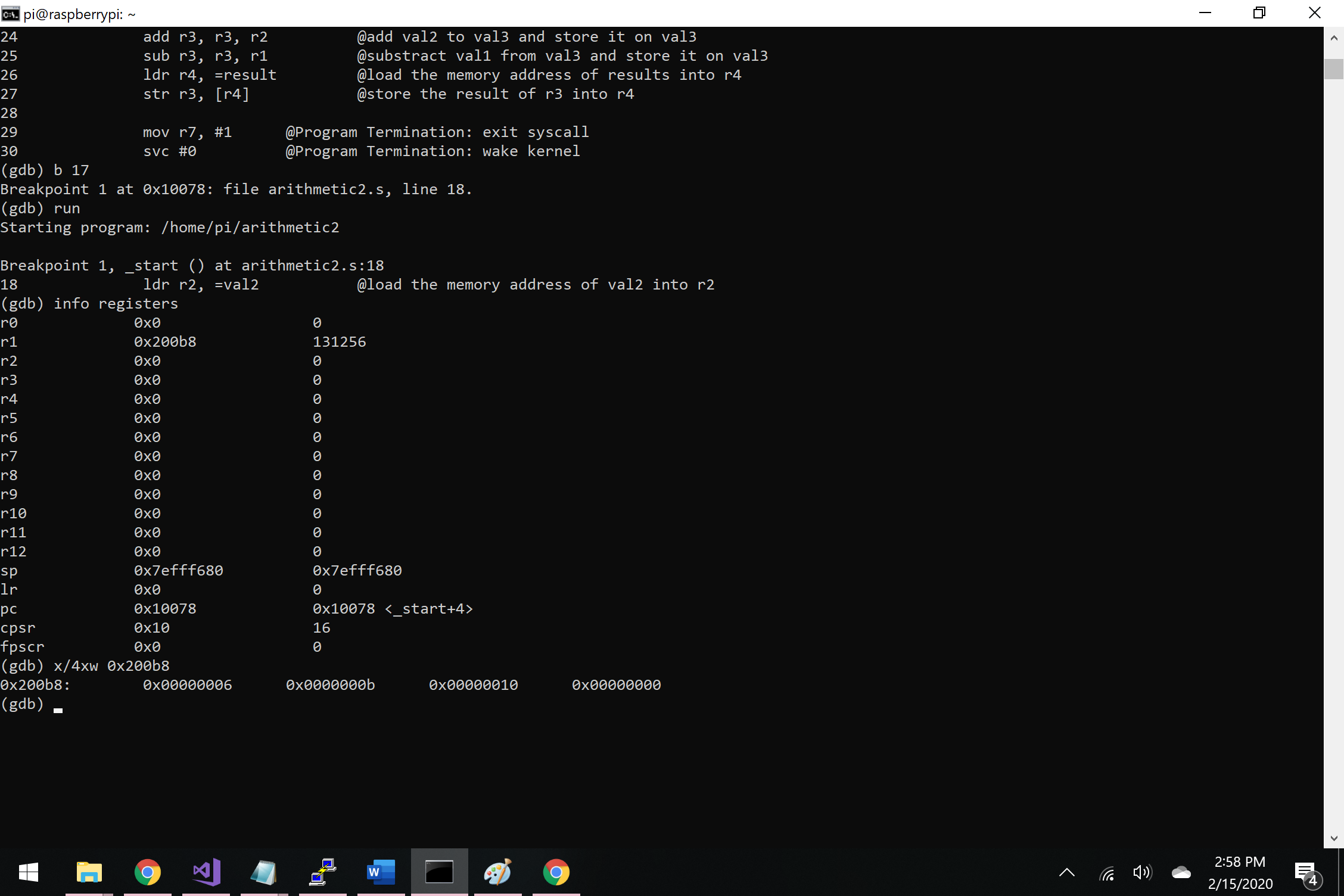


Figure 9 – debugging – setting the first breakpoint

Then, I kept typing “stepi” to see the change in my program. I noticed that on line 24, all three values (val1, val2, val3) are now stored into register r1, r2, r3 (Figure 10).

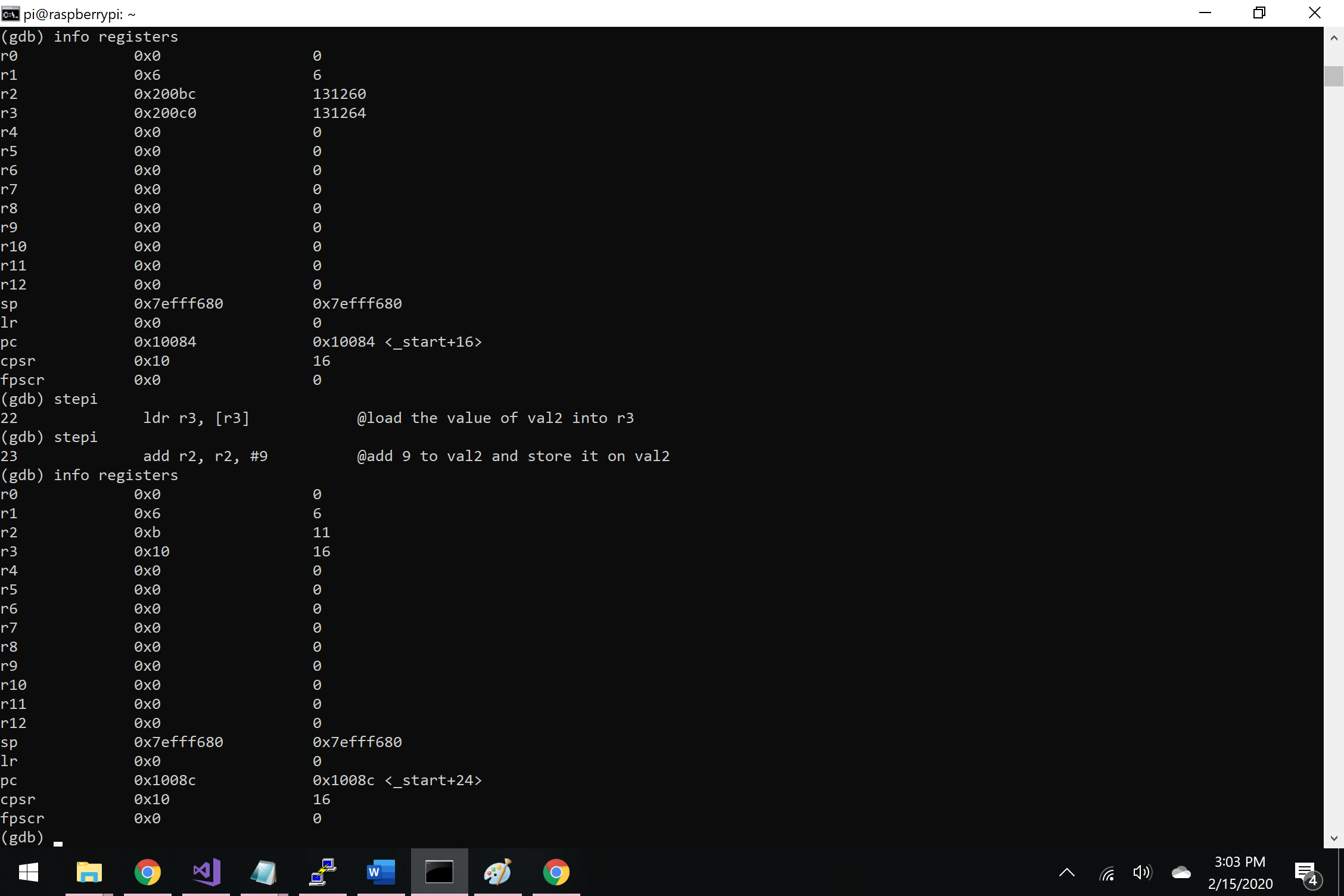


Figure 10 – values of the register r1,r2,r3

On line 24 and line 25 I can see the first addition has been made to my program. On line 24- the number 9 has been added to val2, hence the value 20 in the register r2. On line 25, the addition of val2 (20) to val3 (16) was made and stored in register r3 (Figure 11).

