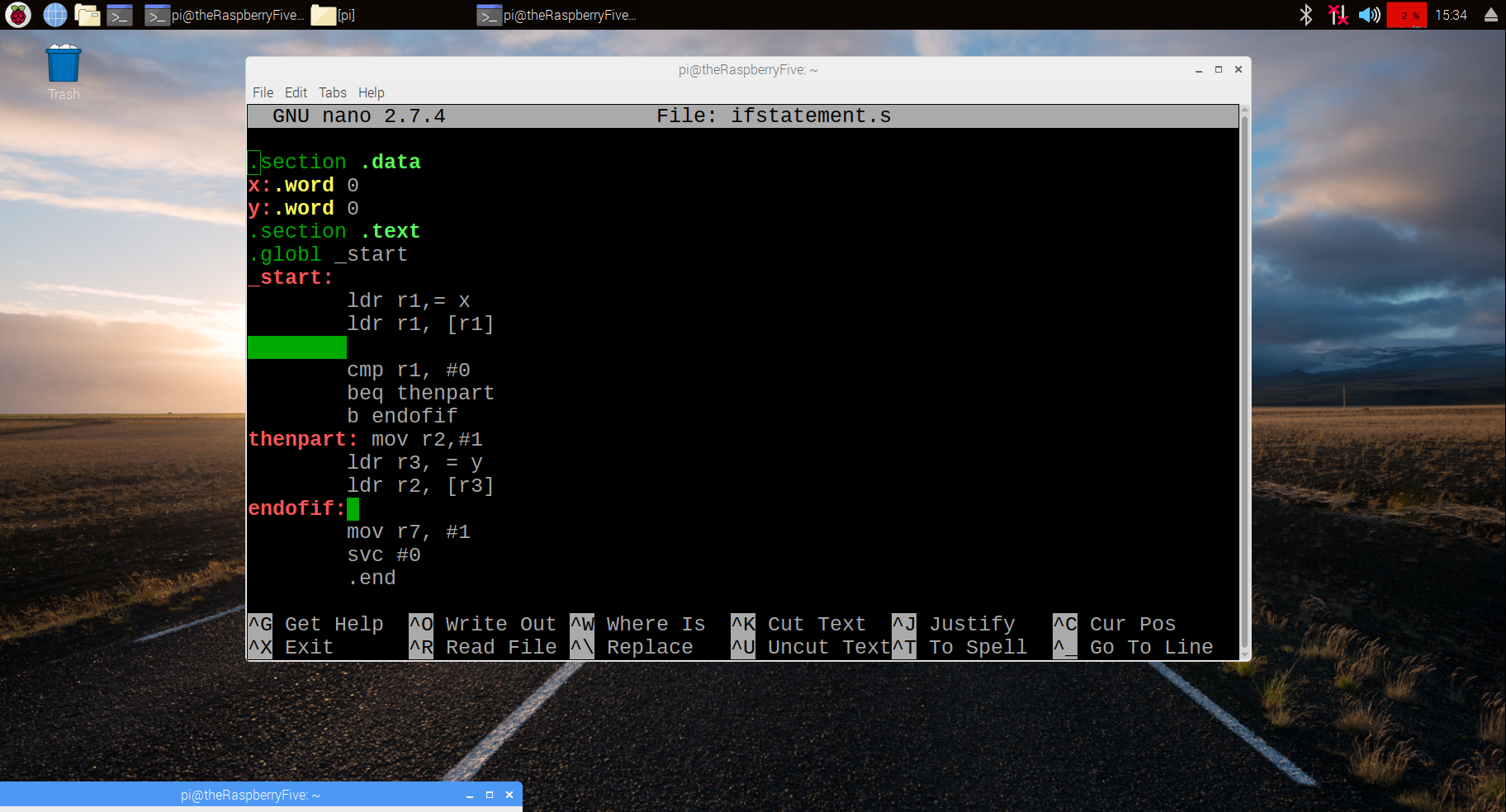
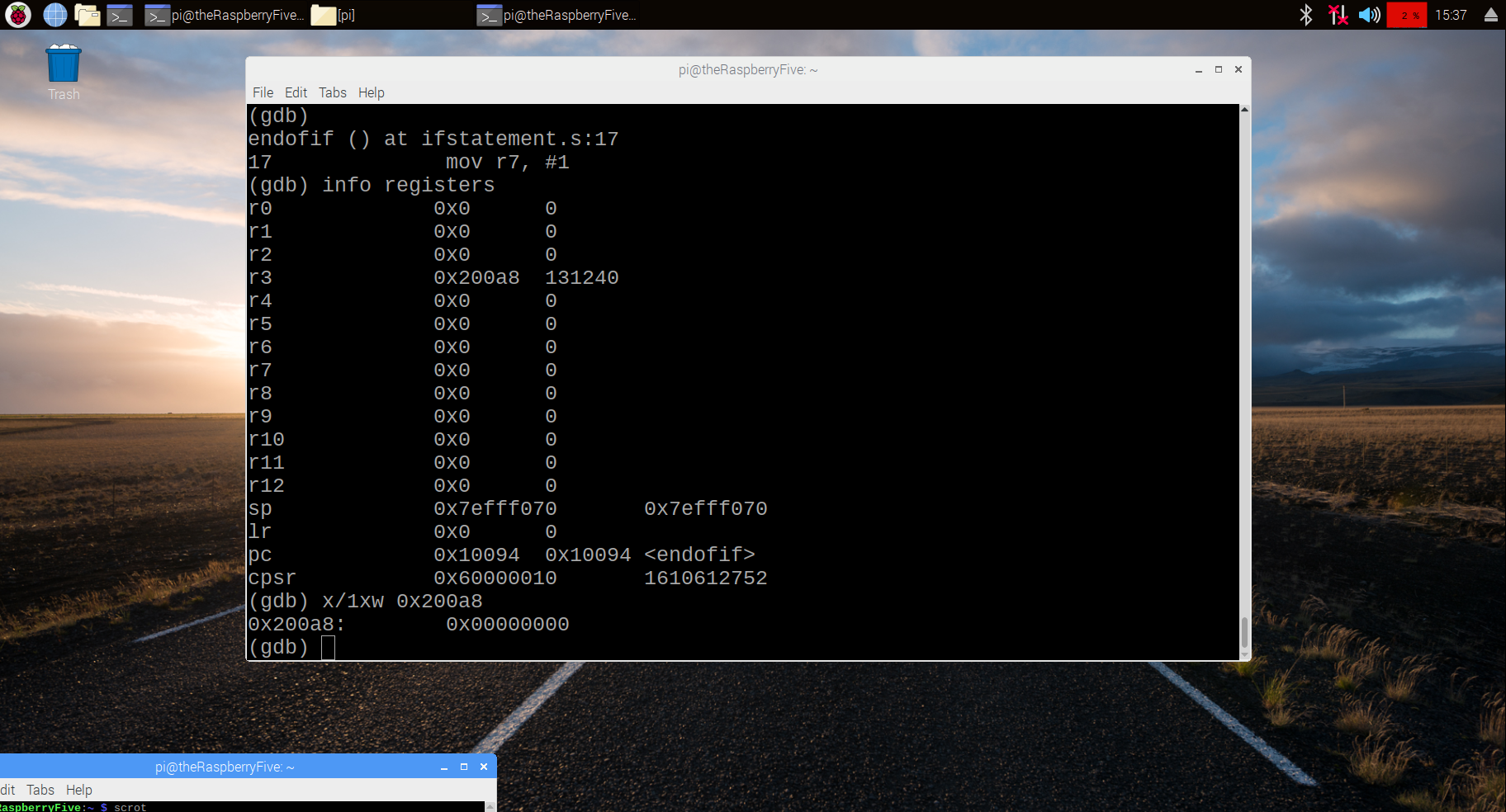
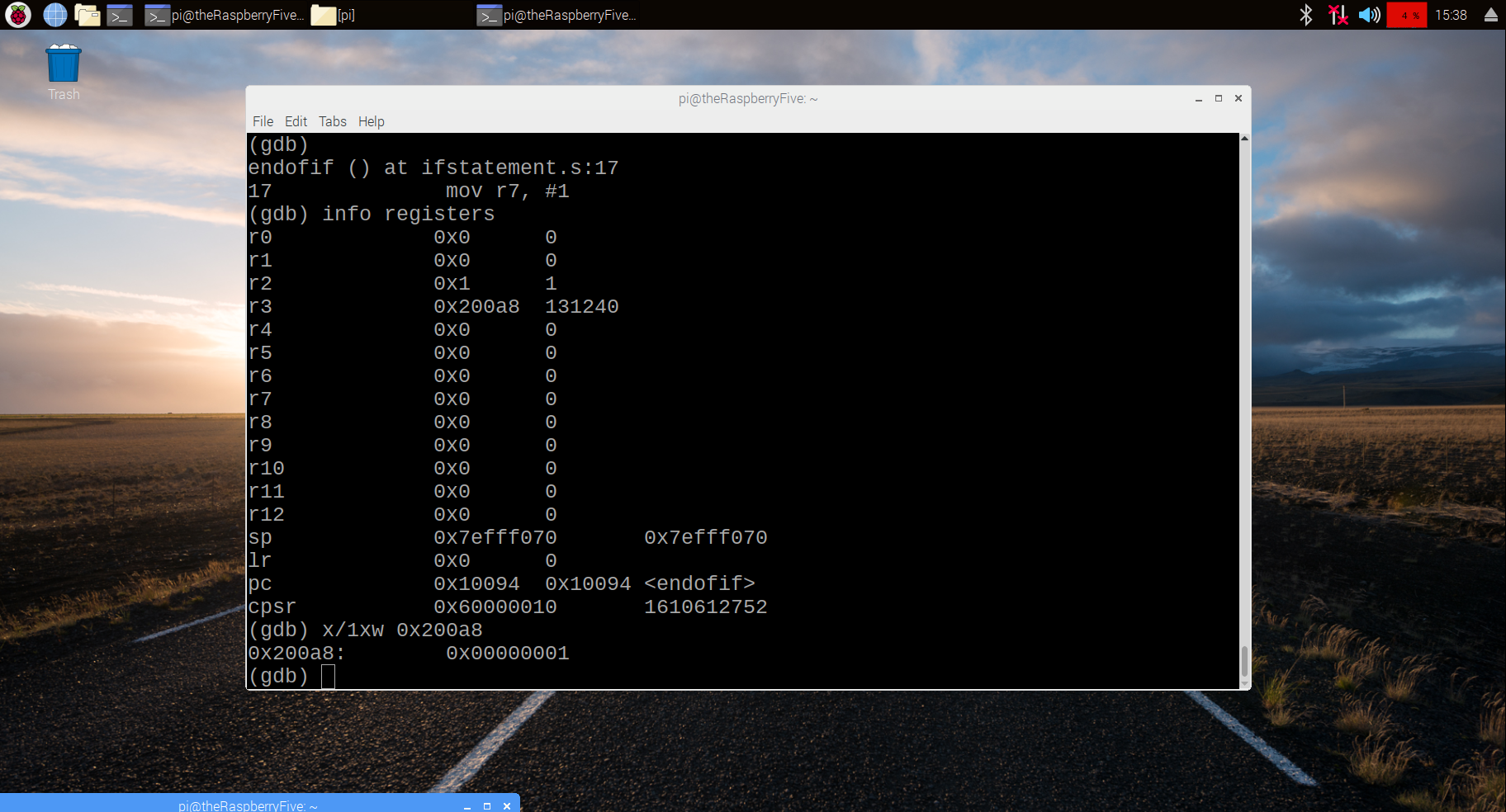
For Part A, I wrote code designed to create an if loop that looks at the value of x and gives an output based on whether or not x is equal to 0. The first error that I ran into was when I was trying to assemble the code. For the statement where I moved 1 to r2, the code was lacking a “#”. This error stopped the code from assembling correctly. Once I made the appropriate edit, the program assembled without a problem.



I then loaded the program, added a breakpoint at line 7, and stepped through the program. After it ran, I displayed the memory addresses in binary. This made it easier to identify the flags and their states. To examine the memory of y, I looked at the memory location 0x200a8 because that is the value stored in r3. I used that value because r3 holds the memory location of y. The value for y was all 0s. This was incorrect, as the result should have been that y was 1. 

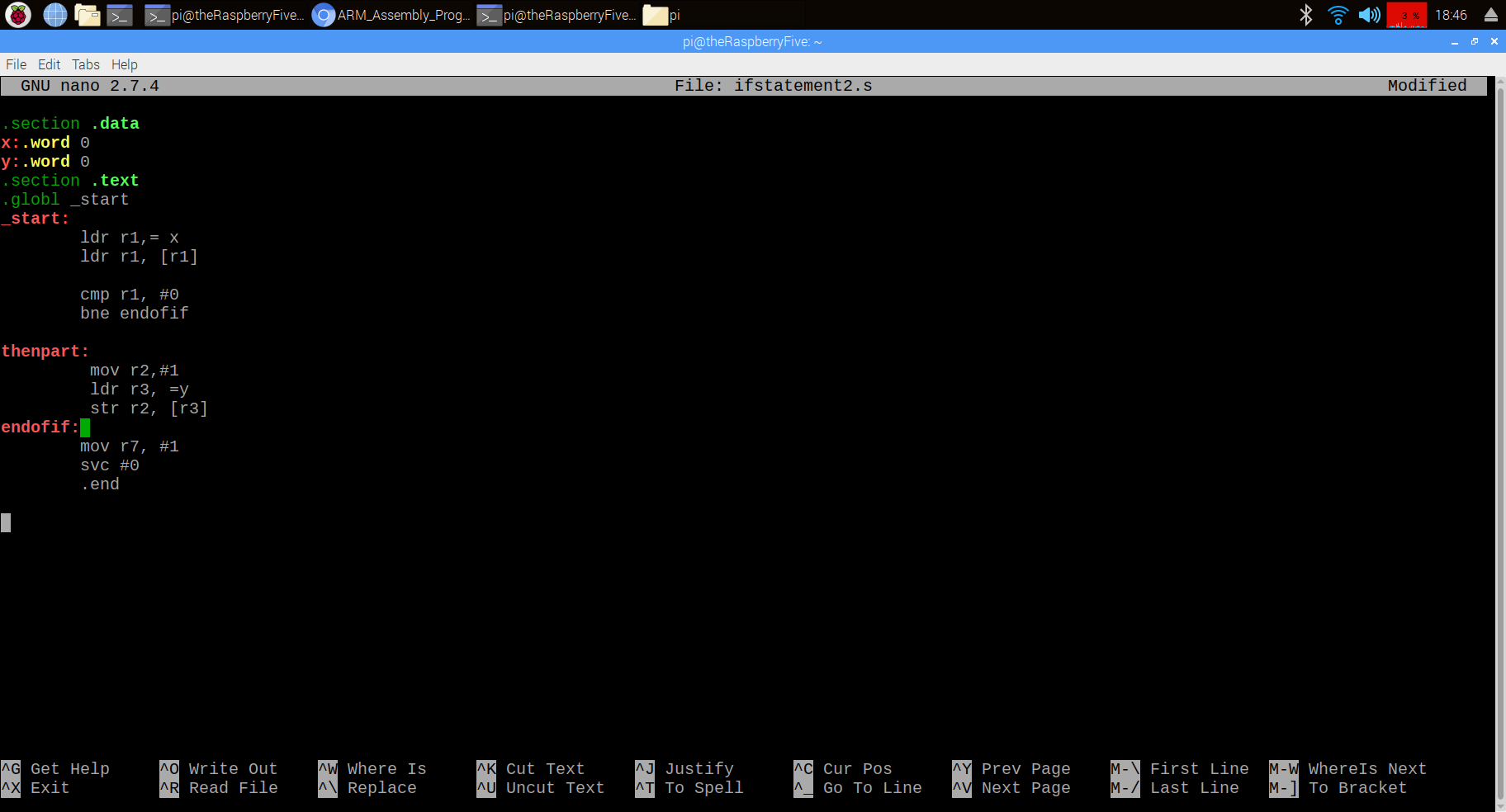
After I examined the code more closely, I realized there was an issue with the code; there was a loader command where there should have been a store command. I made the switch and the code returned expected results.



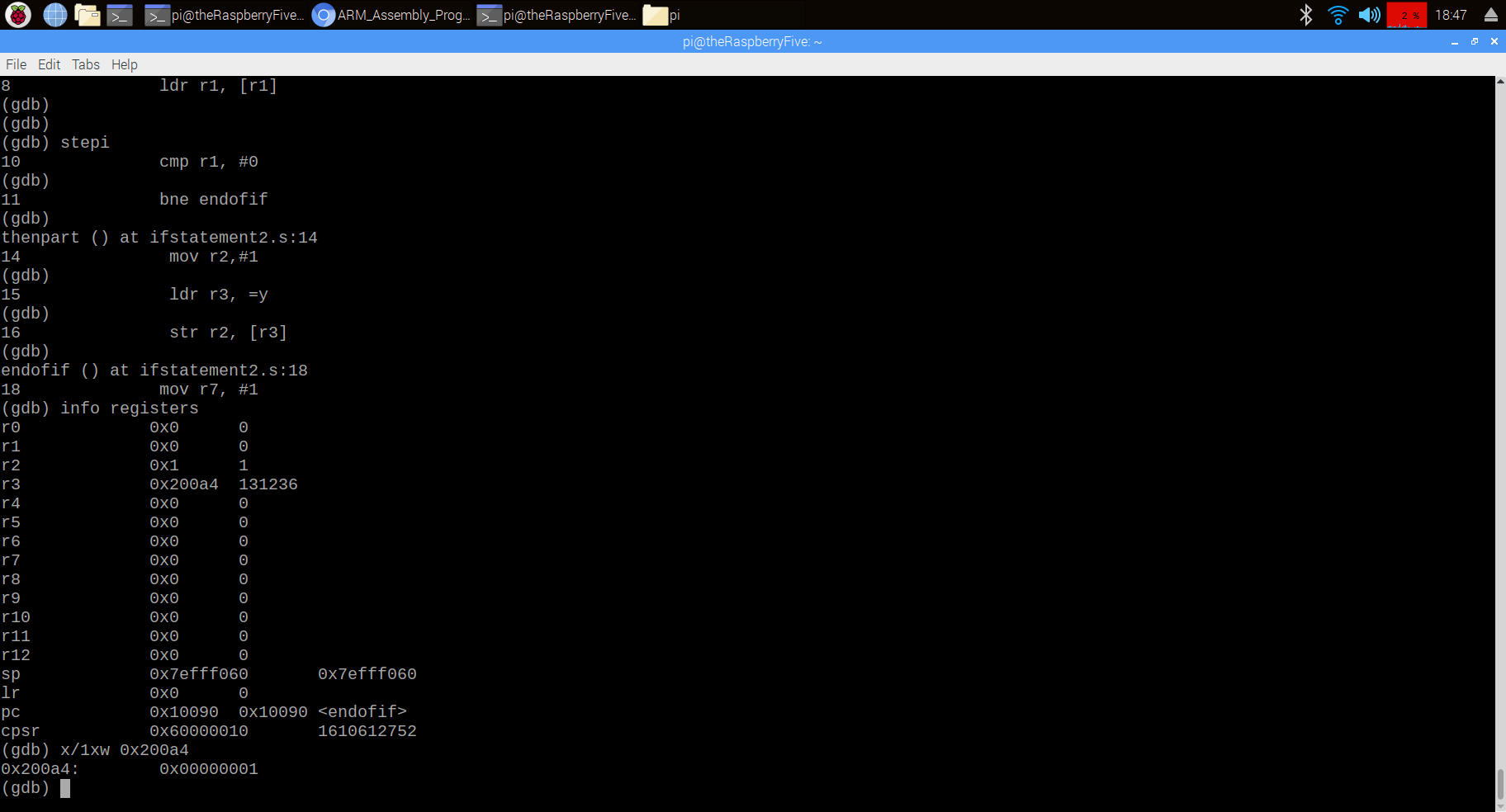


After the program ran, the value of y was 1. Since the value of x was 0, the Zero flag was triggered. This is because the Zero Flag is changed based