Fourth.s register info

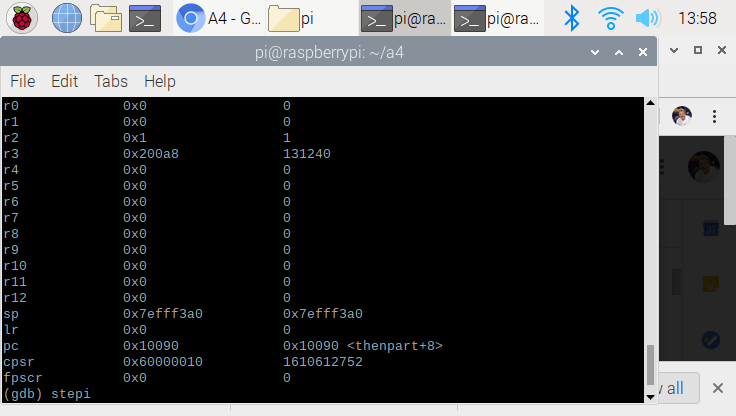


Purpose of the program: If the x is equal to 0, we will make y equal to 1.

From above photo, we know the program worked because the r3(y) is became 1 when r1(x) is 0.

Unfortunately, the code that was given to us (from pdf) is inefficient since there is two innter loop logical algorithm. The part 2 shows how to fix the inefficiency.

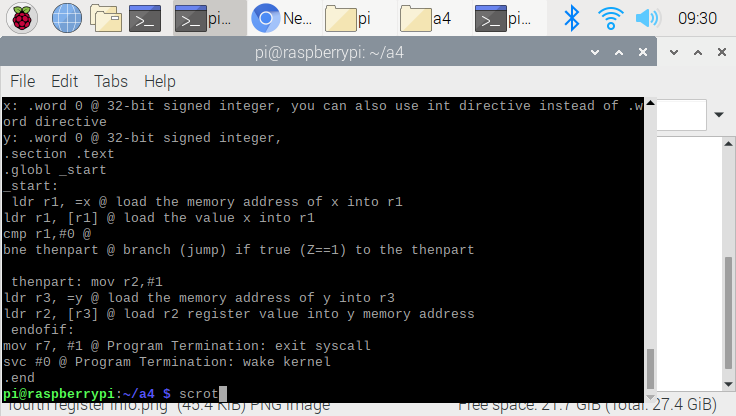
Fourth.s exit register info



This is register info right before the exit.

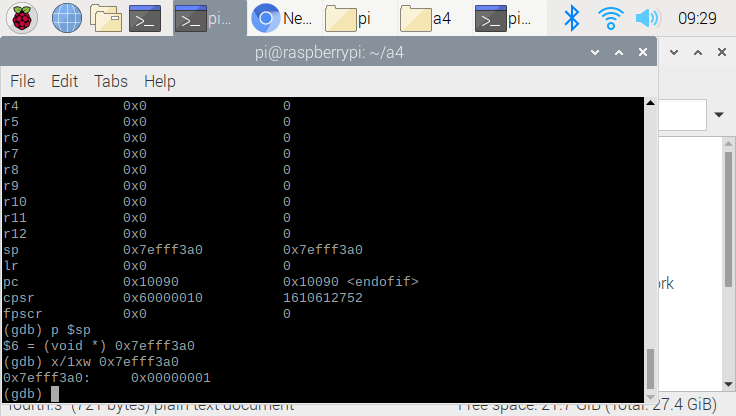
Part2

Fourth.s part2 code

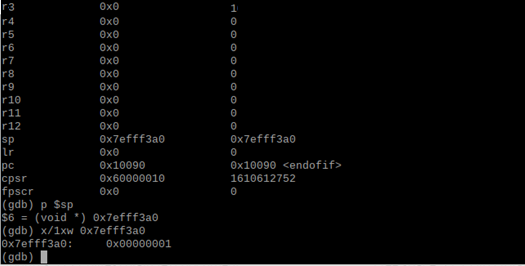


Even though part 1 code does proper job, we know it Is not efficient. What we have done here is negating the logical algorithm to make the program more efficient. What we have done is changing beq to bne. Then remove the b instruction. Above photo is a code after modification.

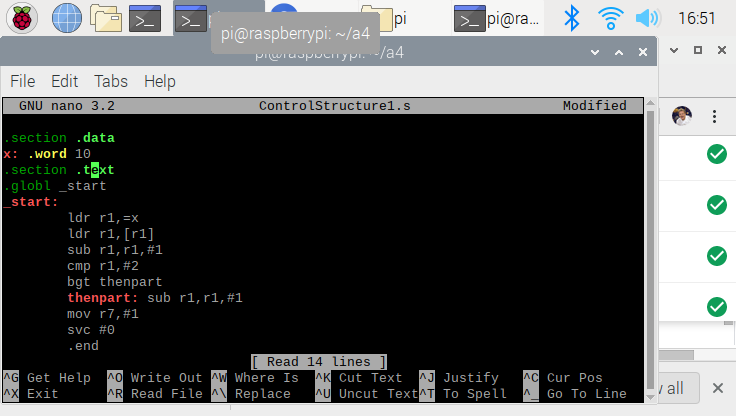
Fourth.s part 2 eflag info



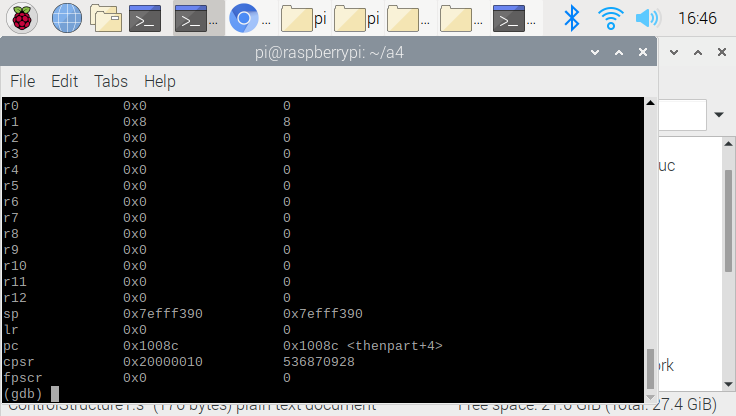
I have used p$sp to define the “eflag register” location. In order to examine the Zflag register, I must have defined the address of the Zflag register. From the photo above, the Zflag register address is 0x7efff3a0. Since we retrieved the address of Zflag, I have used command that examine the value for certain address. At the end, we defined that the Zflag register has value of 1, which we know that the Zflag is being set.

*Once the zero flag(x =0) being set, the r3 become 1*

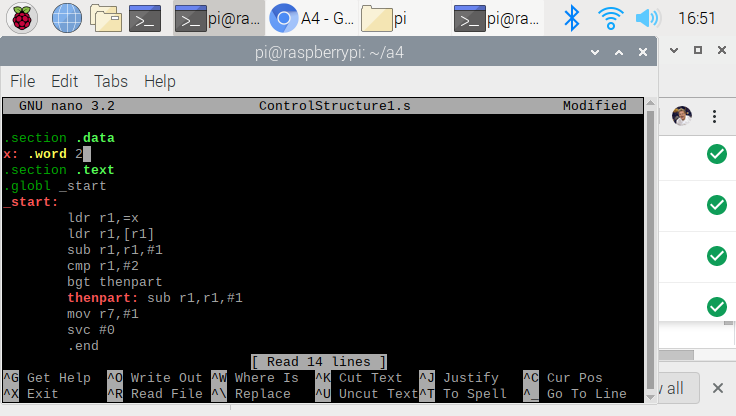
Part3



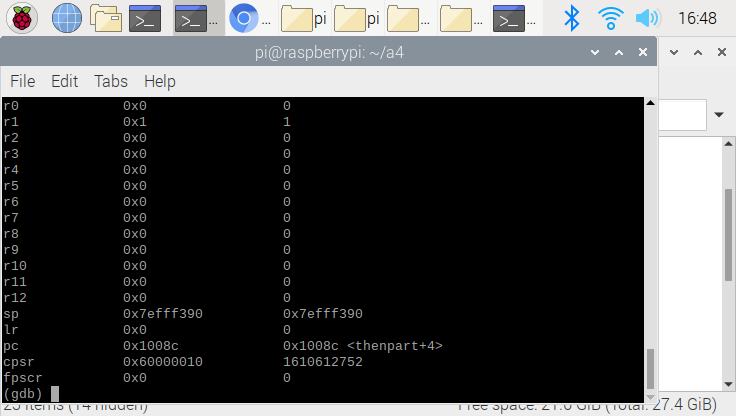
This program is doing a if else statement. This is do “if x is less than or equal to 3, then subtract 1, else subtract 2.”. I have assigned x is equal to 10 for this screenshot. Since 10 is greater than 3, the result for x must be 8.



This is a register information for ControlStructure. Register 1(r1) is assigned to be variable x, and is 8. The original x was 10 and 2 is being subtracted. We know that the code is correct since the 2 must be subtracted from the original. Since 10 -2 = 8 == 8, the program work correctly.



This is exactly same code but with different x value. Now, x become 2. Since 2 is less than equal to 3, the only 1 must be subtracted. From the code above, I have negated the equation form “less than equal” to “greater than”. What does it do is that if the number is greater than 2, then it will jump the statement.



The original value for x was 2. Since it is less than 3, only 1 must be subtracted from the original. We know our program is correct since only 1 was being subtracted.