**CSE 3038 Computer Organization**

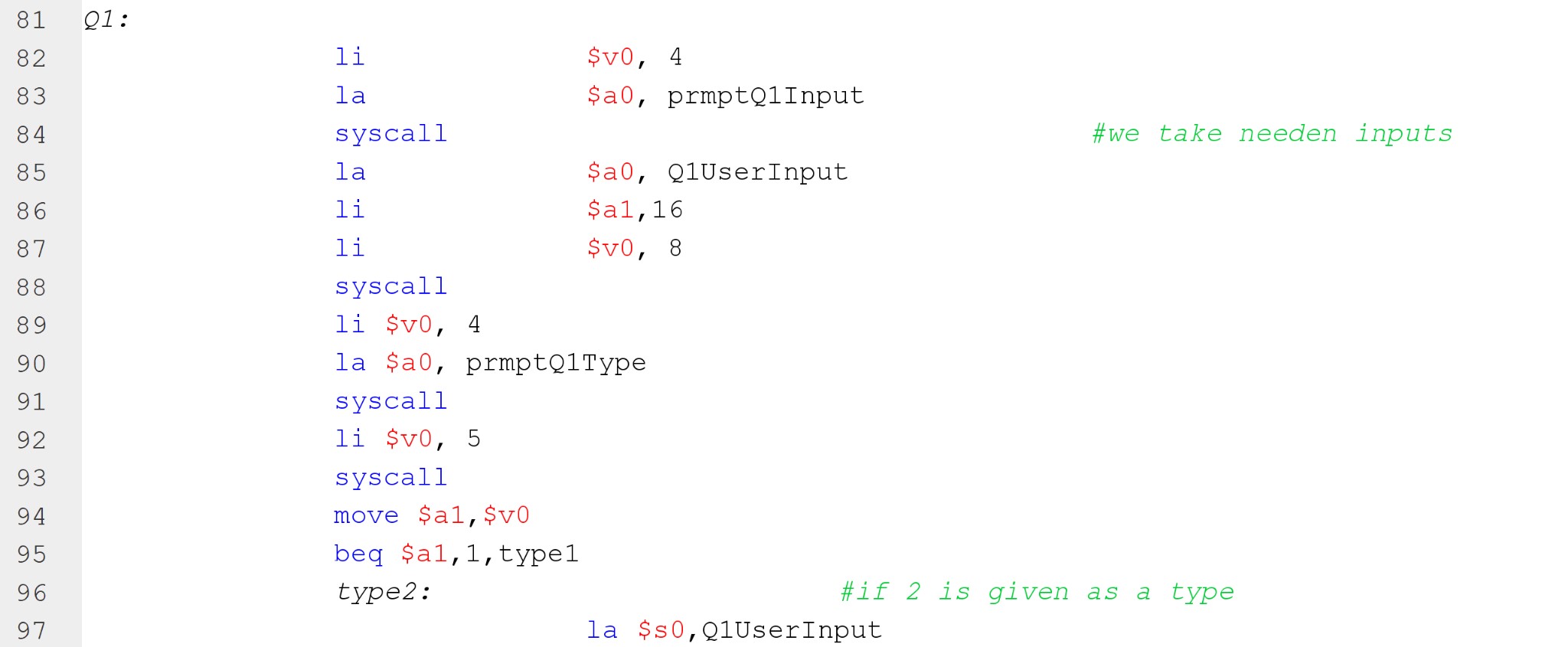
**Project 1 Report**

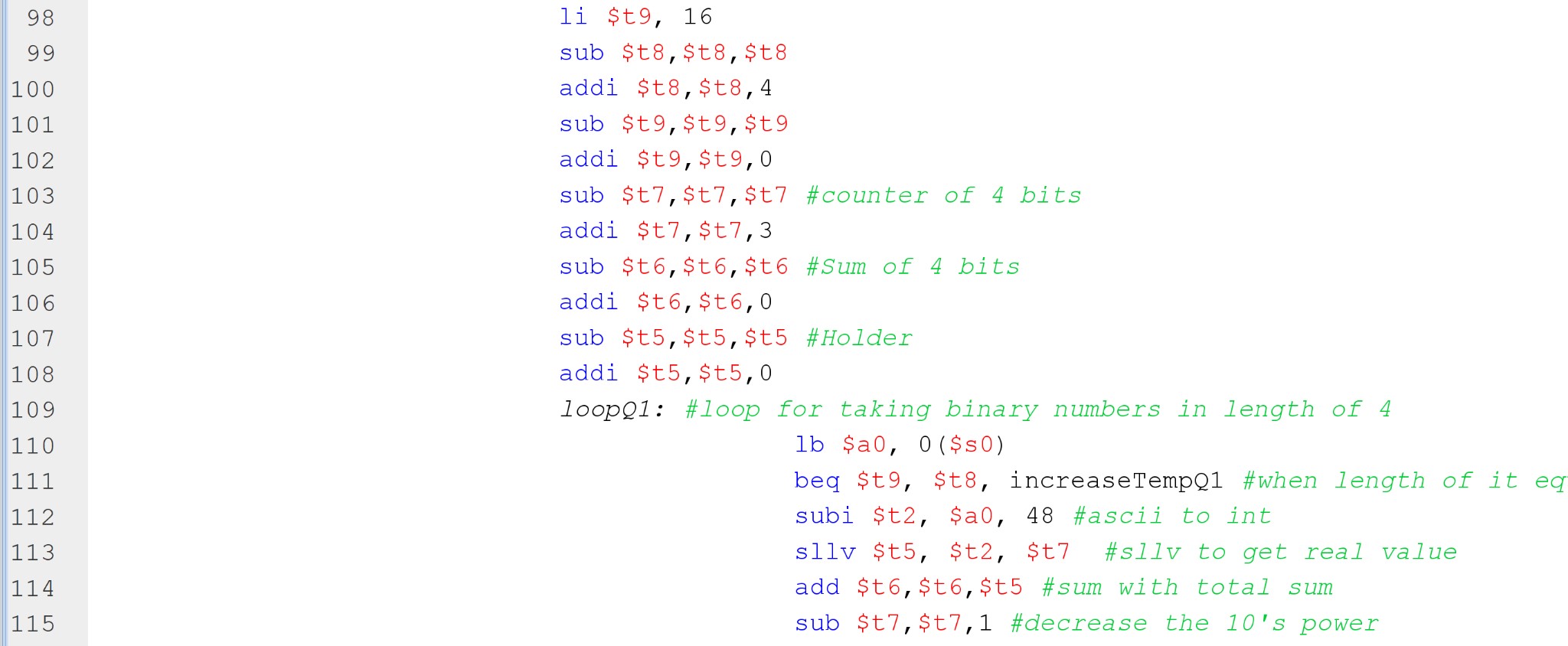
# Q1:Base Converter

After taking input and type we make calculations by type. If type equals to 1 we check the first bit to check whether the given bit is negative or positive. Therefore calculate by considering being negative or positive. After checking the first bit, we iterate through into the other bits by checking whether they are equal 0 or 1. If they are zero we do not do any calculation and if they are 1 take their index and make the power of 2 calculation then as we are iterating through them hold addition of the calculated power of 2s. If type is equal to 2, we take input in length of 4 bit and if the first bit is 1, we add 8, for the second bit we add 4, for the third bit we add 2 and for the last bit we add 1 to the total calculation of the taken 4. After calculating summation we are printing the result according to its ascii values. And it runs until it hits the end of the string.

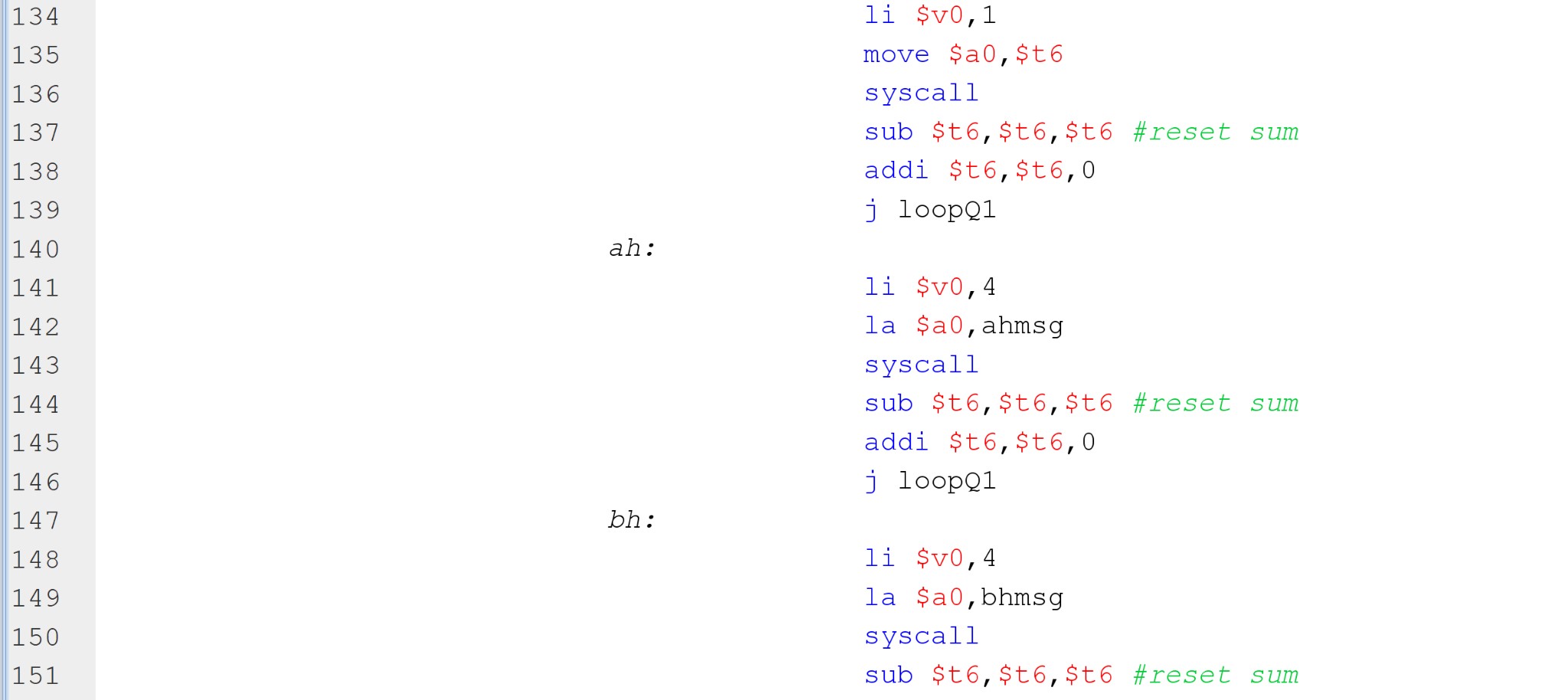
Screenshots:

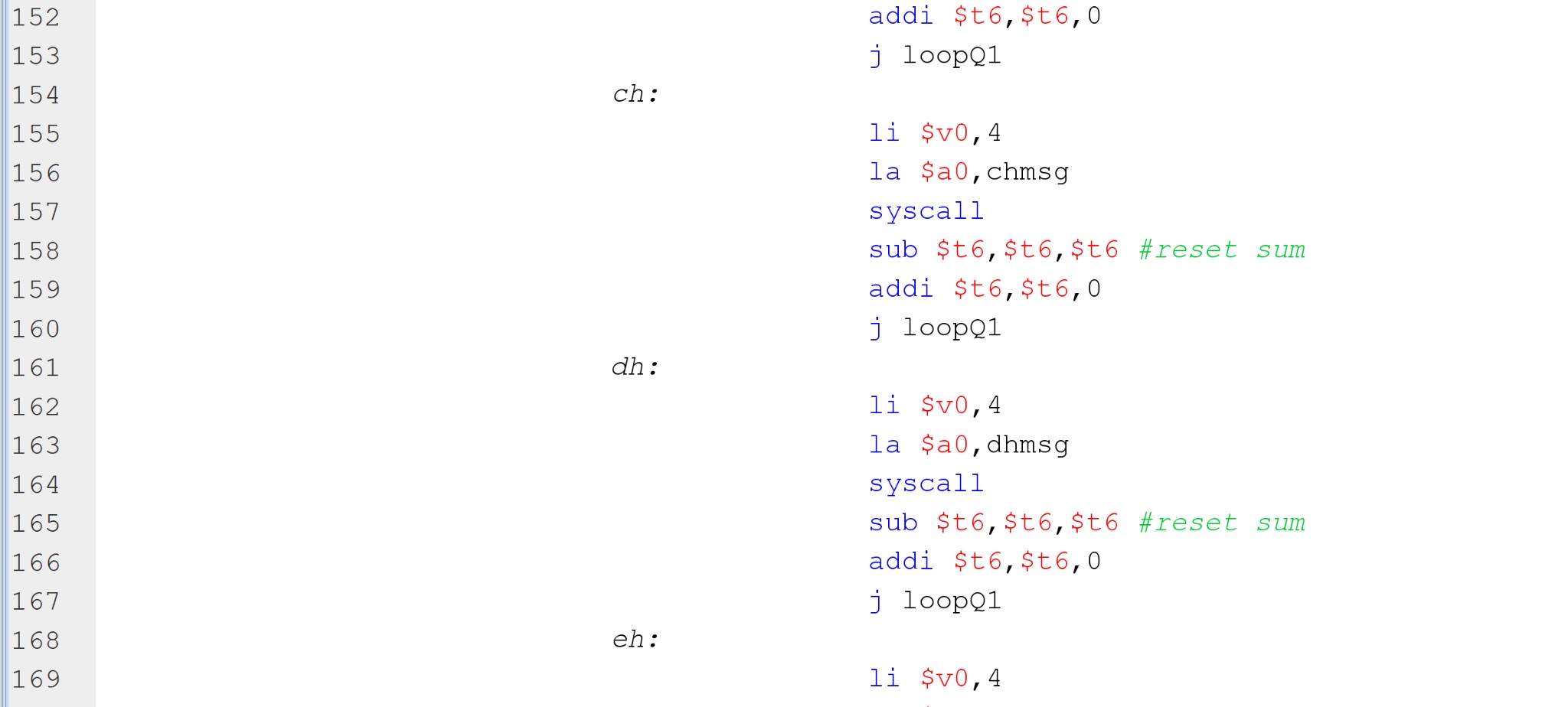
Code:

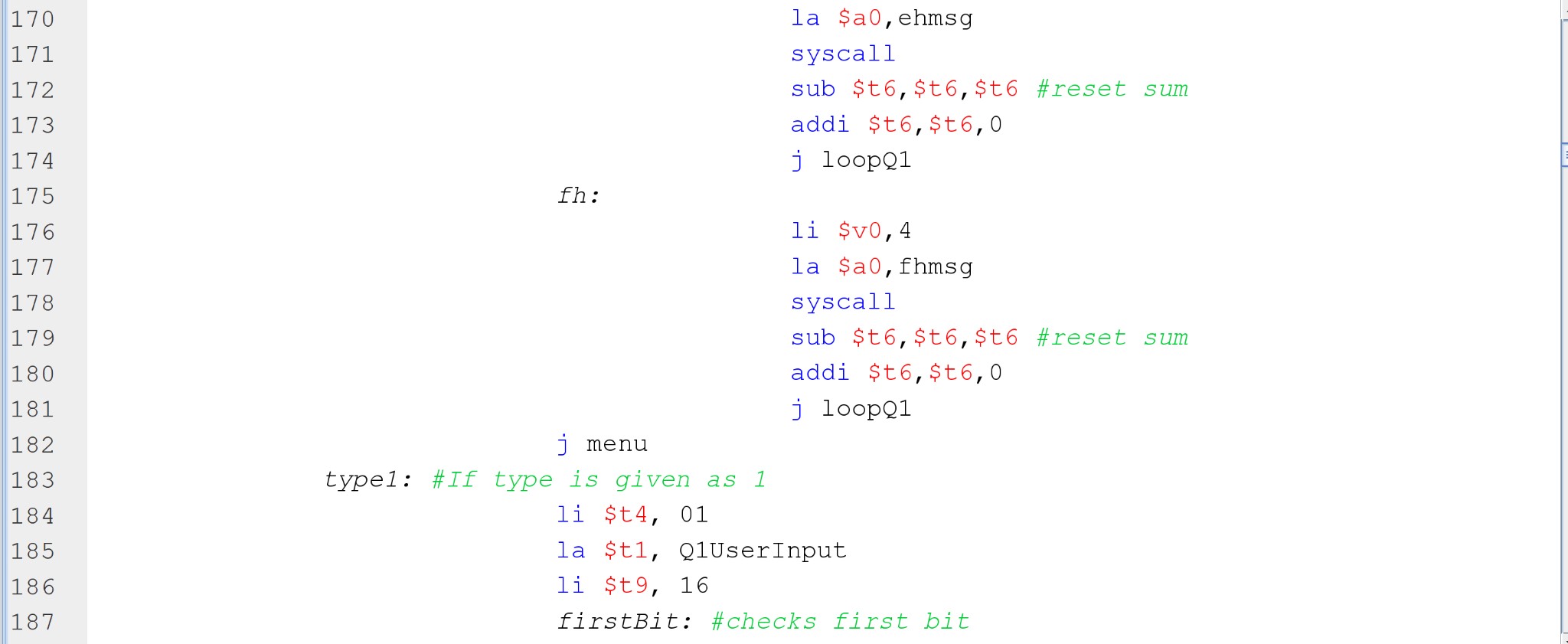


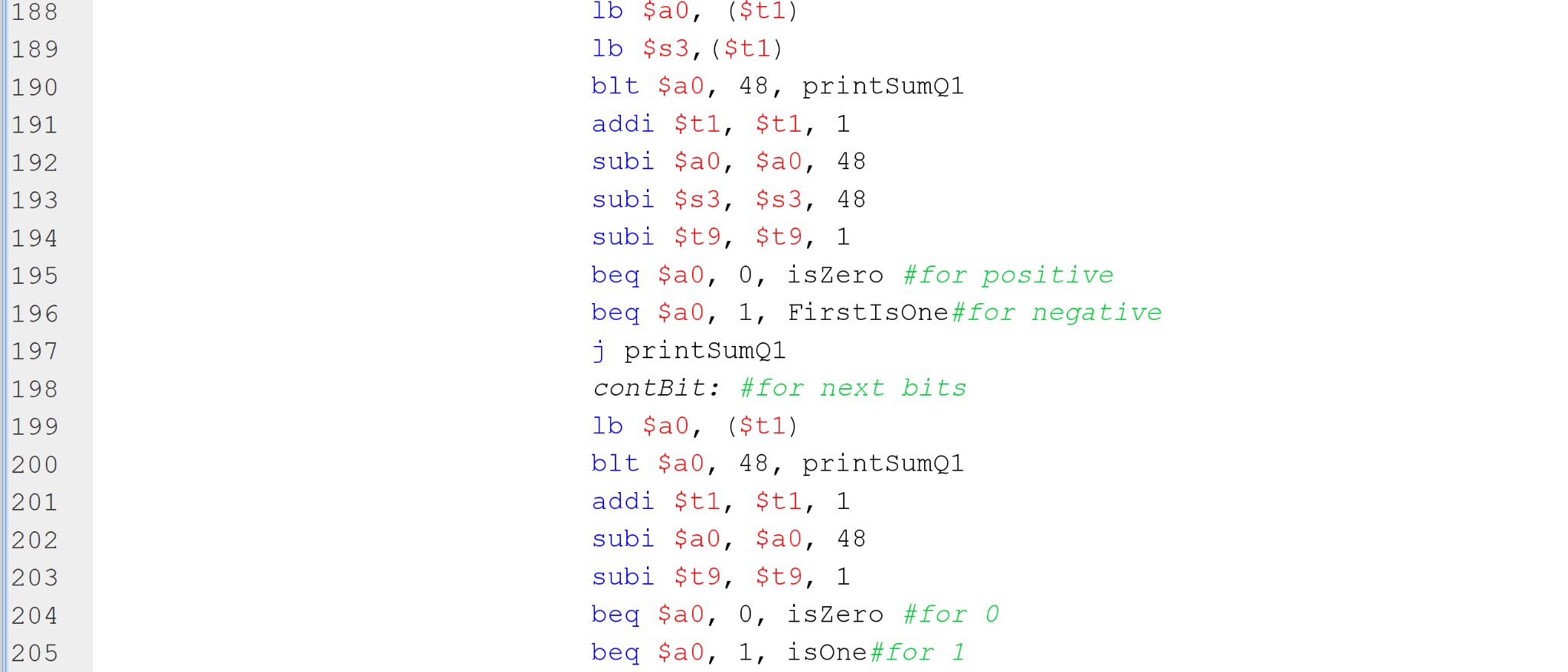


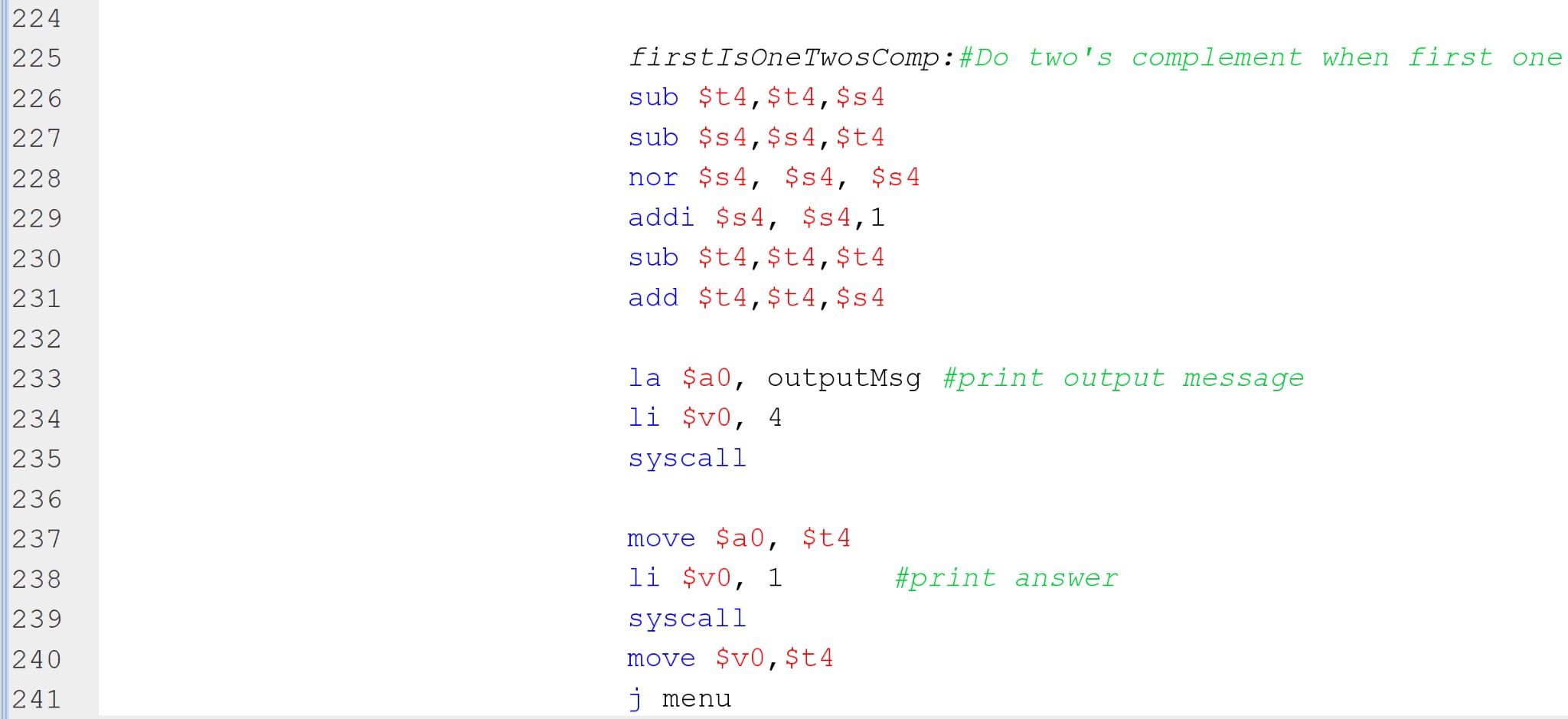


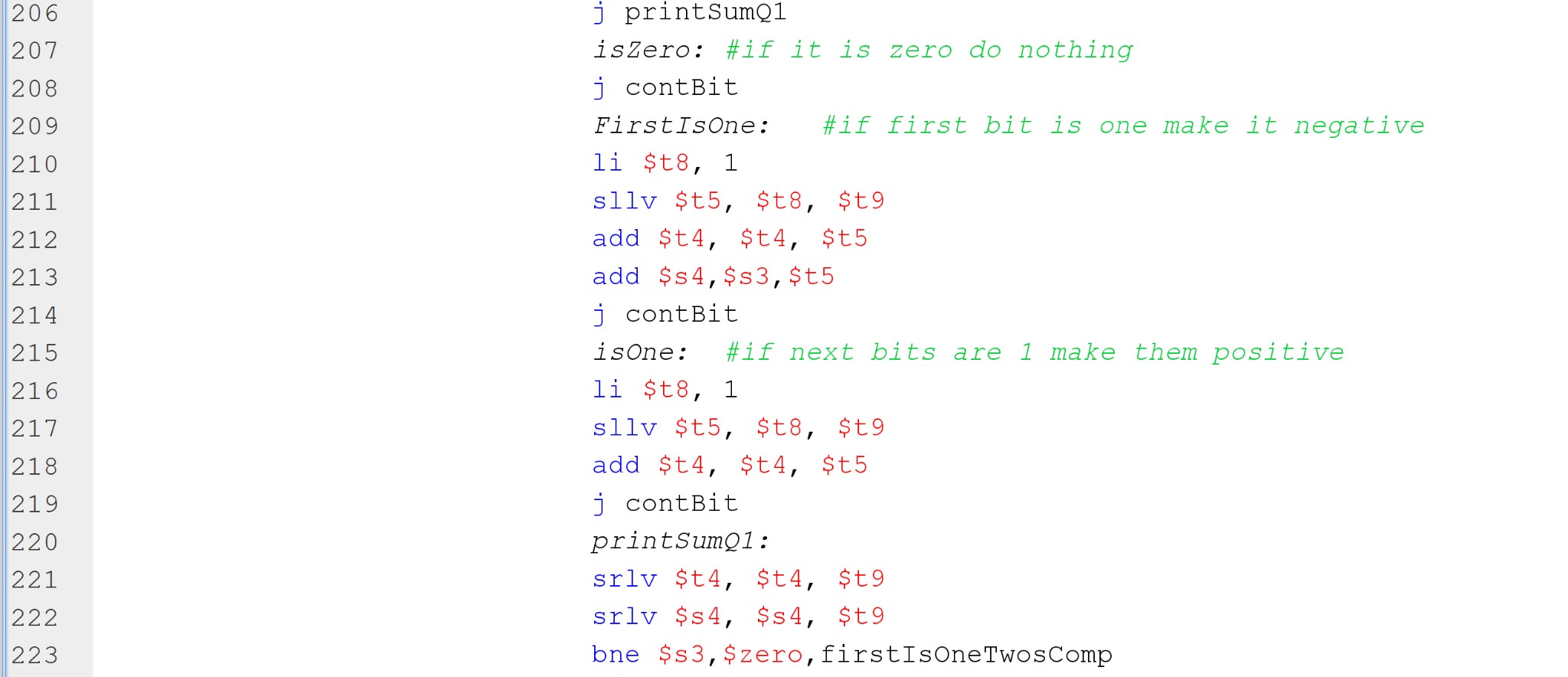






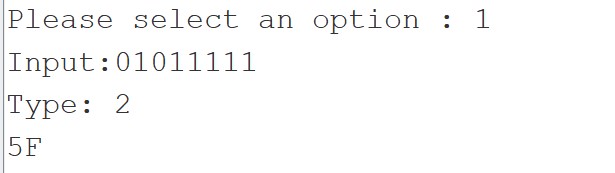




Run:

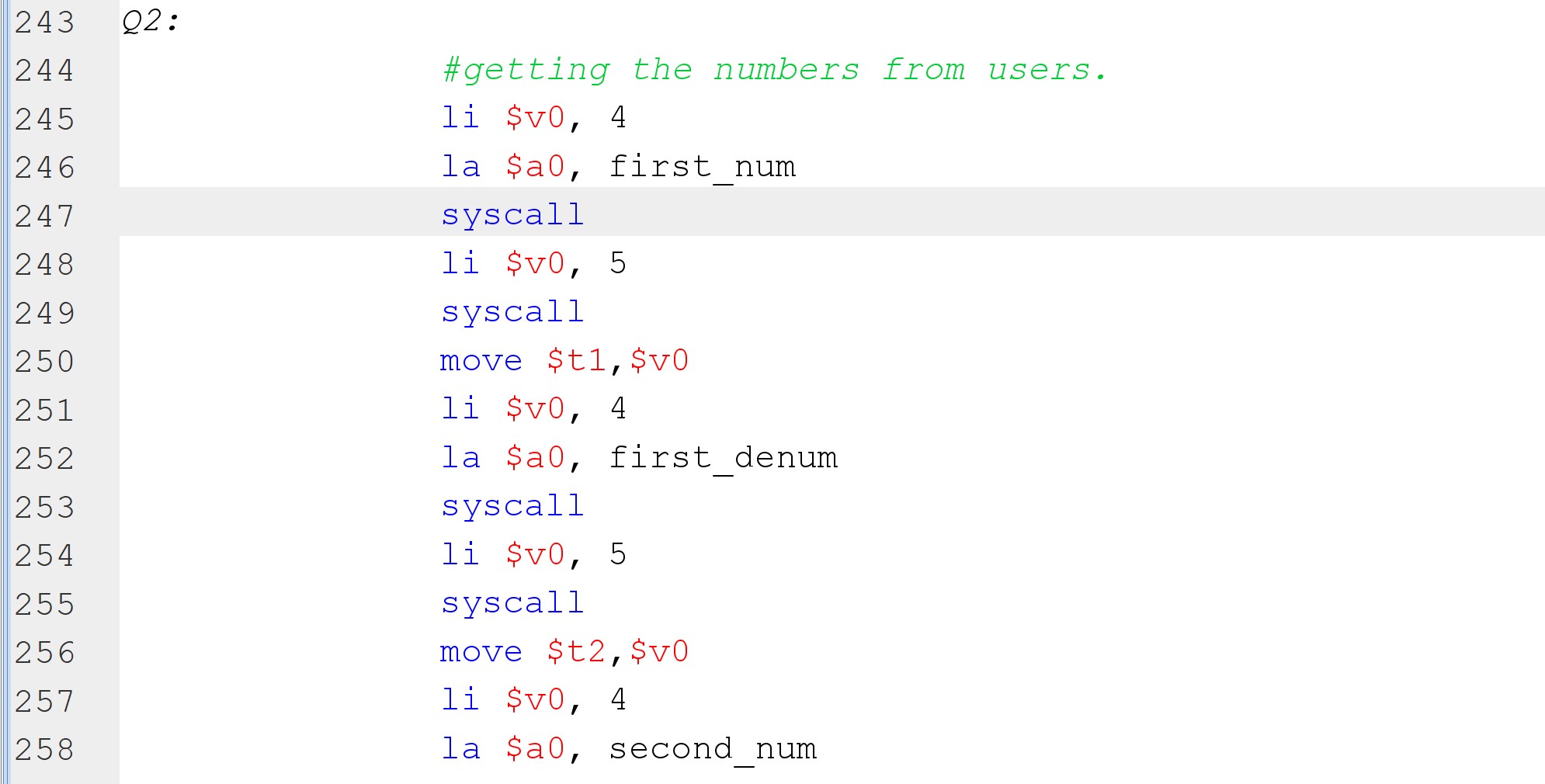


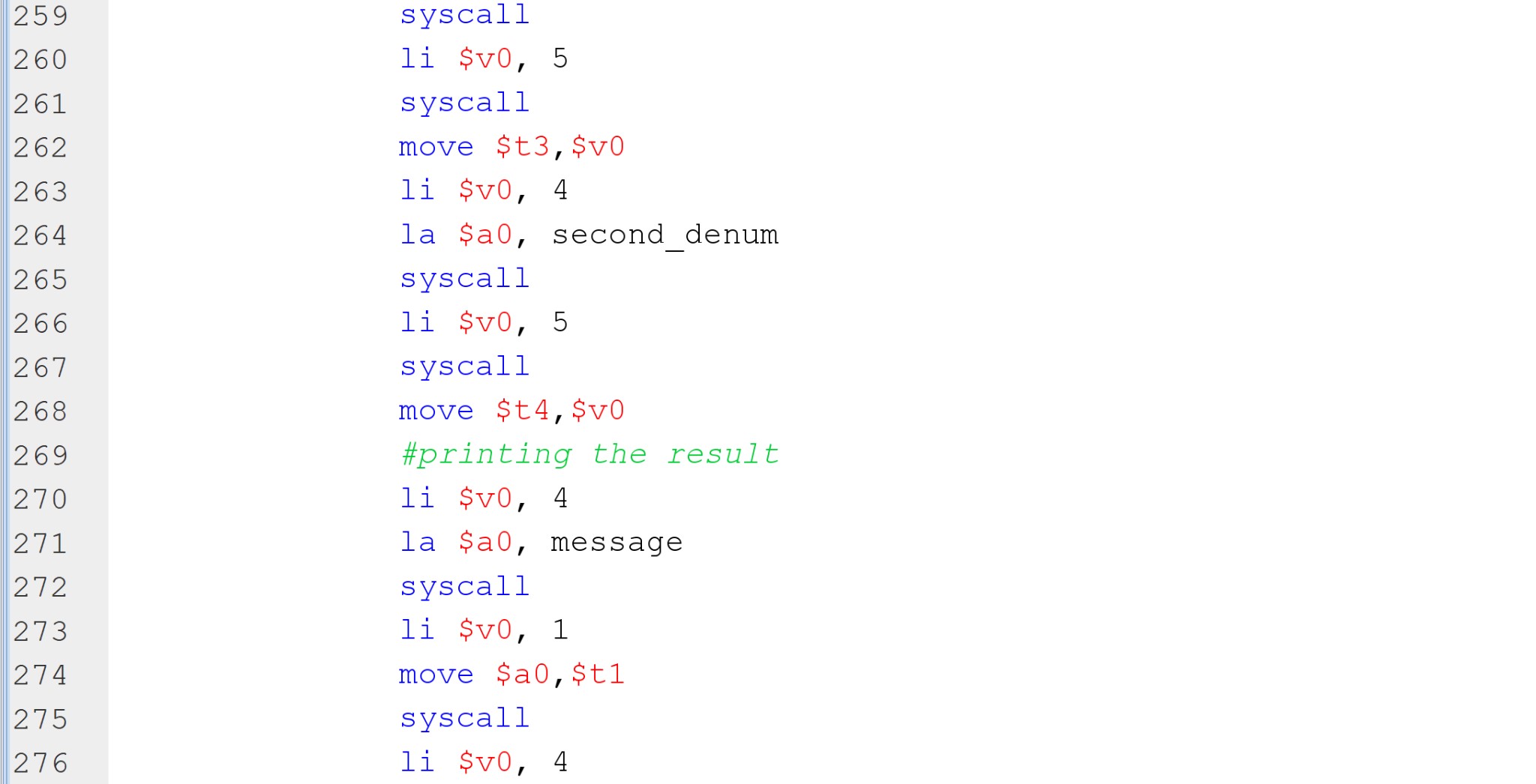
## Q2. Printing the sum of two rational numbers as a rational number

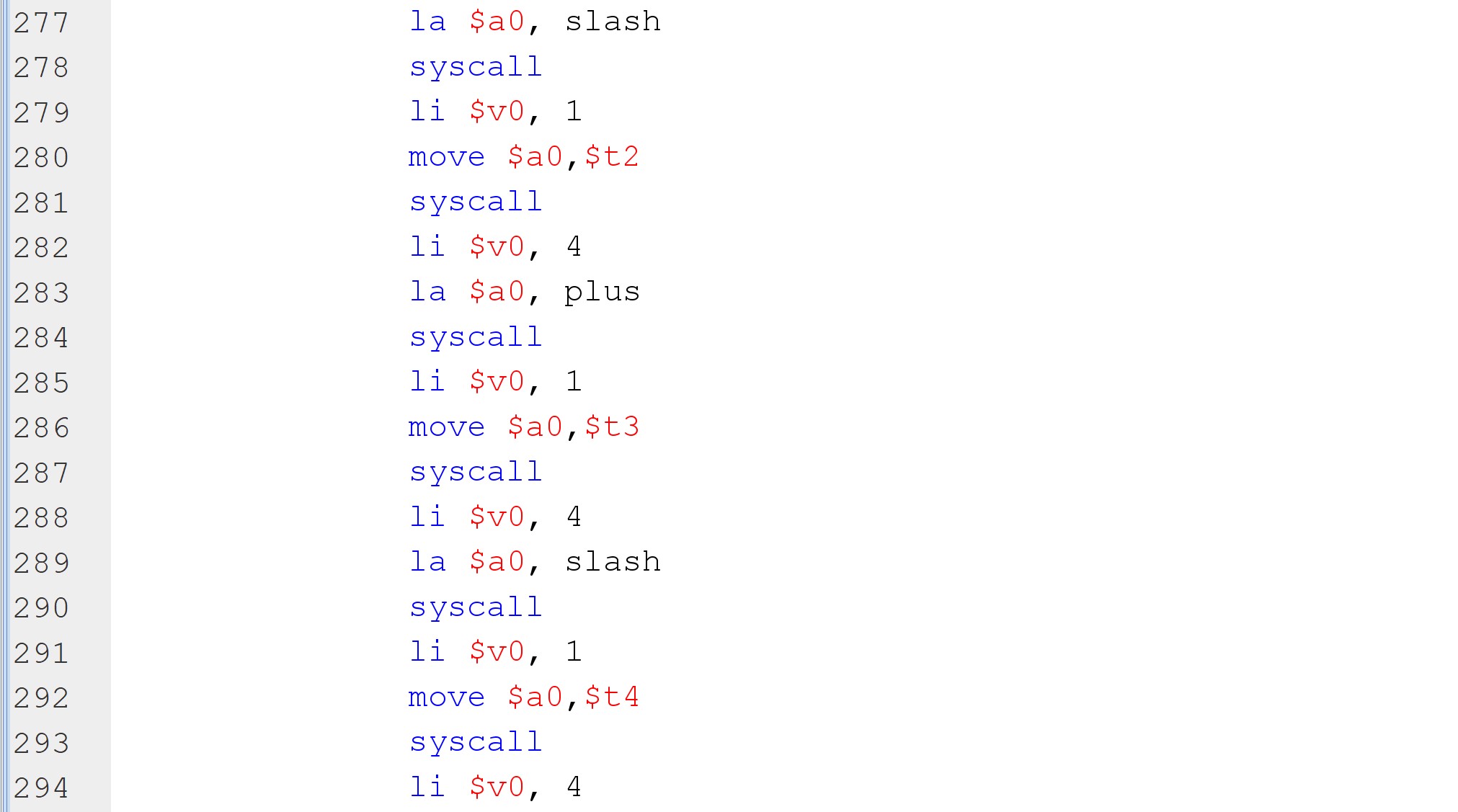
We wrote our code for this question on MARS. First, we got our numerator and denominator values from the user. After multiplying this data we received later in accordance with the formula thanks to the mult and add instructions, we found the answer to the process, but the result we found was not a simplified version. Then we printed our result by simplifying the beq, bge instructions using the euclid algorithm.

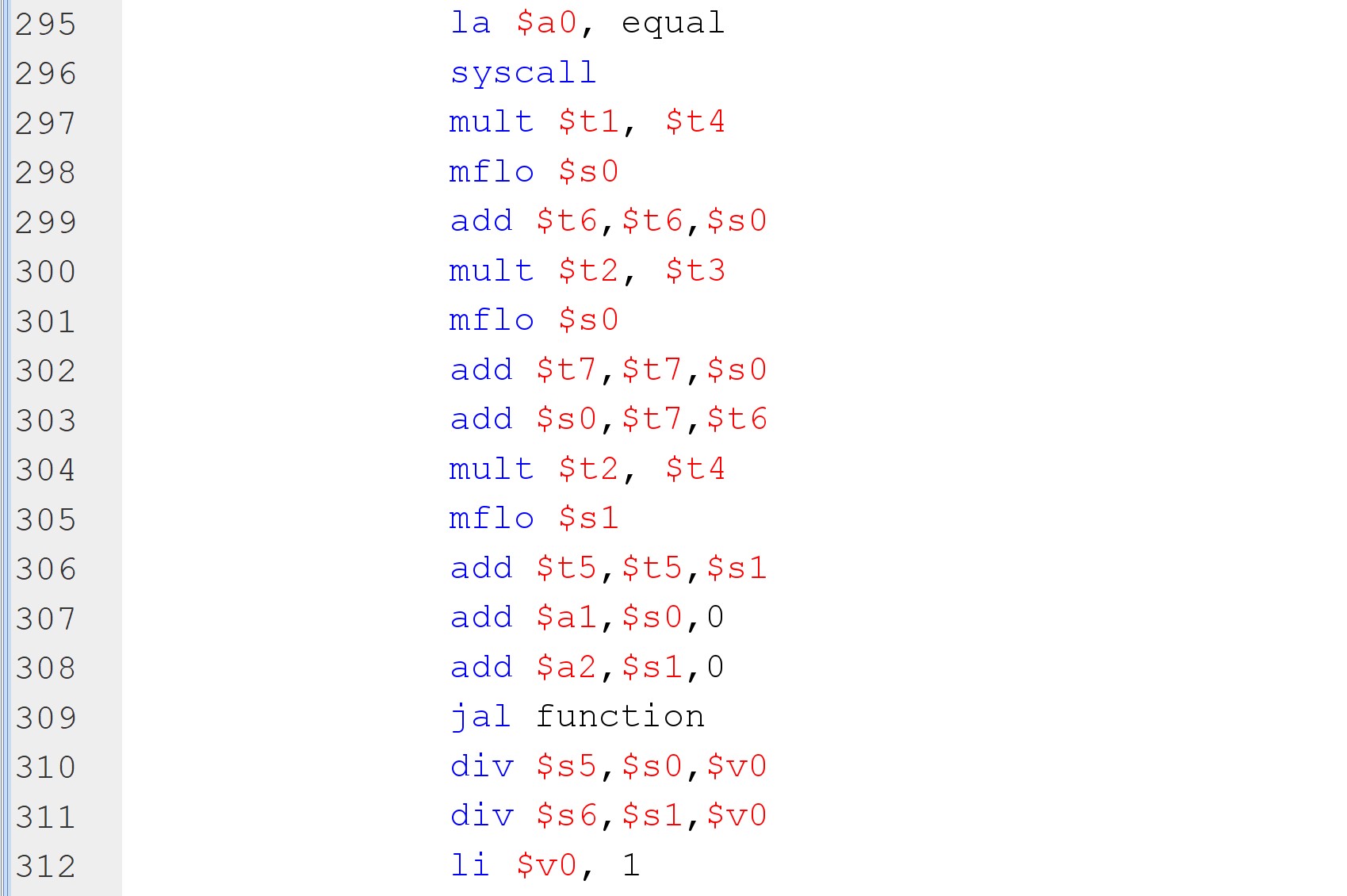
Screenshots:

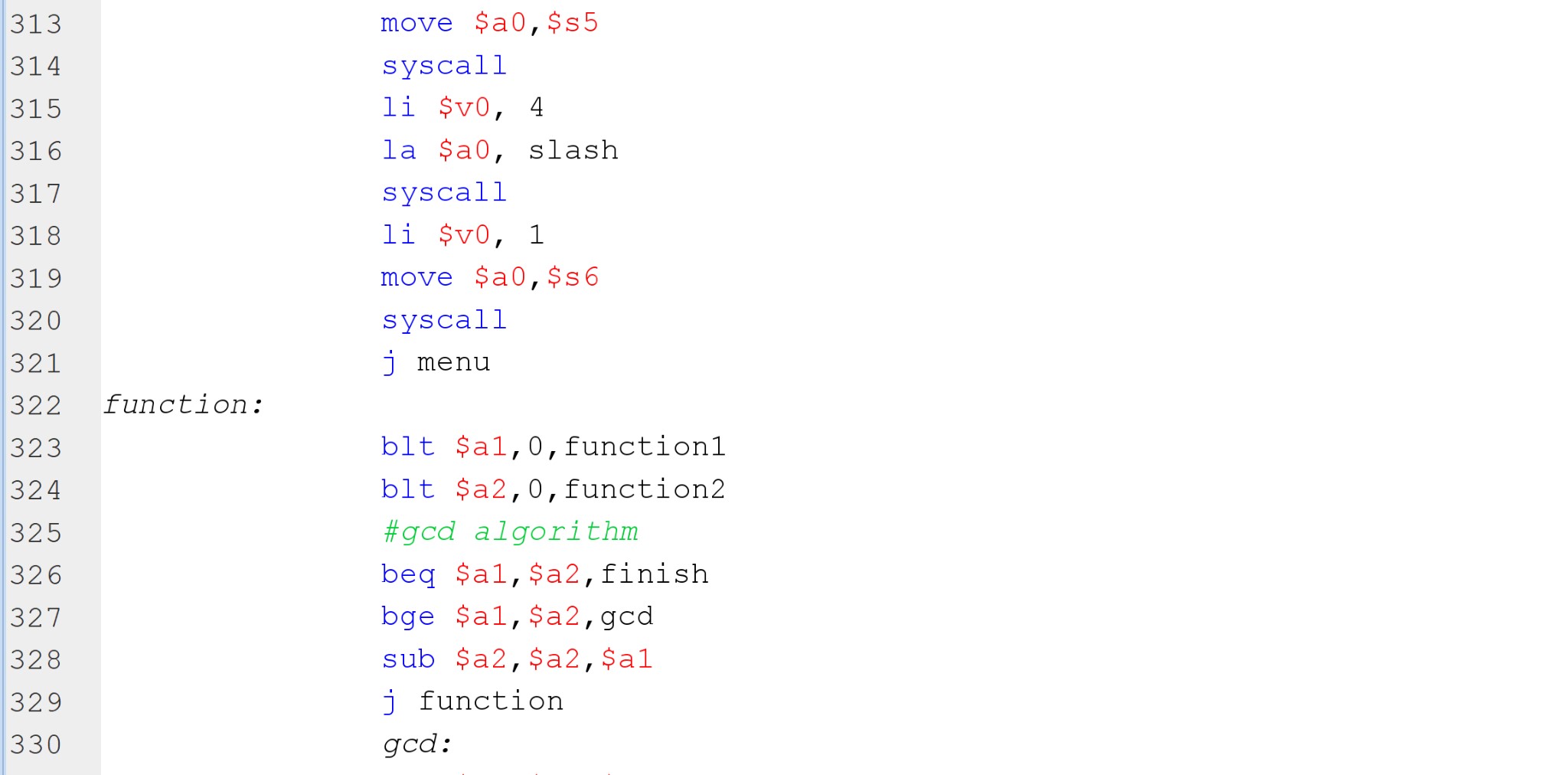
Code:

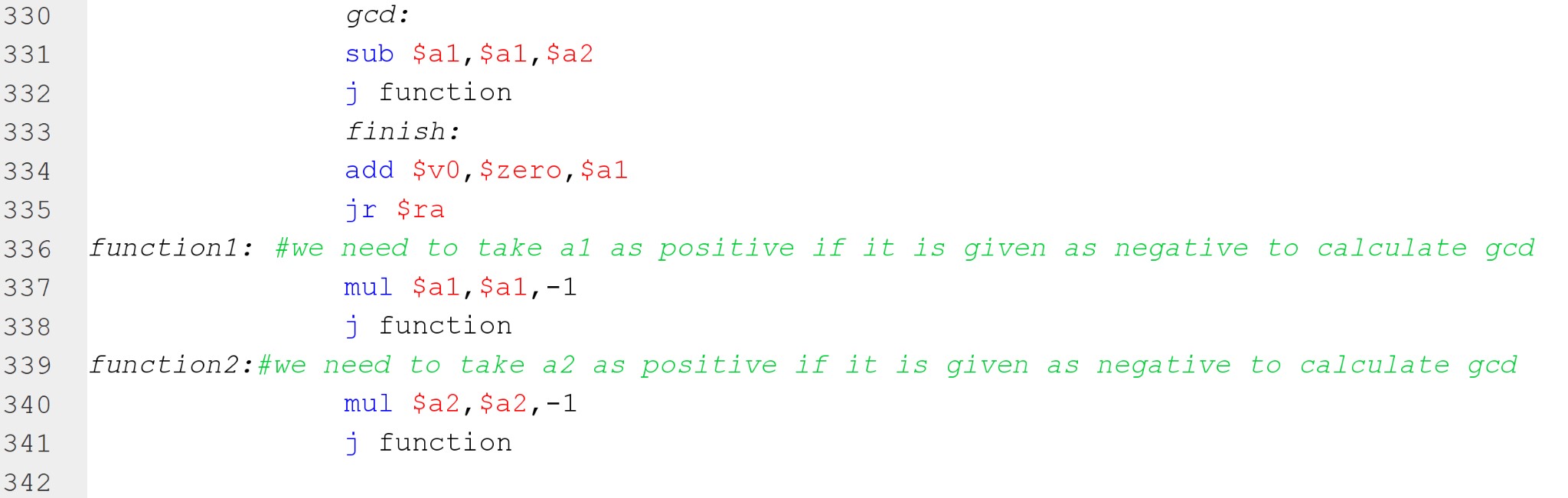




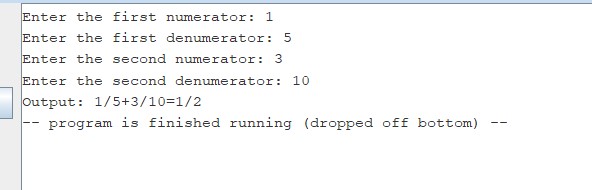








Run:

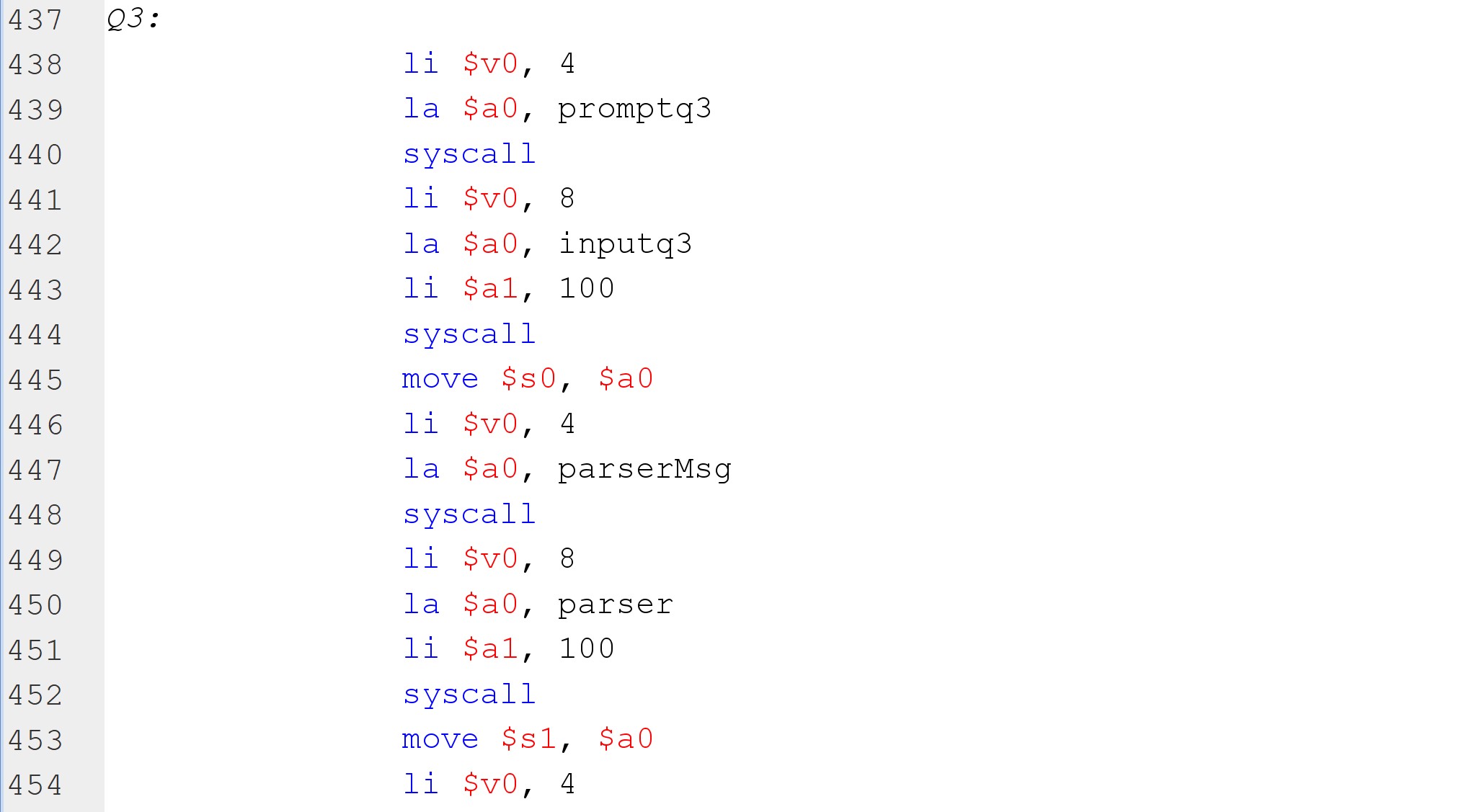


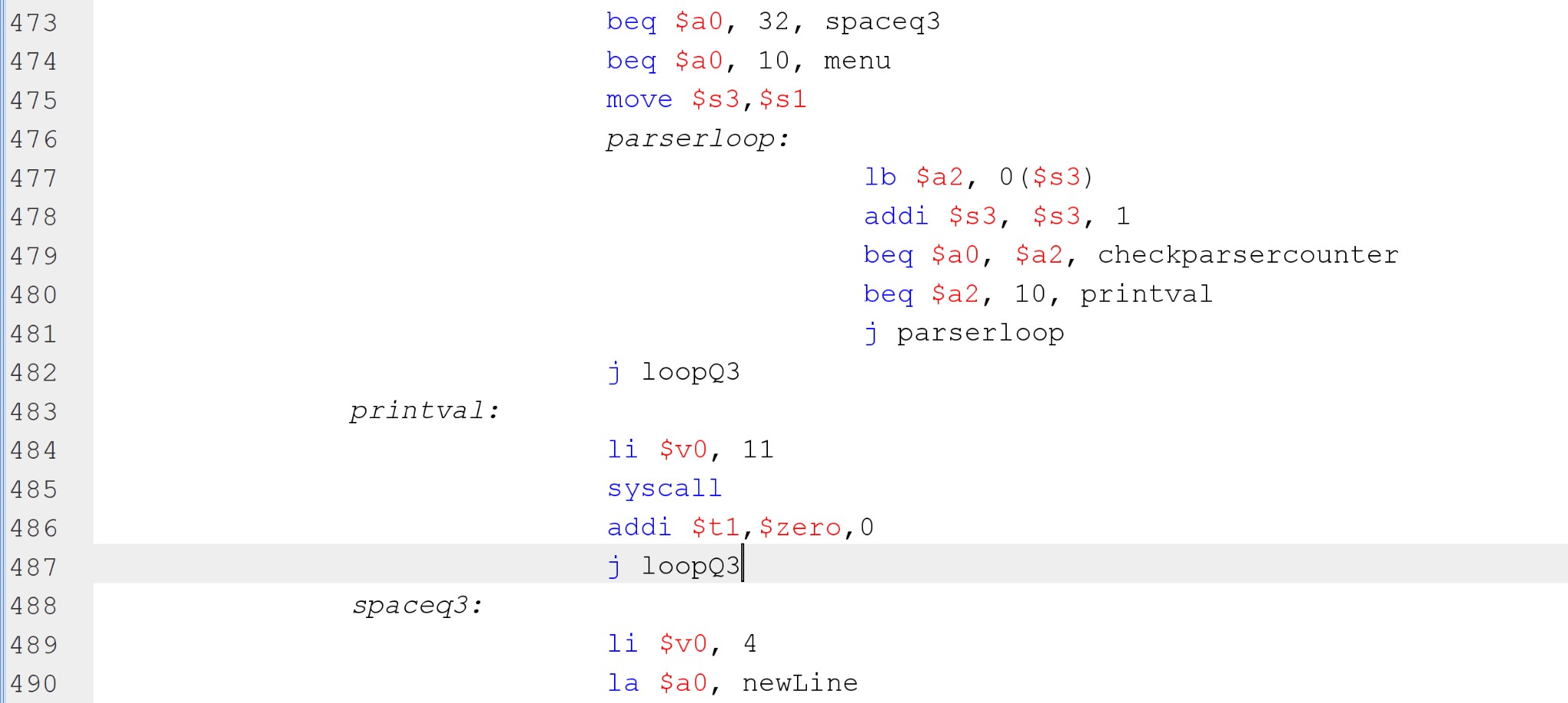
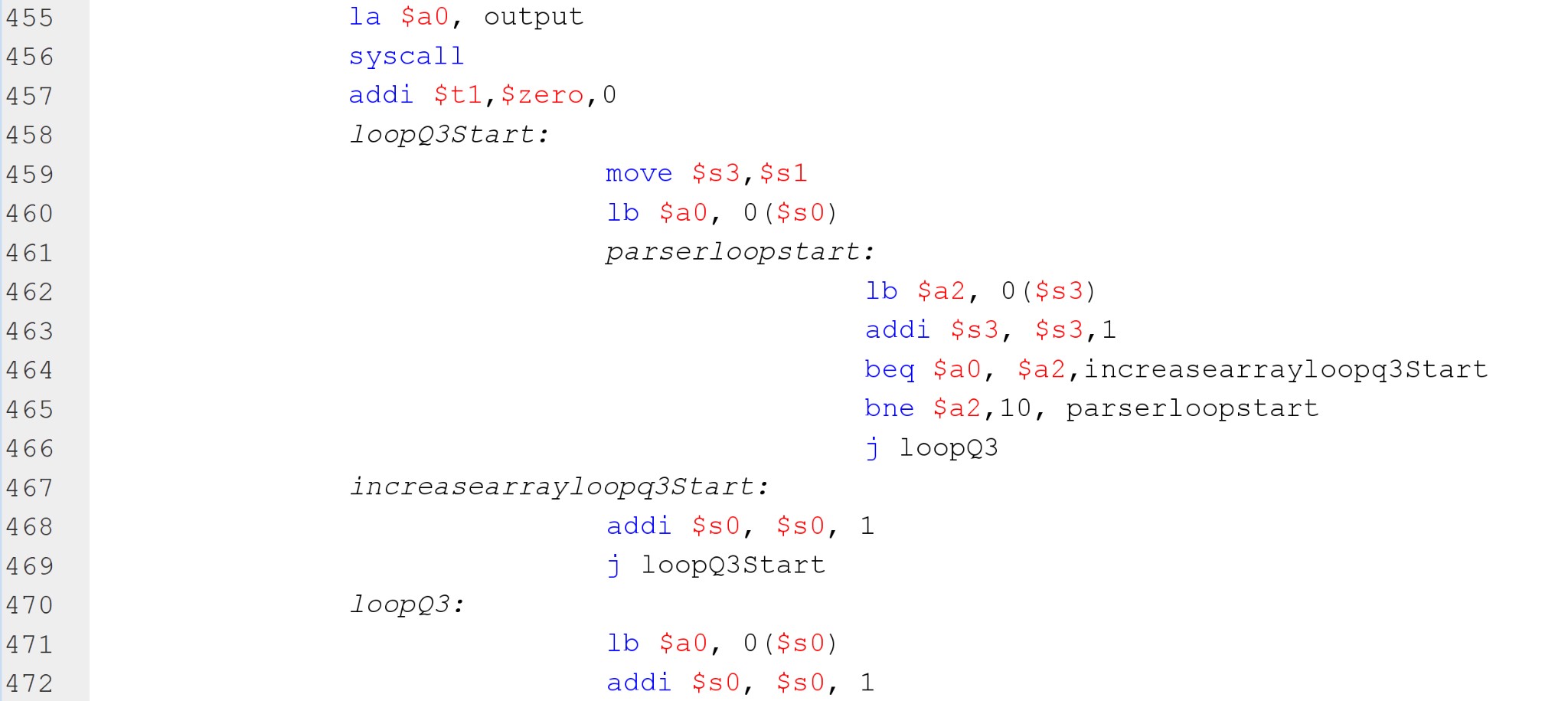
**Q3: Printing the given input sentences one after the other without printing the parser signs to the output.**

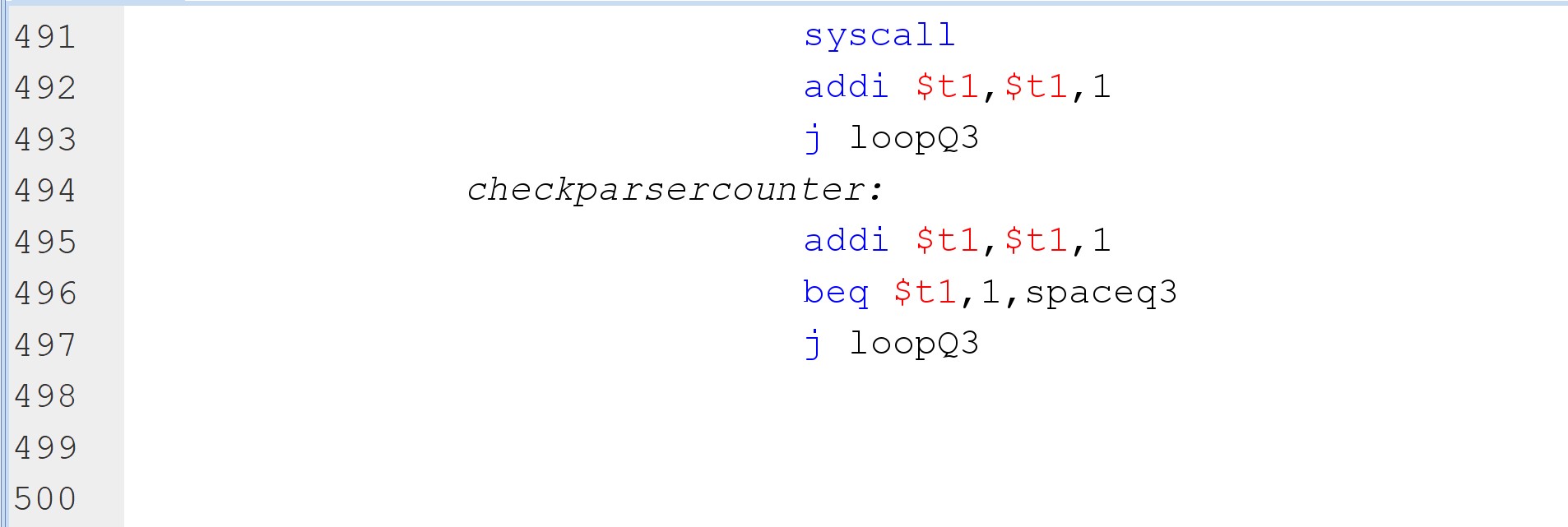
We write a MIPS procedure that takes input sentences and prints the words in it by parsing with a given set of characters. A word is a sequence of characters without any **whitespaces** and a given set of punctuation marks. The only punctuation marks that you have to consider will be given as second input. Size of the punctuation array can be at most 8 ASCII characters.

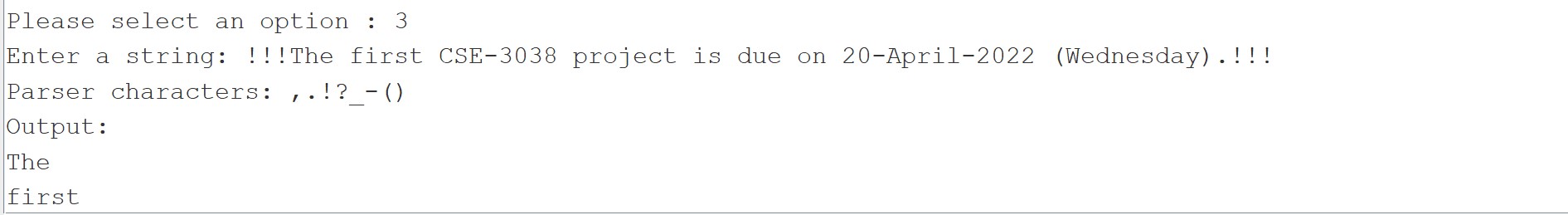
Screenshots:

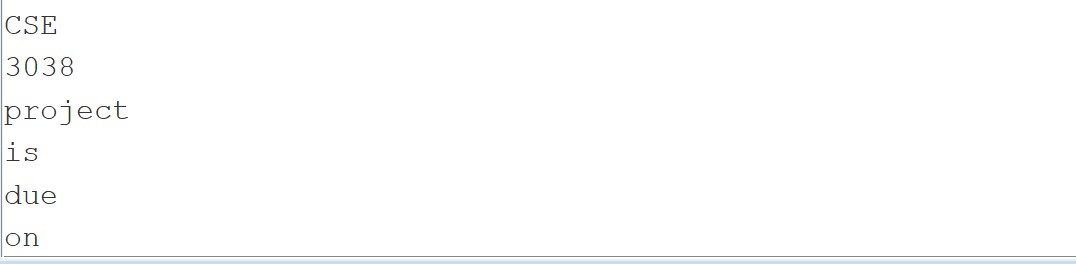
Code:







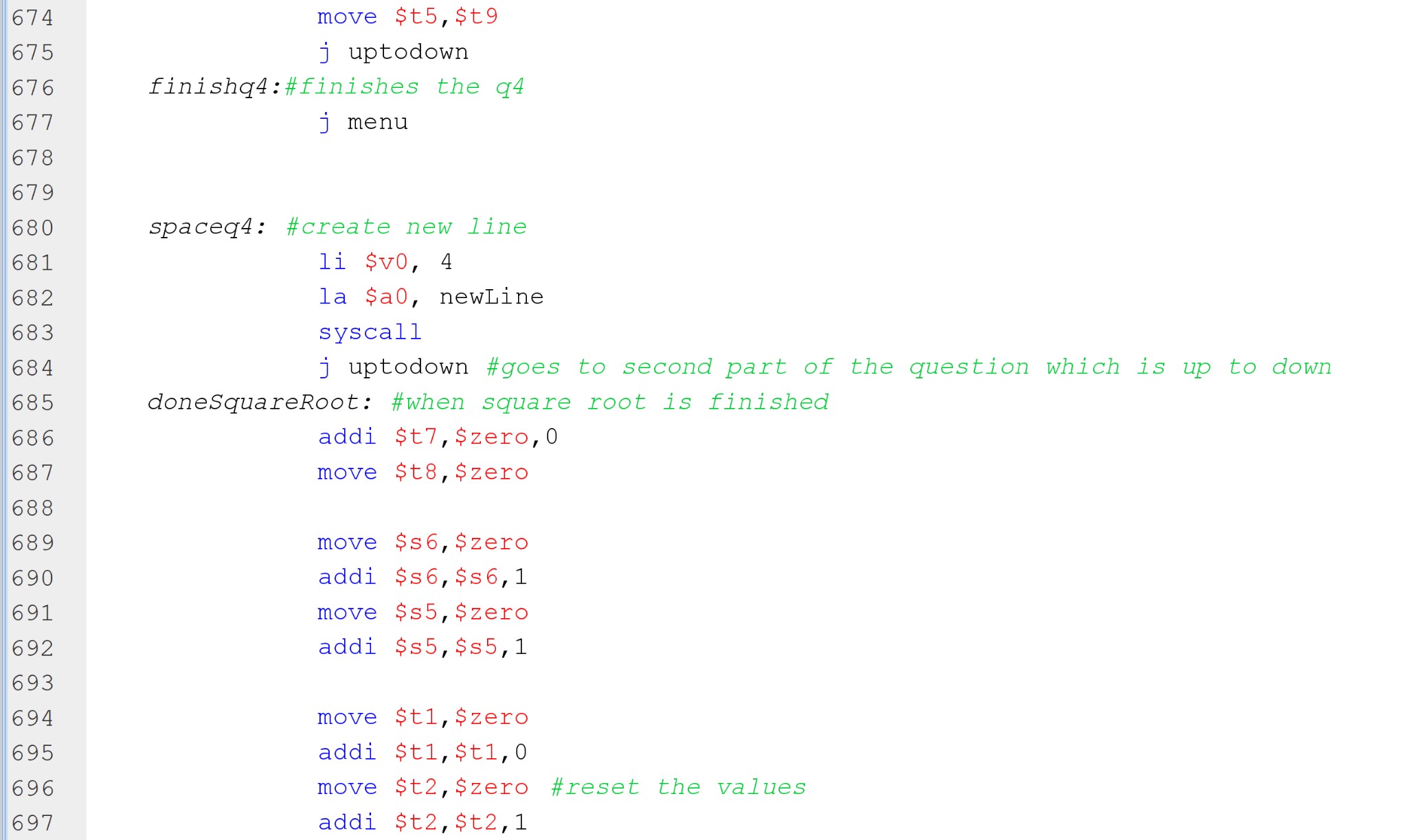
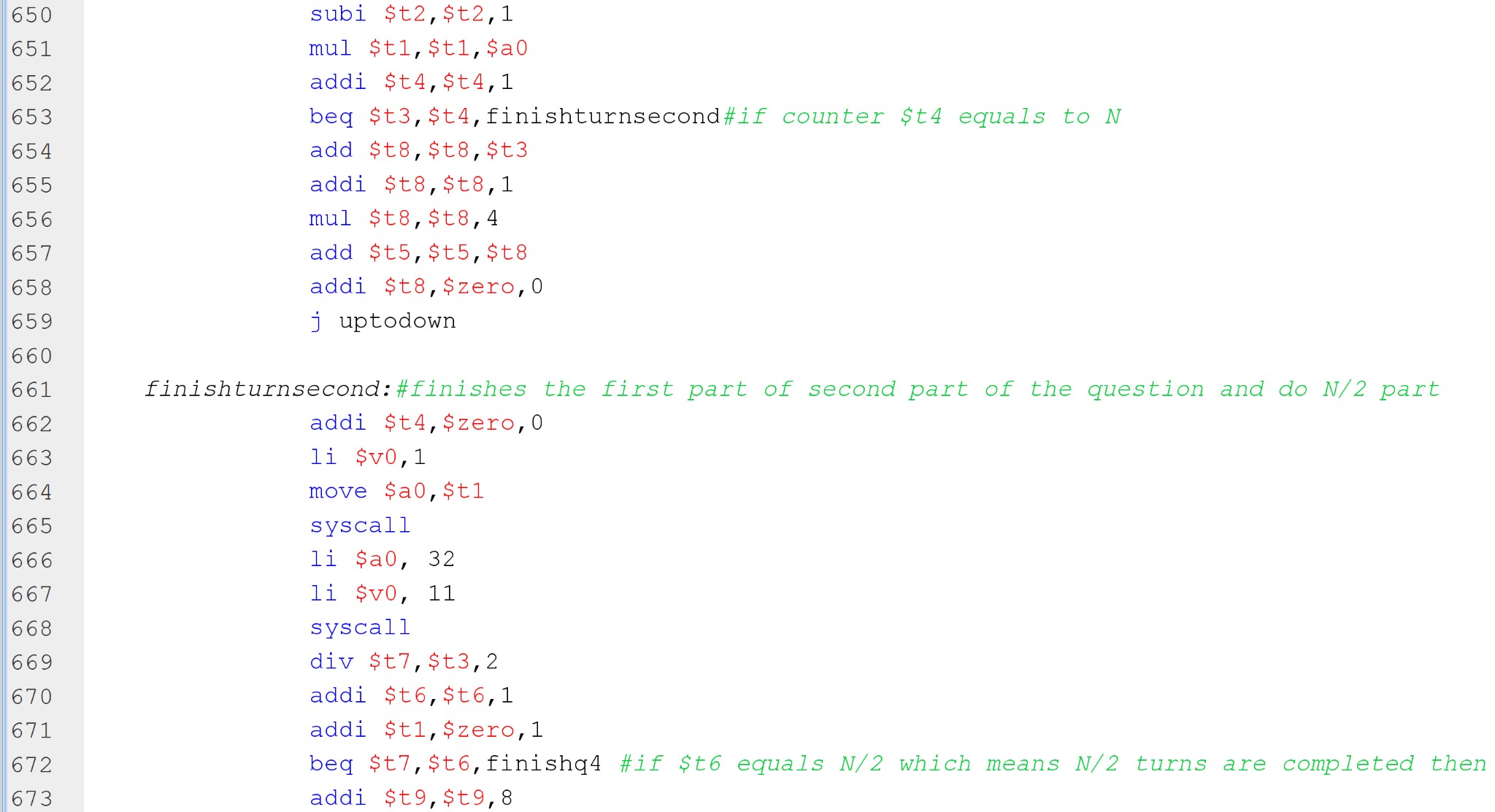
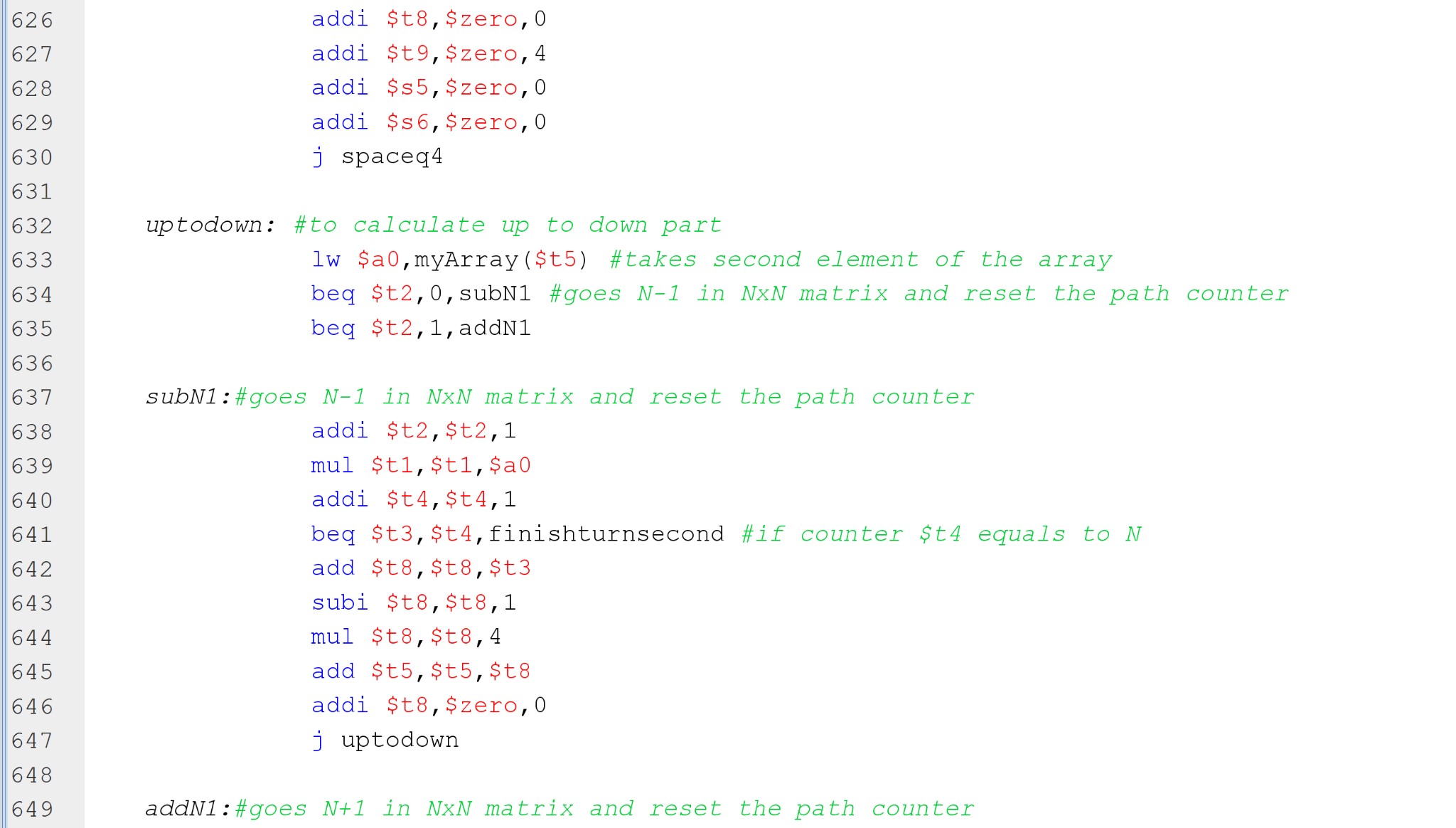
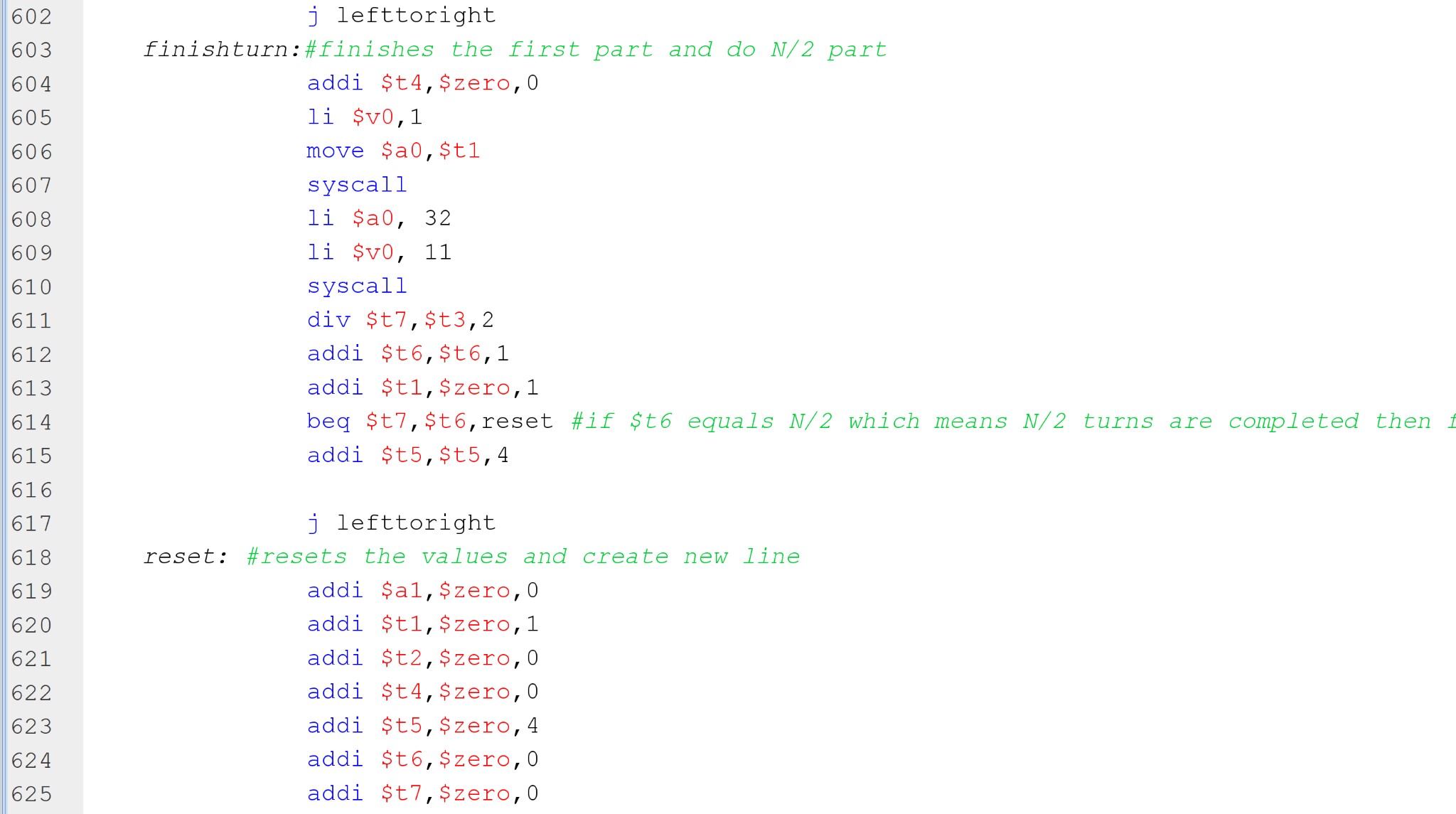
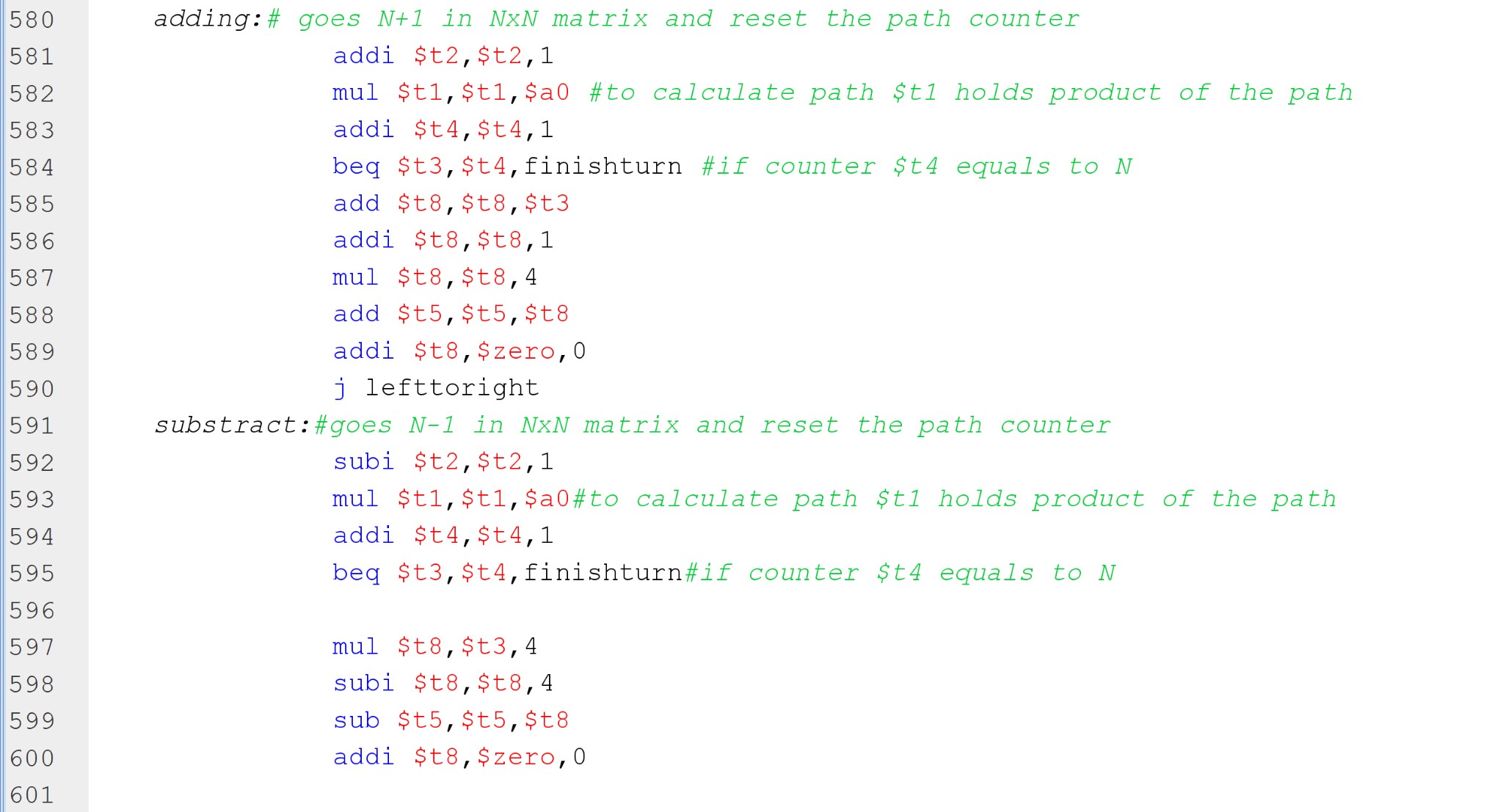
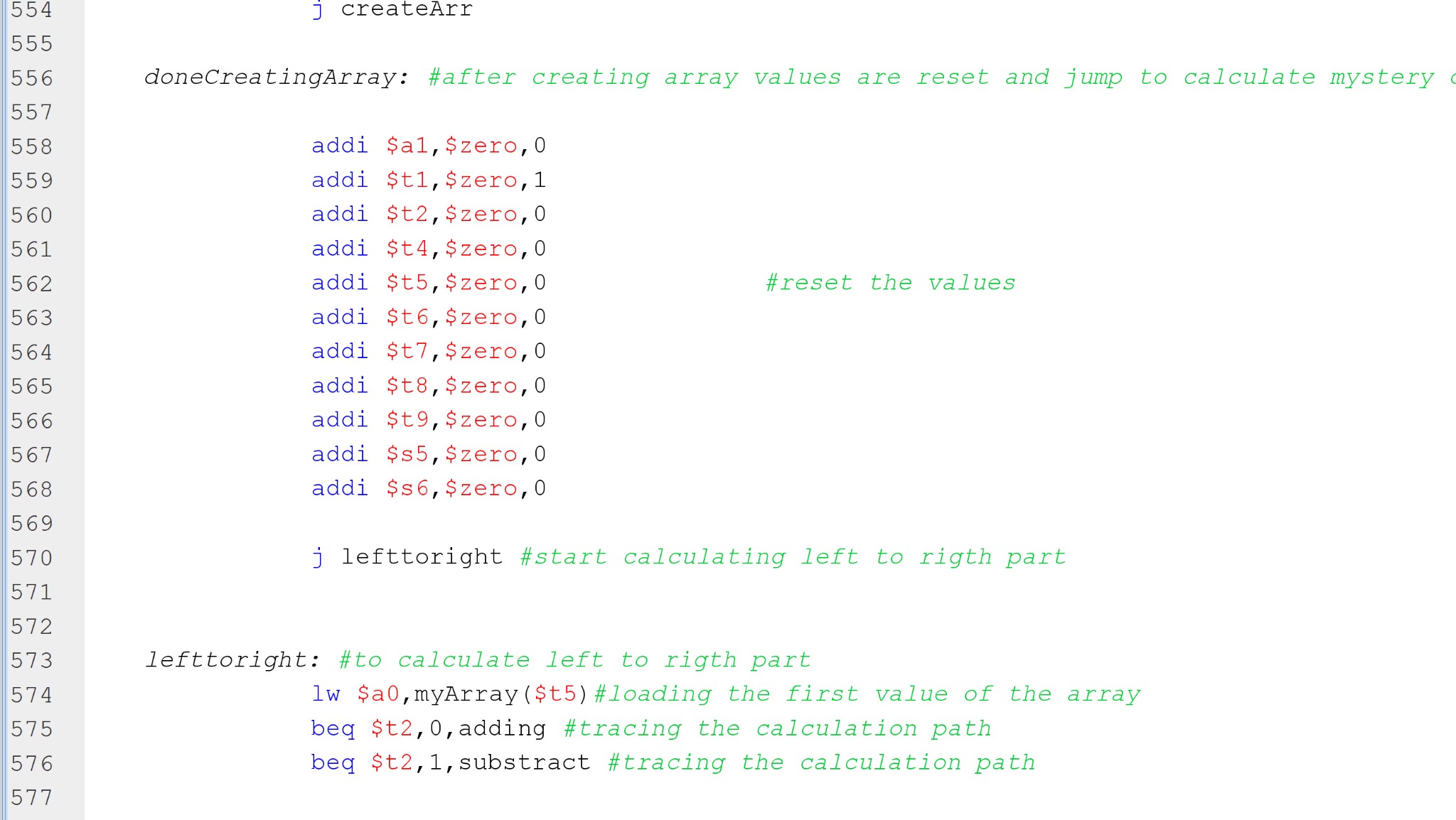
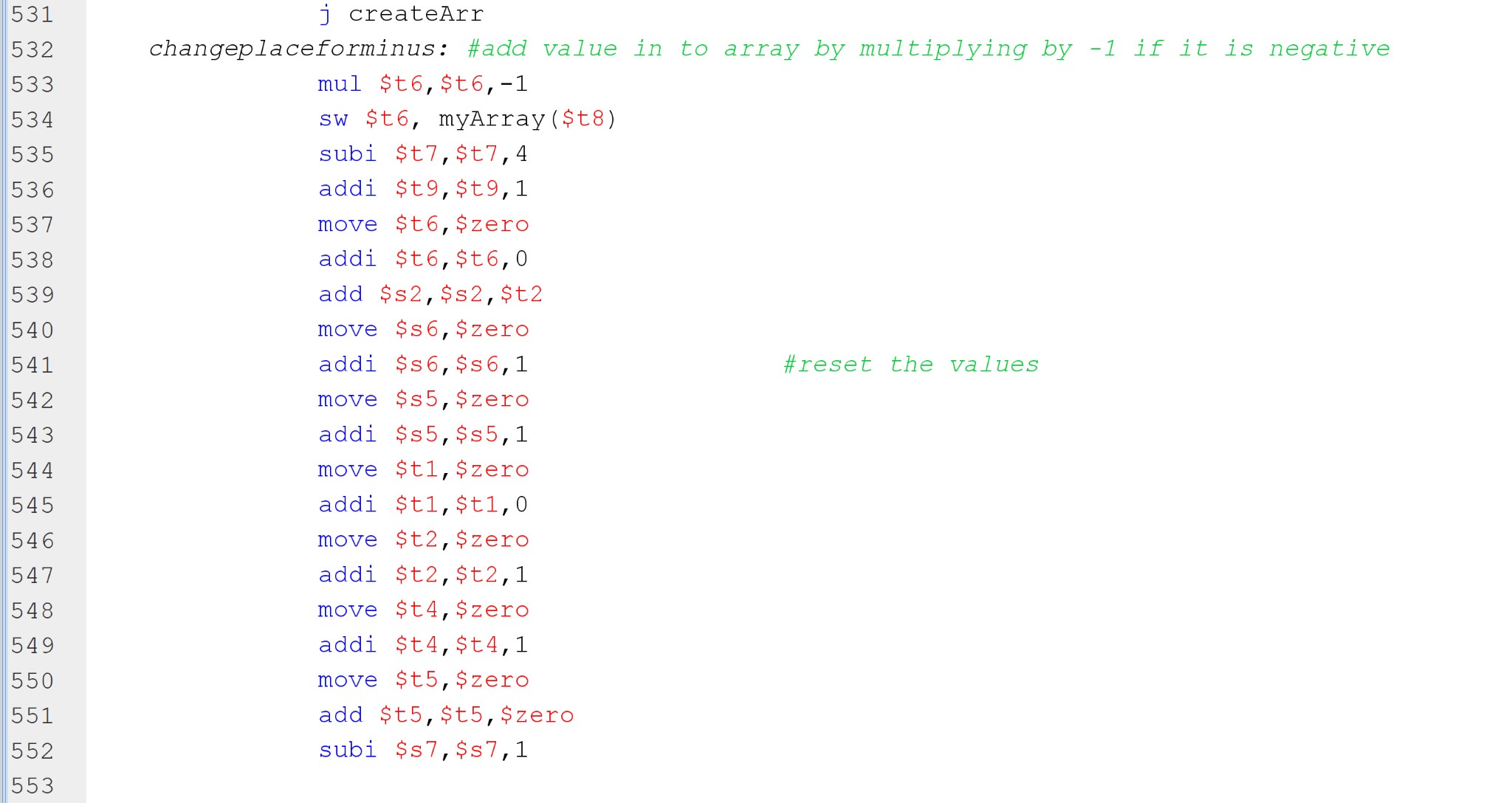
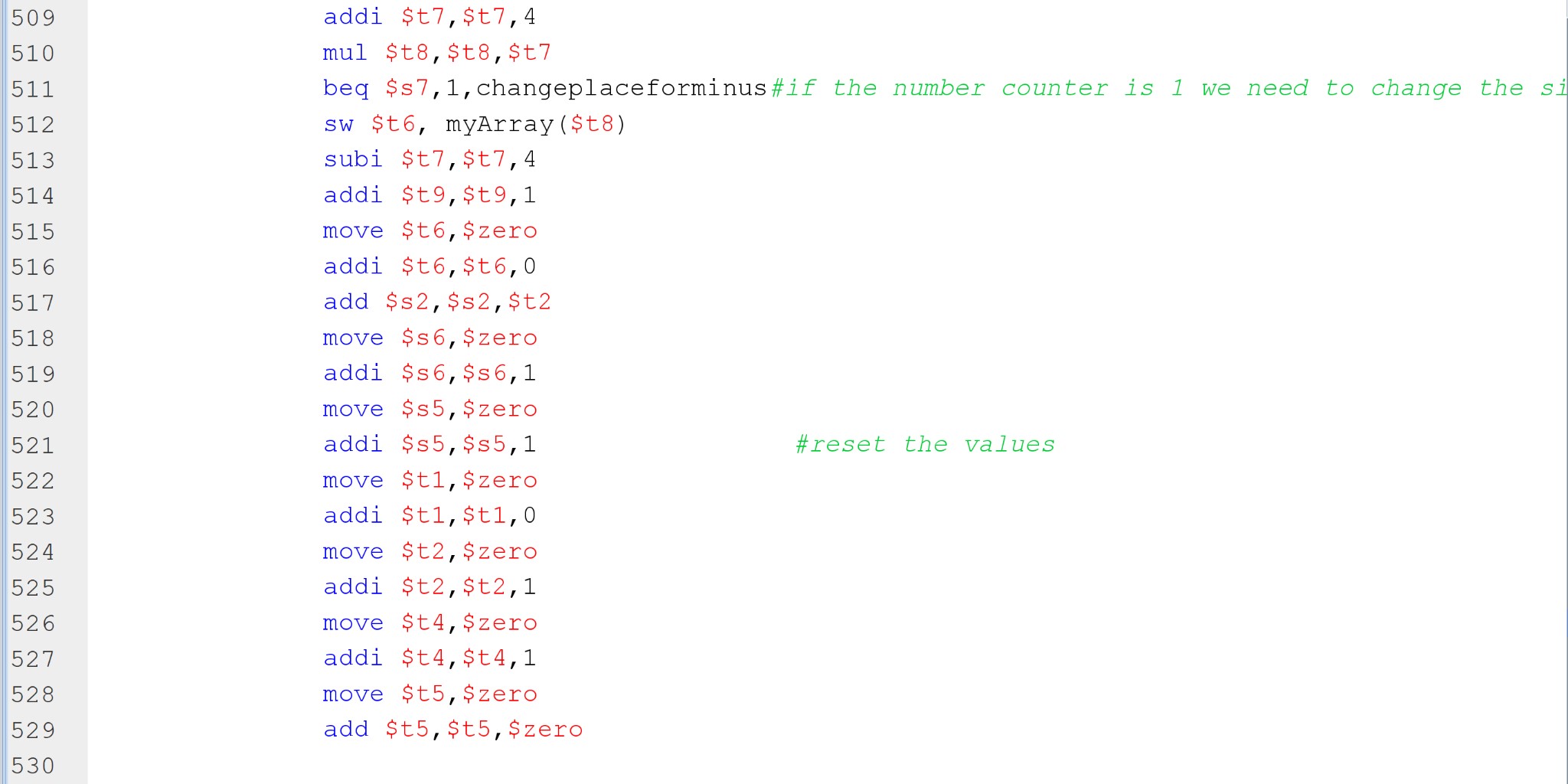
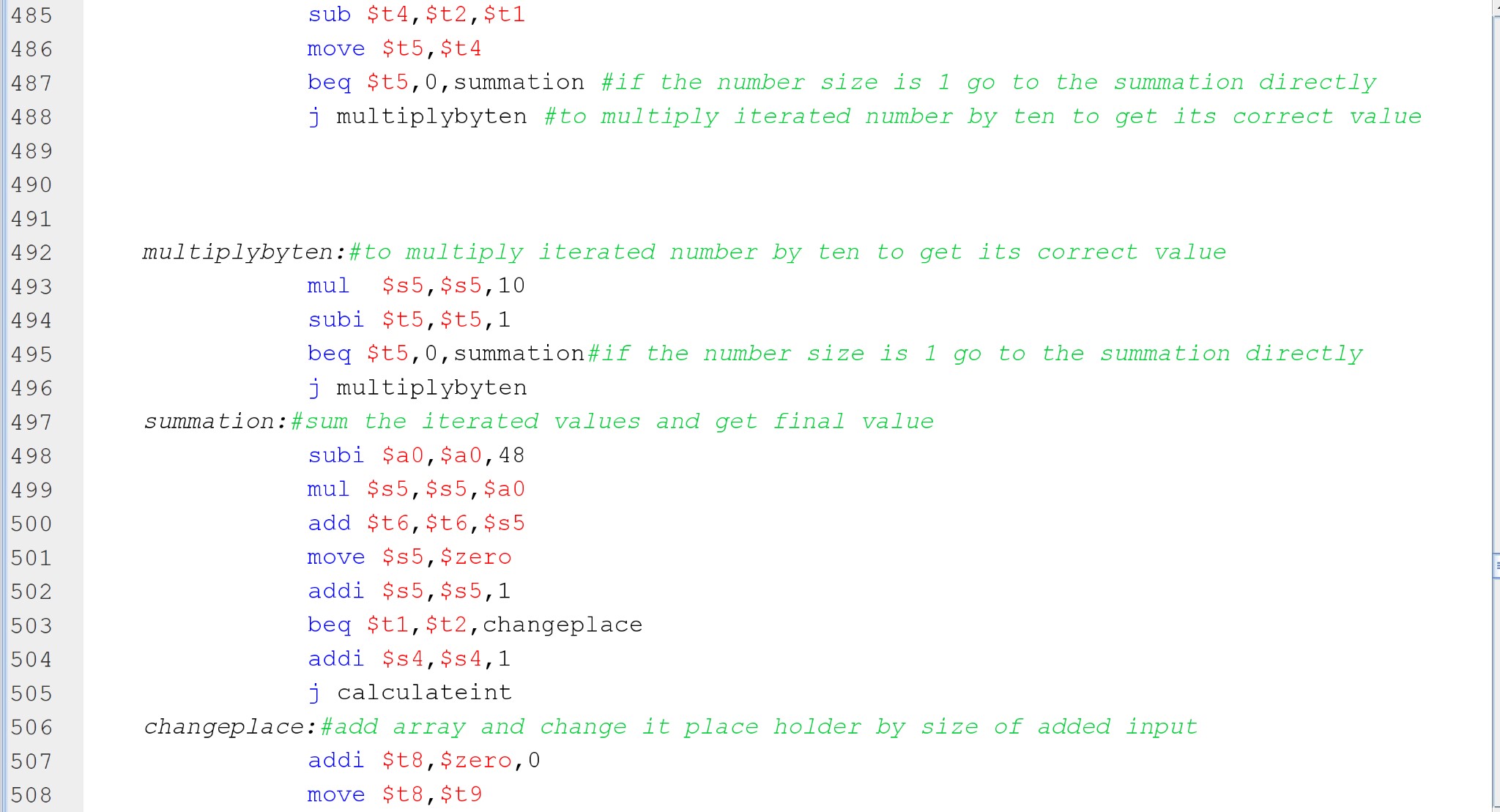
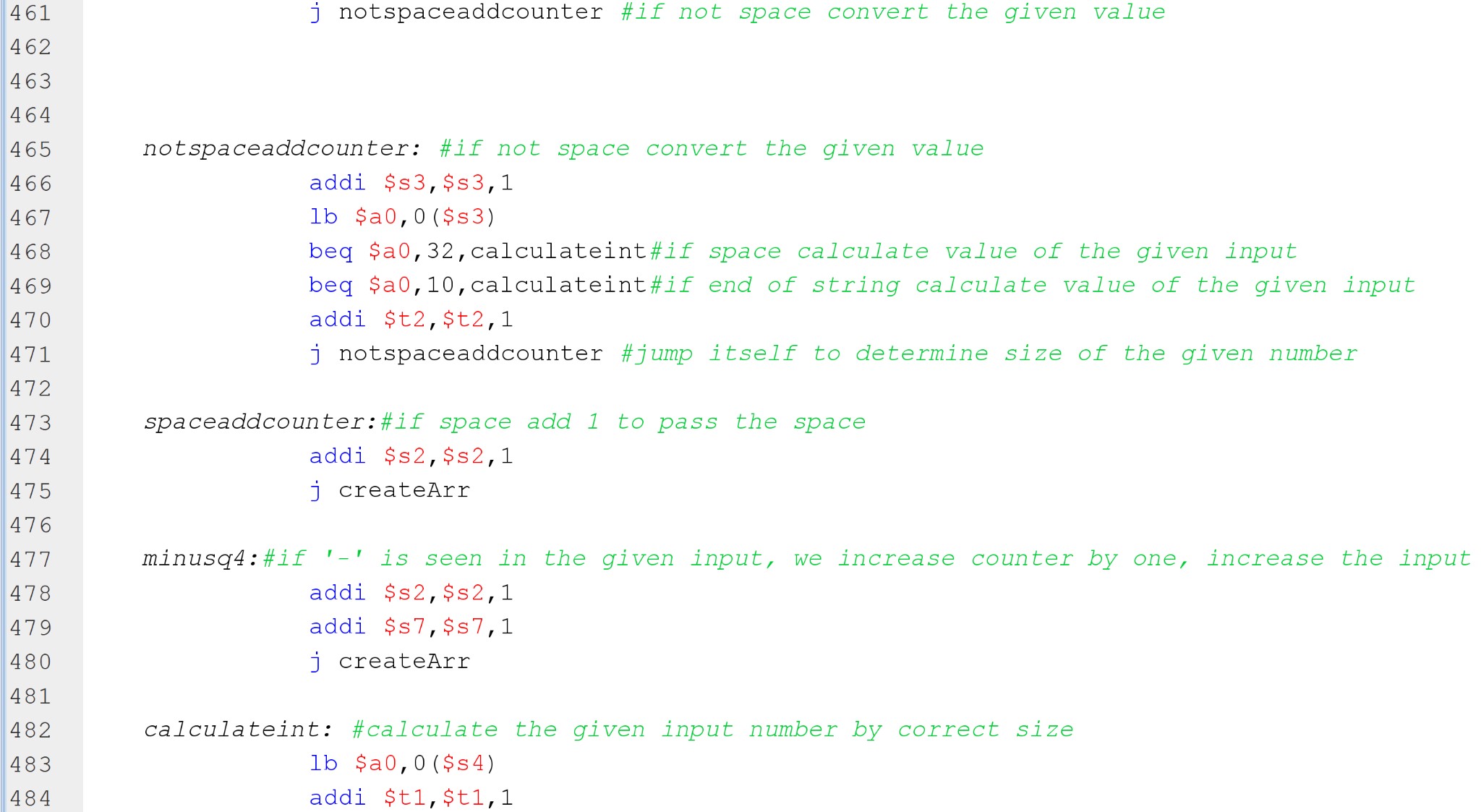
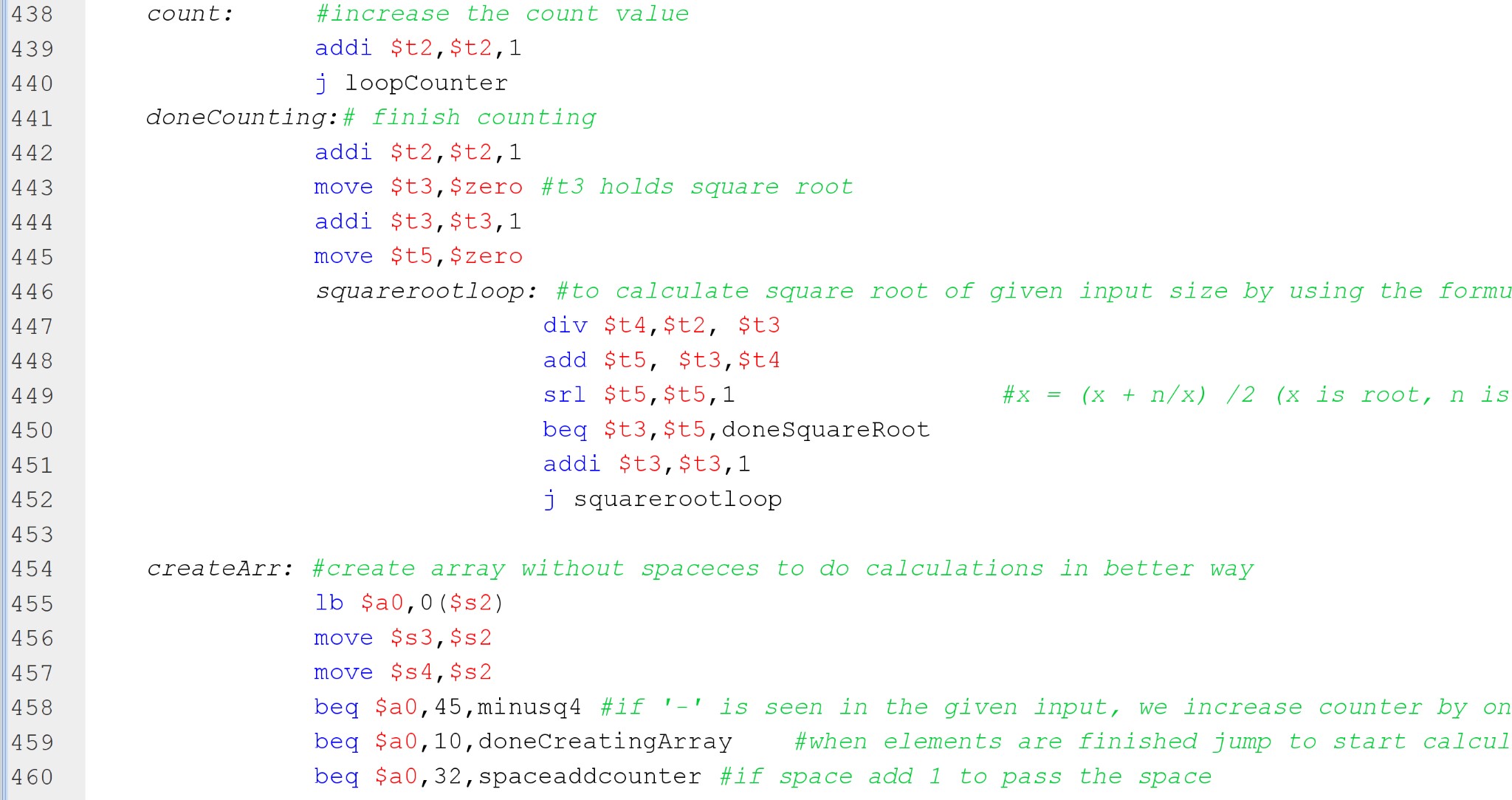
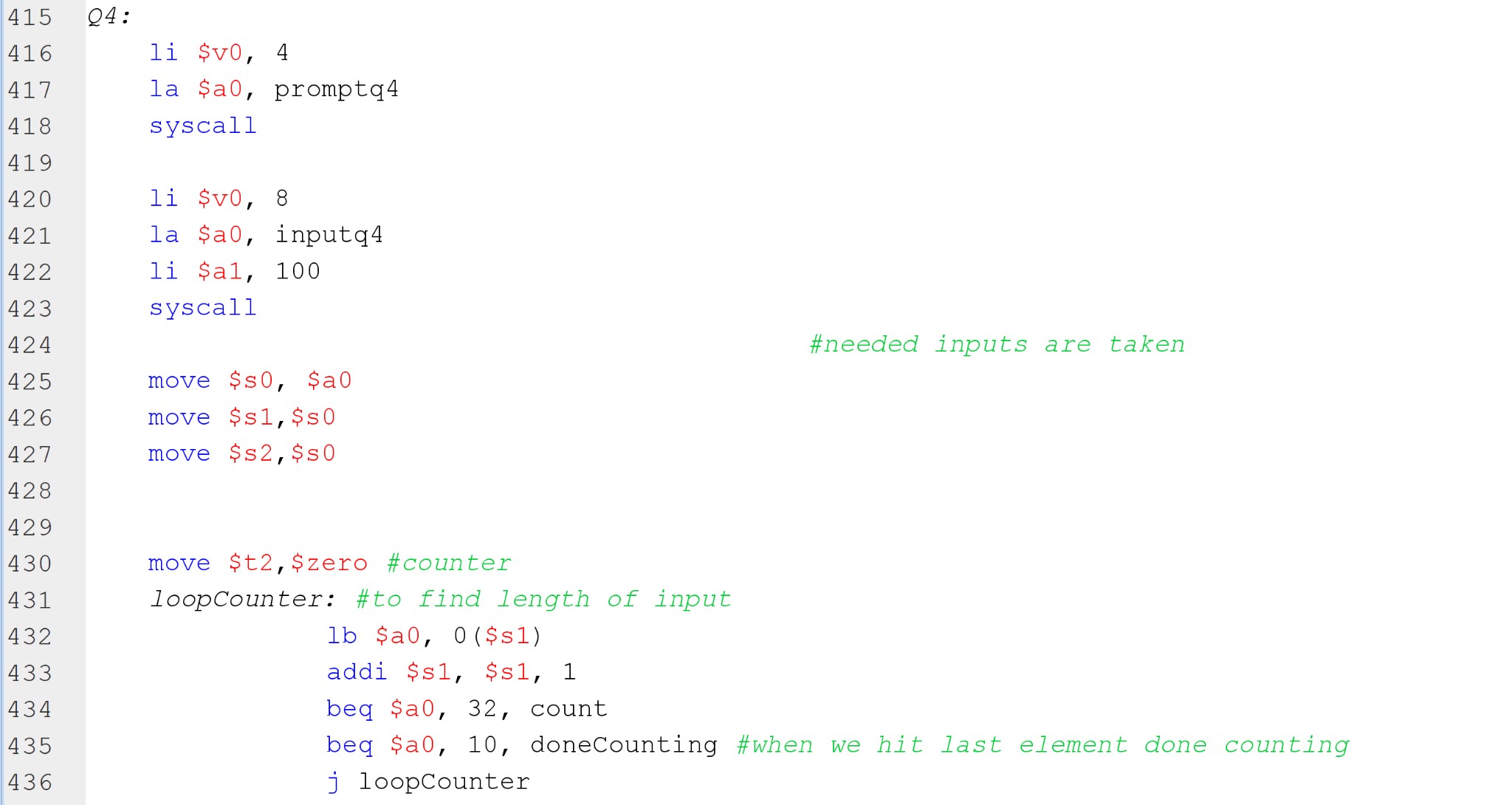
Run:



# Q4: Mystery Matrix Operation

Make some mysterious calculation in the given NxN matrix. First we calculate the input size and square root of the size to keep track of correct positioning. We convert given string input to integer array by calculating the string to integer operation(358=3\*10^2+5\*10^1+8\*10^0). If the given string starts with ‘-’, we calculate it the same way in calculating 358 but after that multiply it by -1 and append it to our array. After that we started to do matrix calculations. First we calculate left to right. It traverses by using N+1 and N-1 and updating its current position after these calculations. Therefore it makes N times for each part and totally N/2 parts. In addition after N times are finished our place holder increases by 4. After left to right, create a new line and start up to down calculation by choosing index 1 element. Do the similar operations from left to right. However instead of subtracting N-1 from our index we add N-1 into the index but in a different order than left to right.

Code:





Run2:



Run1:

