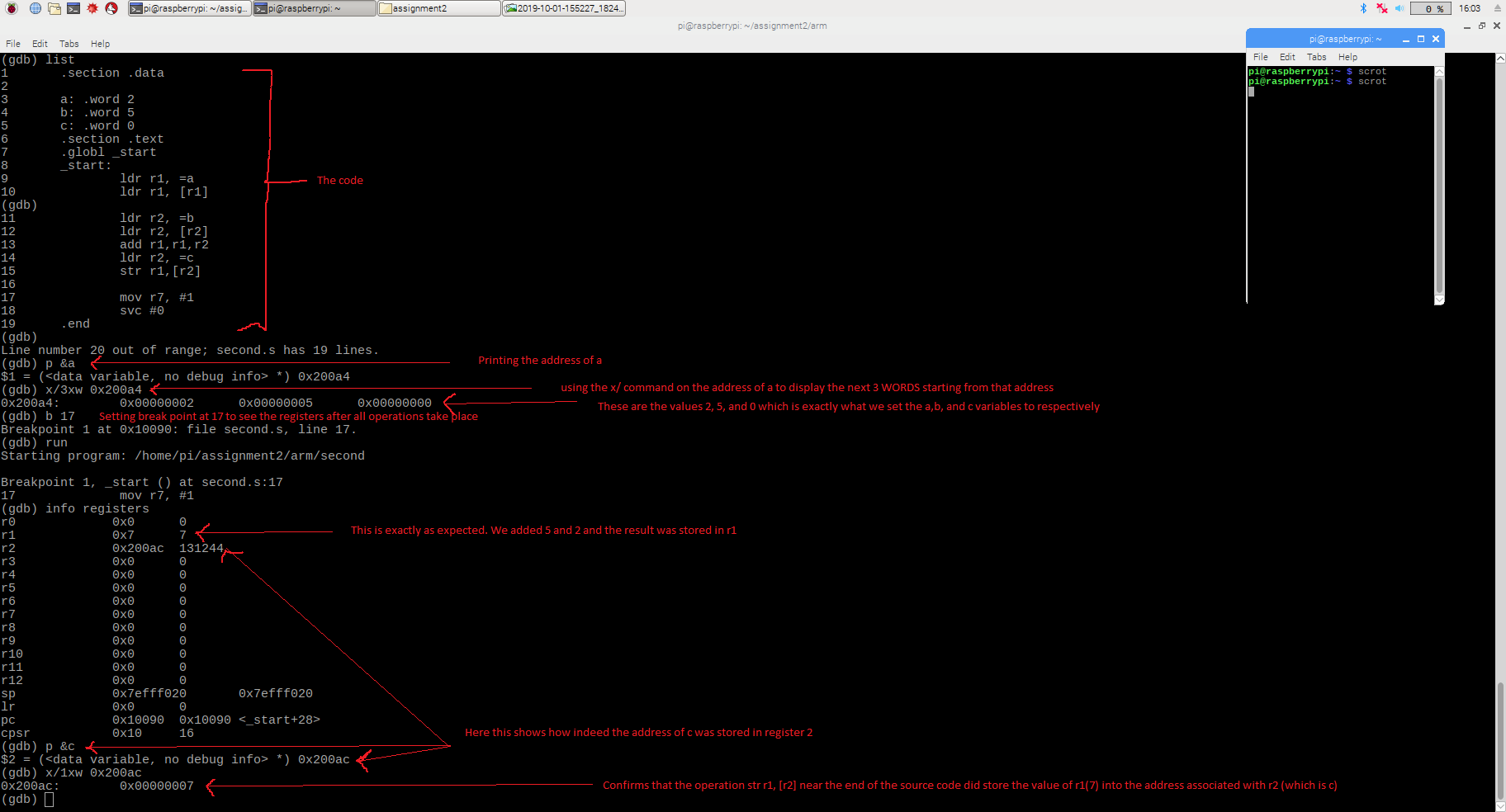
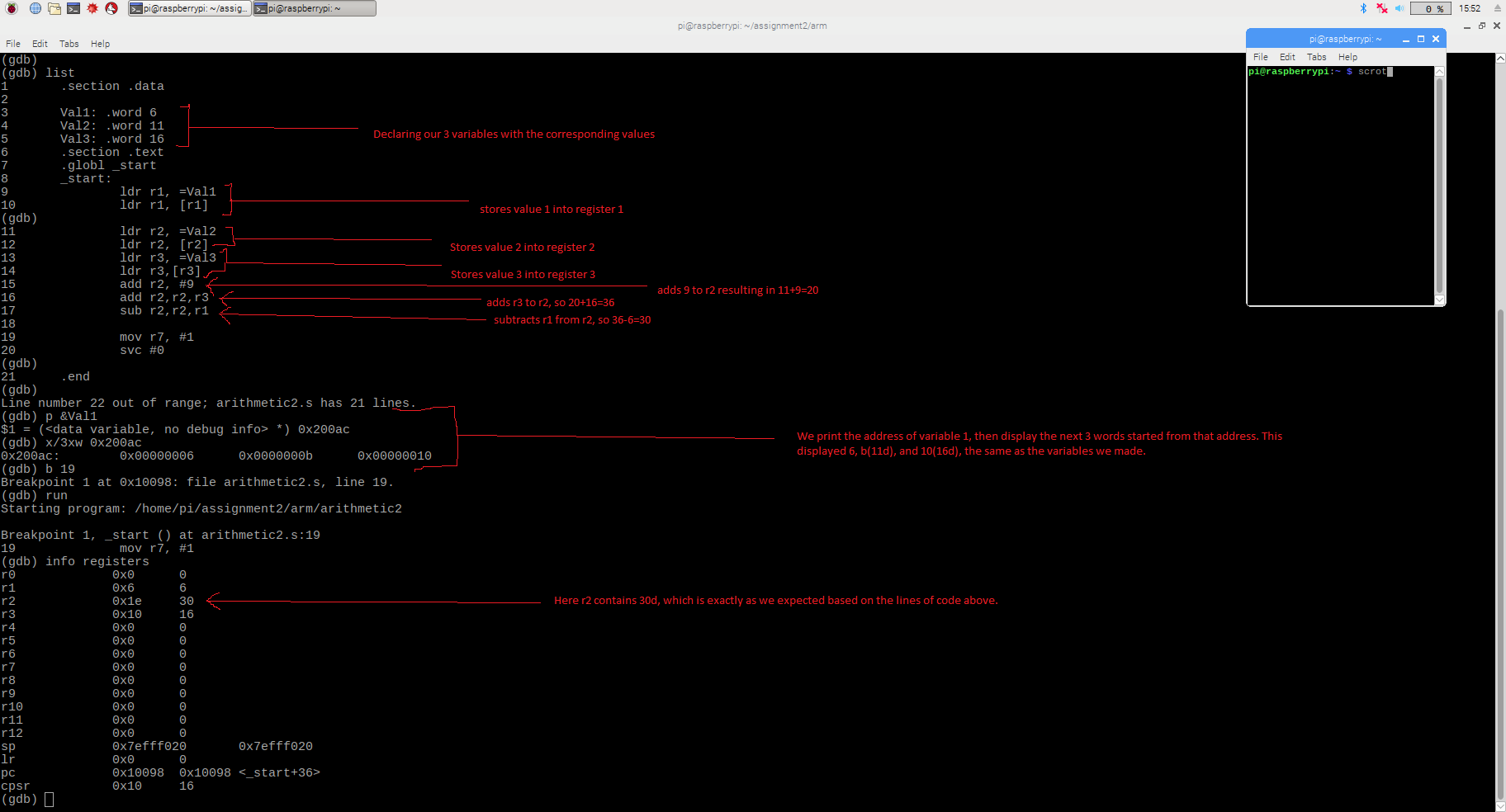
Arm Assembly

For part 1 we simply typed out all the code supplied in the instructions. We went through the steps given next in order to assemble and view with gdb. We used p &a to find the address of the variable a, then used the x/3xw. This showed the first 3 WORDS starting at the address we got from p &a, showing the values 2, 5, and 0. This showed how the variables were stored as expected in memory. We then put a break point at line 17 so that all operations we are concerned with were done and then examined the registers. 7 was stored in register 1, as expected since all that was done was adding 5 and 2 and storing it in r1. R2 contained an address that we confirmed was the address of the c variable with p &c, which is a result of “ldr r2, =c” near the bottom of the source code. The line “str r1, [r2]” was supposed to store r1’s value into the memory address associated with r2, which is the address of the variable c as we confirmed above. To see this we typed x/1xw 0x200ac(the c address) to see what value was stored at c’s address. The value was 7, which is the same as was in r1.

See appendix picture(whatever number you put it as Akiva) It explains all these things next to the lines in the terminal for a better visualization



Part 2 required do the same thing as the mathematical expression but in coding. First we made 3 variables: val1, val2, and val3 and stored the values we were told to in the instructions. We stored the three variables in three different registers, similarly to part 1. Next we added 9 to r2 making 11+9=20. Then we added r3 to r2, making 20+16=36. Next we subtracted r1 from r2, making 36-6 =30. We examined the registers after making a break point and found than indeed register 2 contained 30d.



See appendix picture number…