on the outcome of the compare command. The Zero Flag was 0 because the value of x (0) is equal to the number 0. I know this because the first digit in the cpsr register was 6. Because cpsr is in hex, that digit represents a byte. After converting 6h to binary, you get 0110. This shows that the Zero flag and Carry flag are triggered.

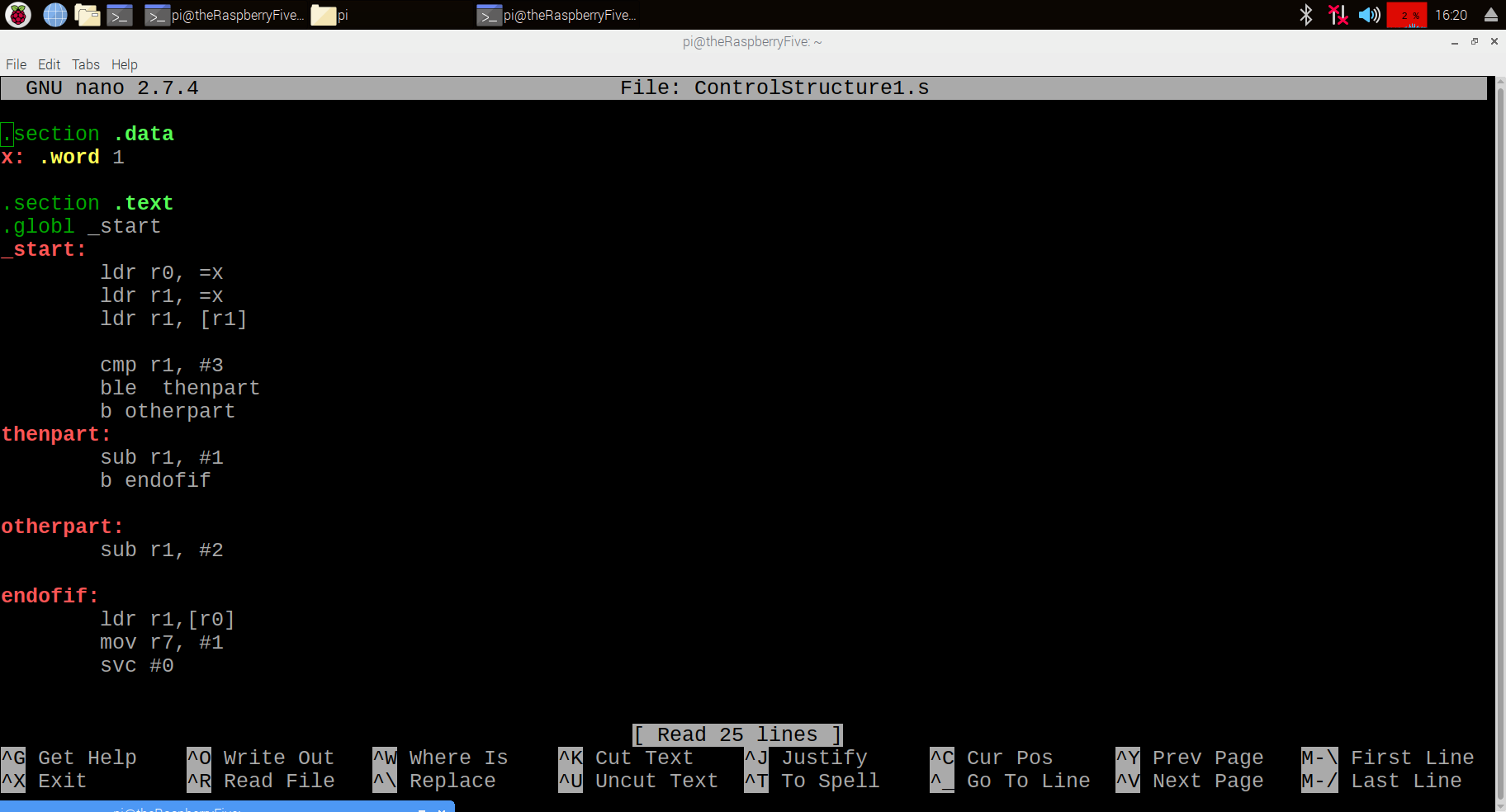
For Part B, I made a copy of ifstatement1 and named it ifstatement2. I changed the beq command to bne and removed the b command line.

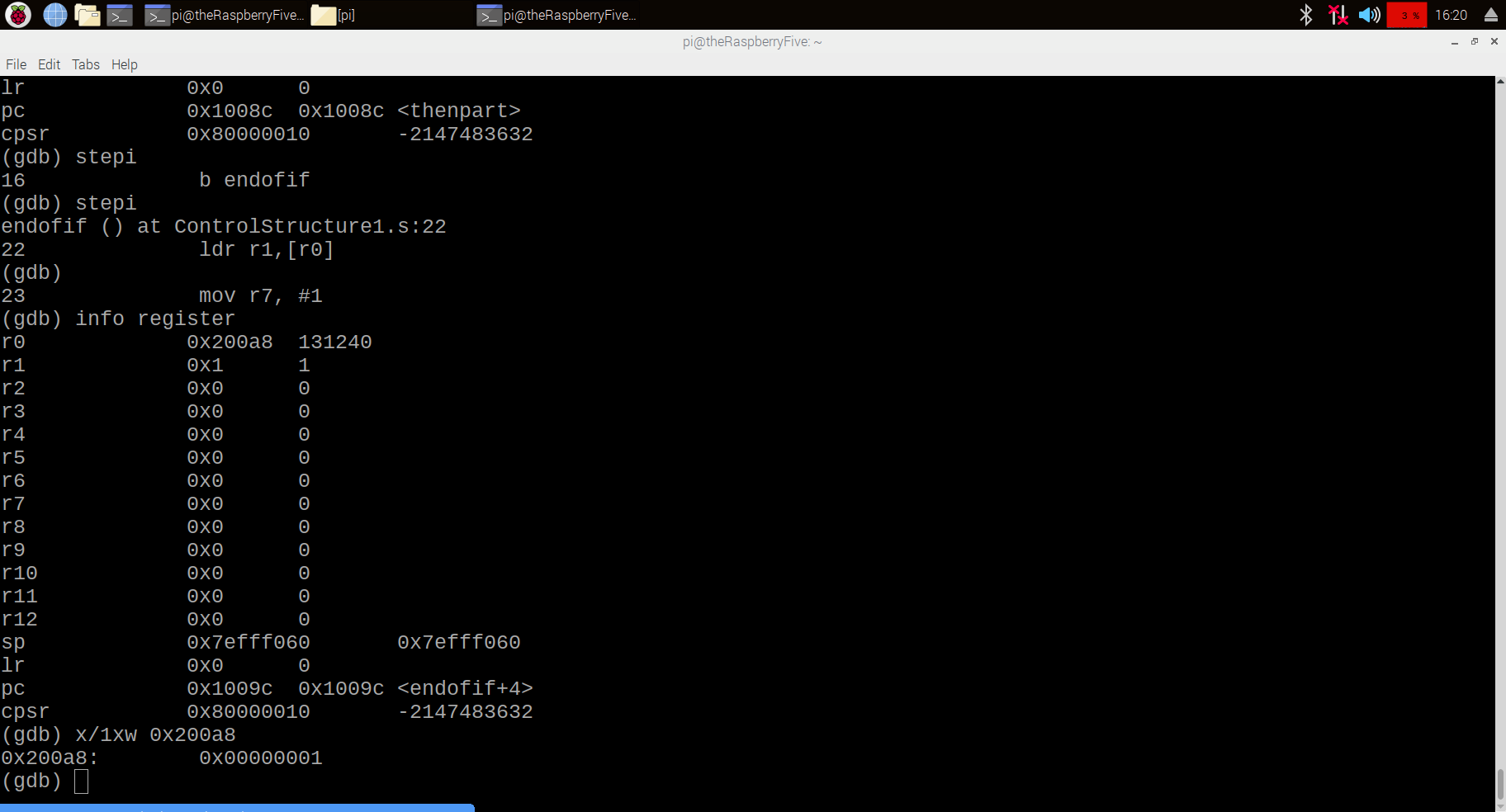


I then assembled, loaded, and ran the program. After examining the value of y(memory location at 0x200a4), I saw that it equaled 1. Accordingly, the Zero Flag was also triggered.