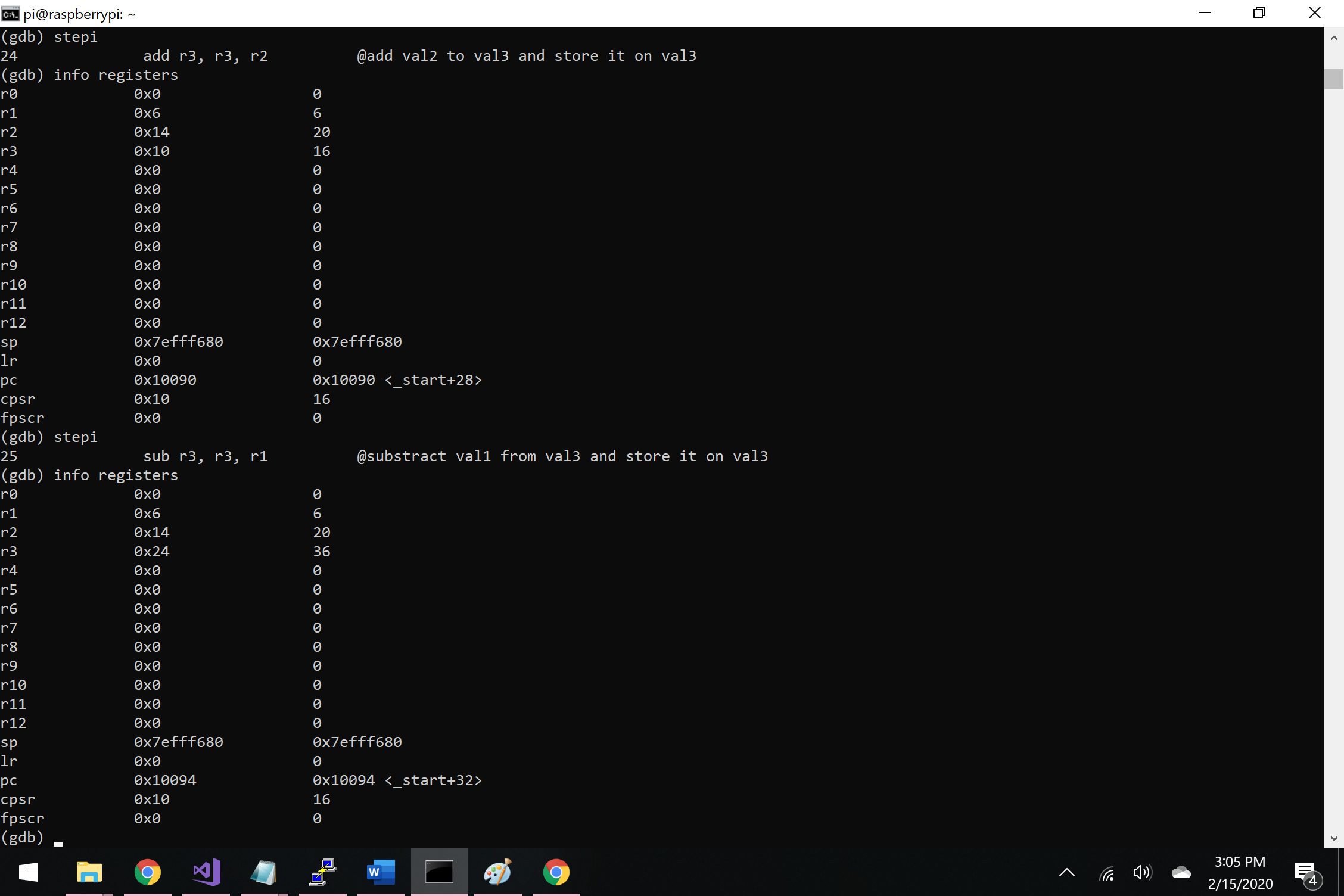


Figure 11 – registers after first additions changes

Then, on line 26 the program subtracted the value of register r1 (6) from register r3 (36) and stored in on register r4 (Figure 12).

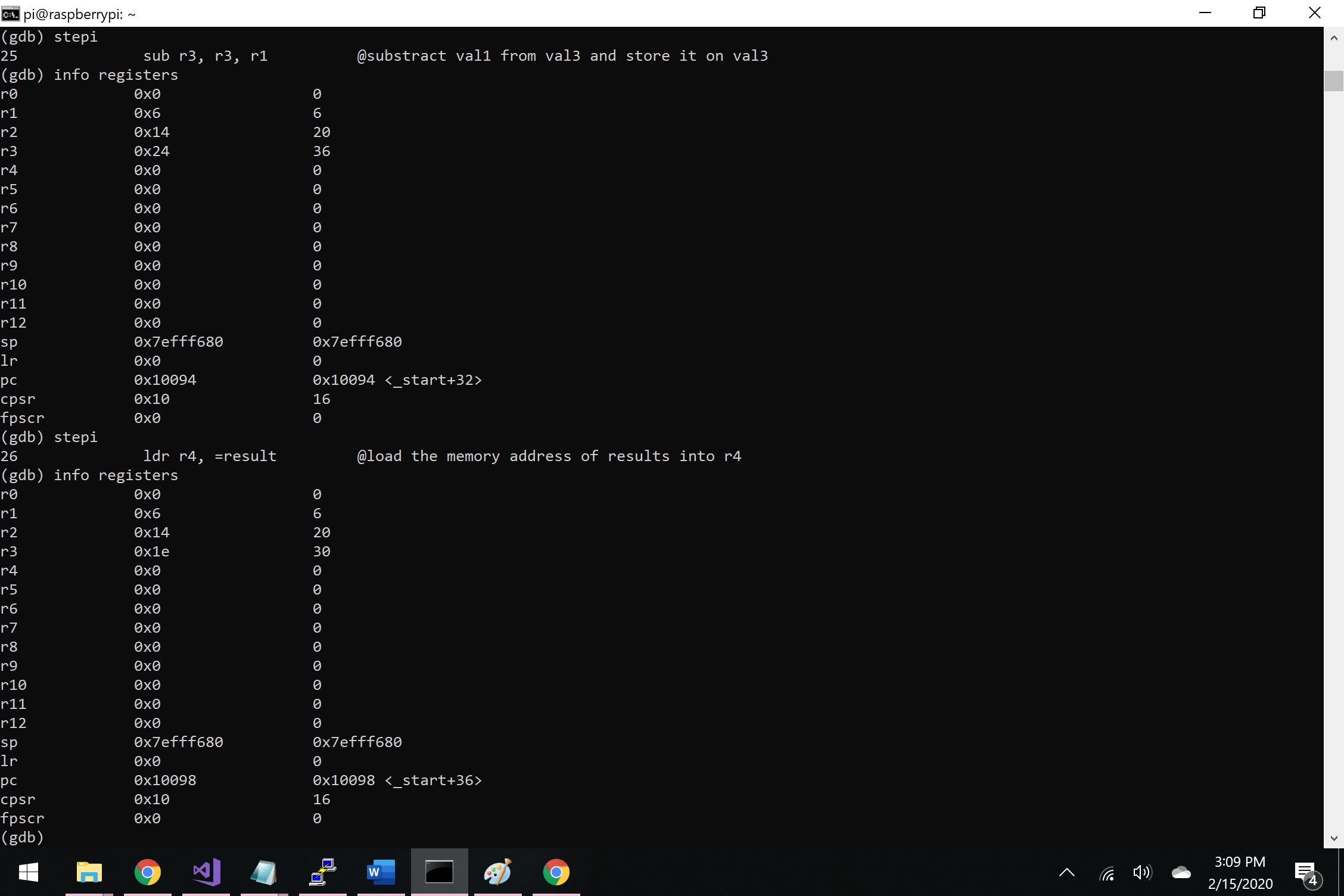


Figure 12 – registers after subtraction

Lastly but not least, after the program stored the result of the arithmetic expression on register r4. To further examine the register r4, I began typing “x/1xw 0x200c4”. The command showed “0000001e” which is 30 in hexadecimal (Figure 13).

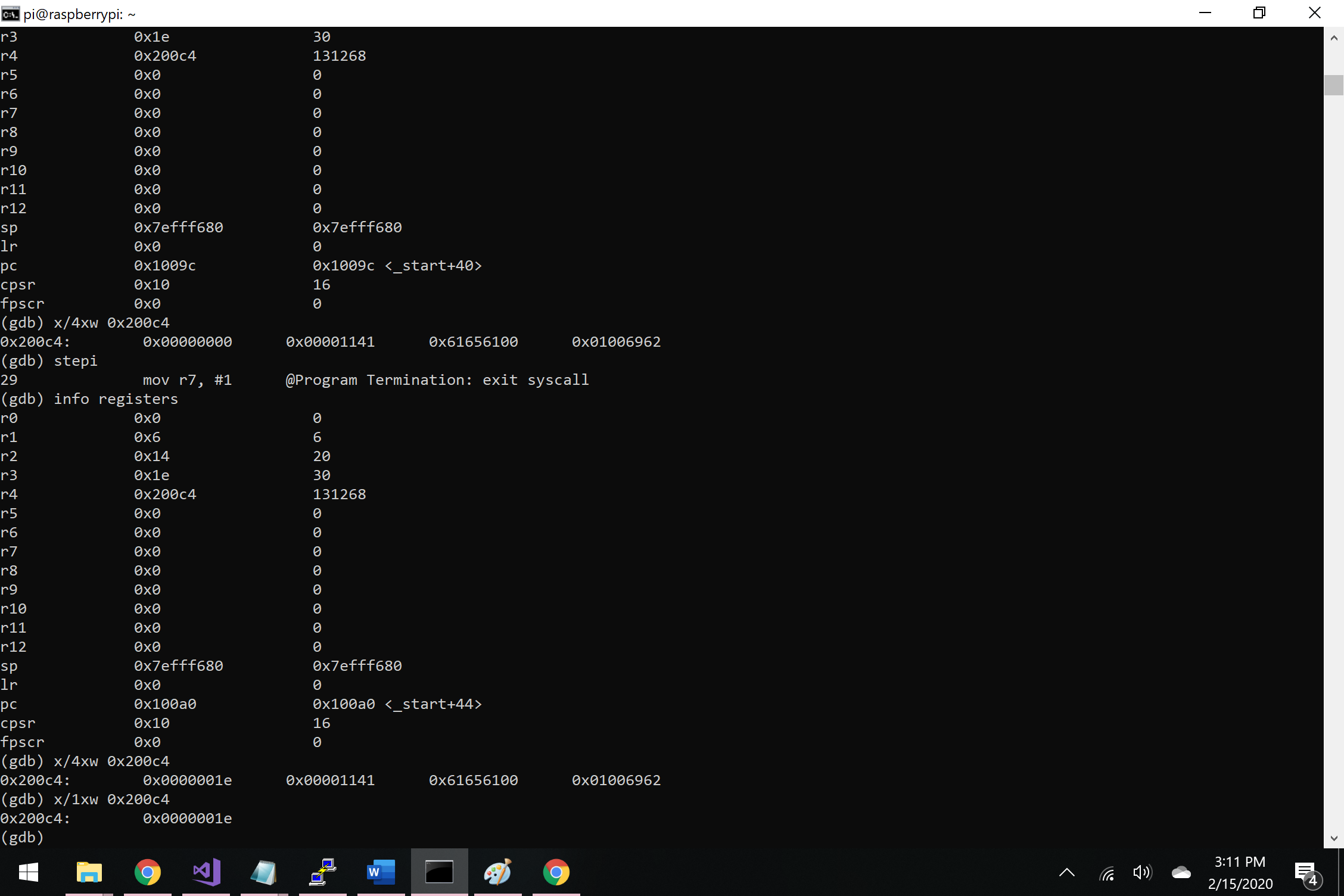


Figure 13 – final result of the arithmetic expression val2+9+val3-val1