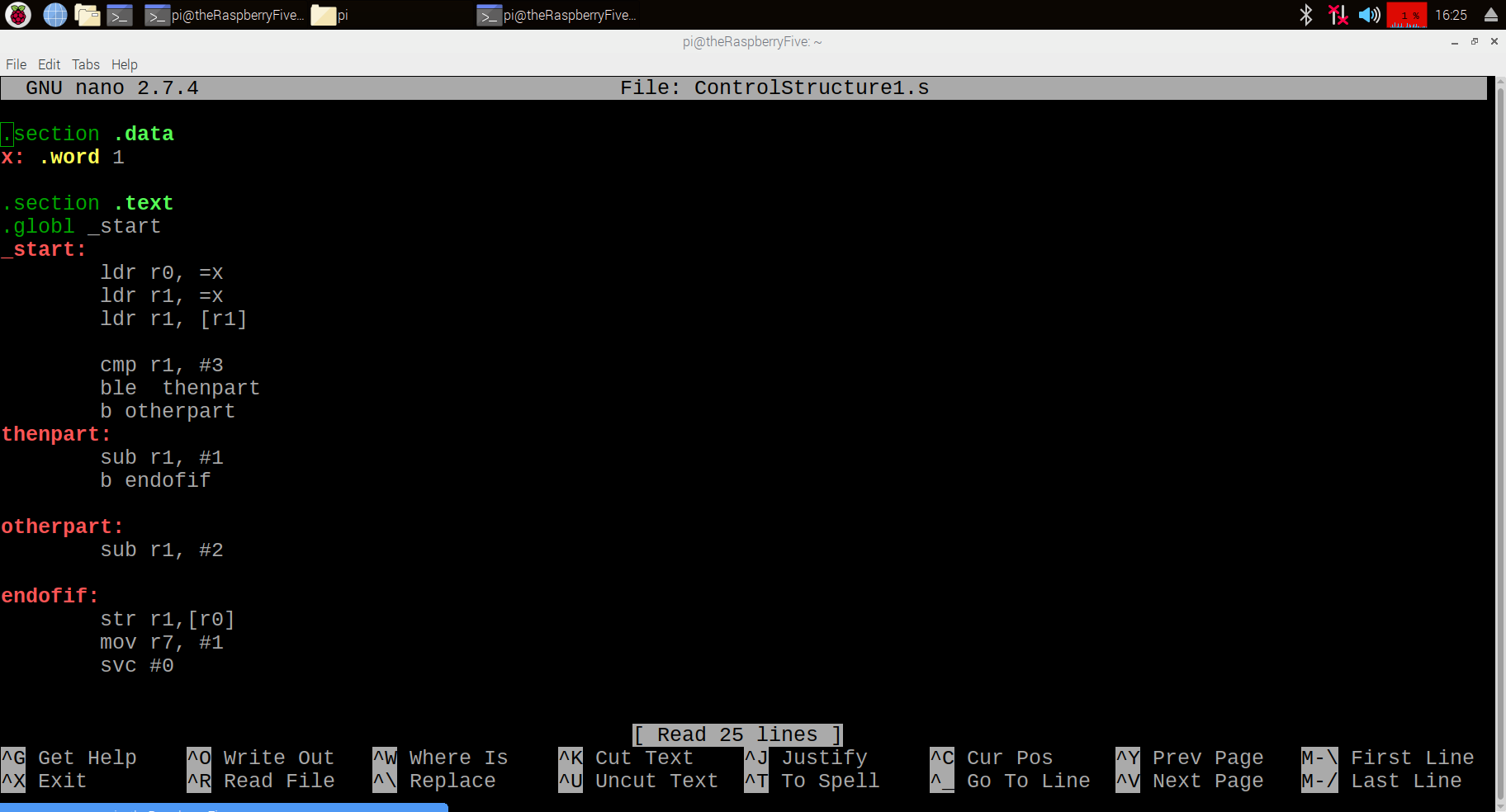


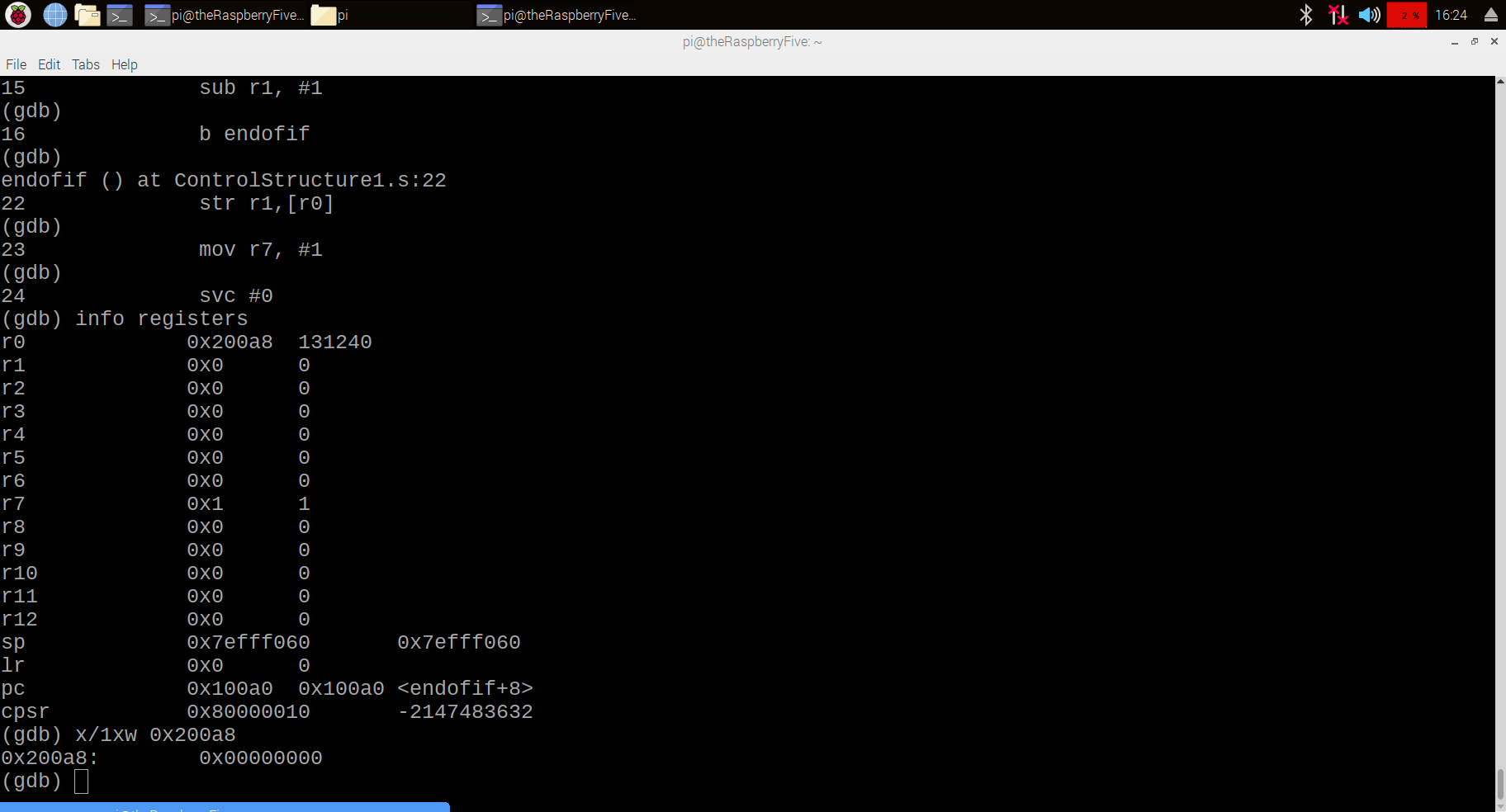
For Part C, I wrote a program called ControlStructure1 and ran it. I used code from the previous parts as a reference for this code. The code compares the values of x against 3. If x is less than or equal to 3, x equals x -1. If x is greater than 3, then the new value of x is x-2. The program runs as it should.However, I encountered an issue with assigning the final value back to the X variable.





The problem was solved once I replaced the loader command with the store command.





The final value of x is 0h. This happens because x is less than 3, which triggers the instruction for x to equal x - 1. The Z Flag is not triggered. I know this because the first digit of cspr, which represents a byte, has the value of 8. In binary, this is represented as 1000.