****

**MARMARA UNIVERSITY**

**CSE3038 PROJECT 1 REPORT**

**Group Members**

**150119908-Mertcan Çiy**

**150119637- Alperen Gönül**

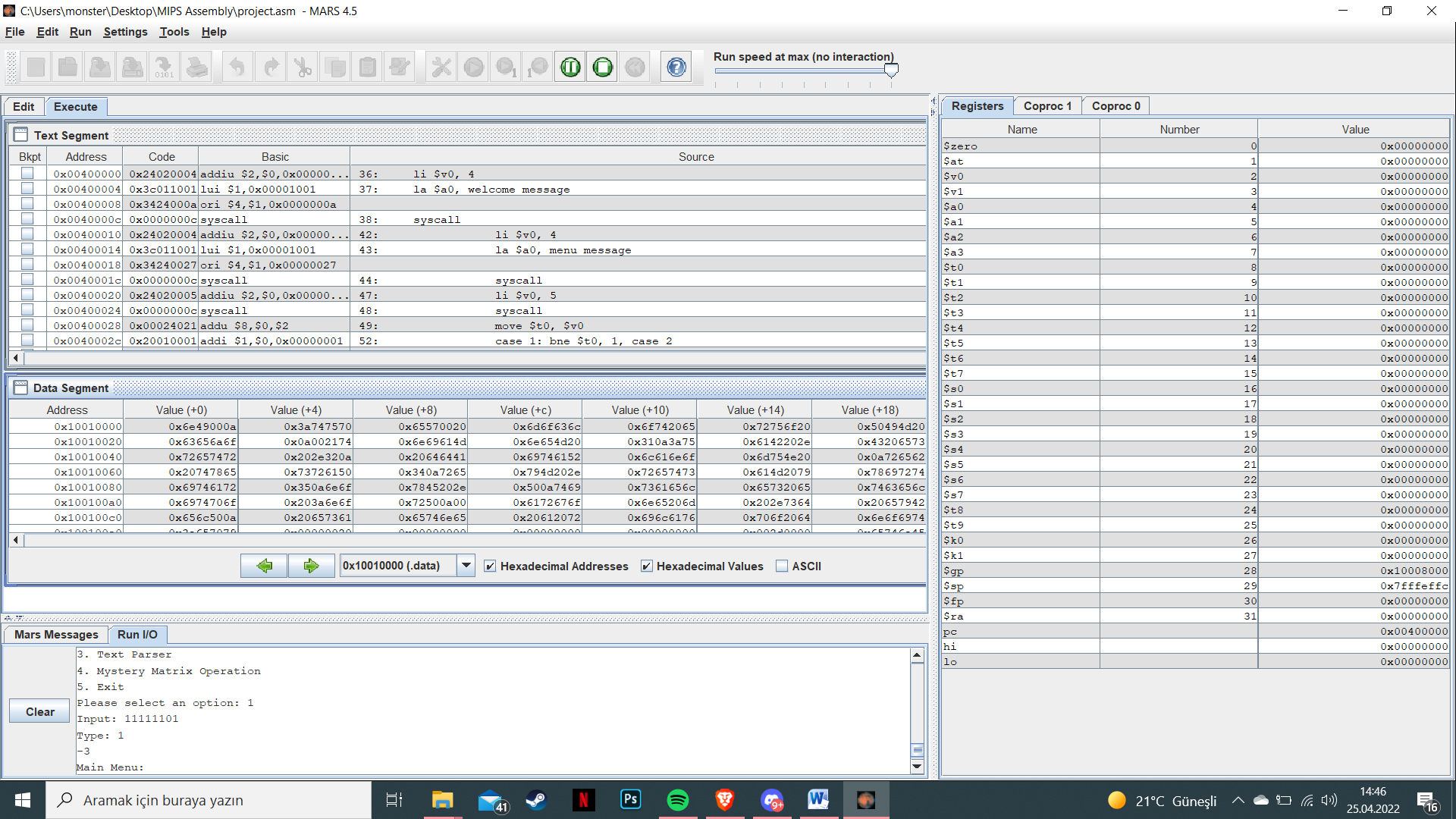
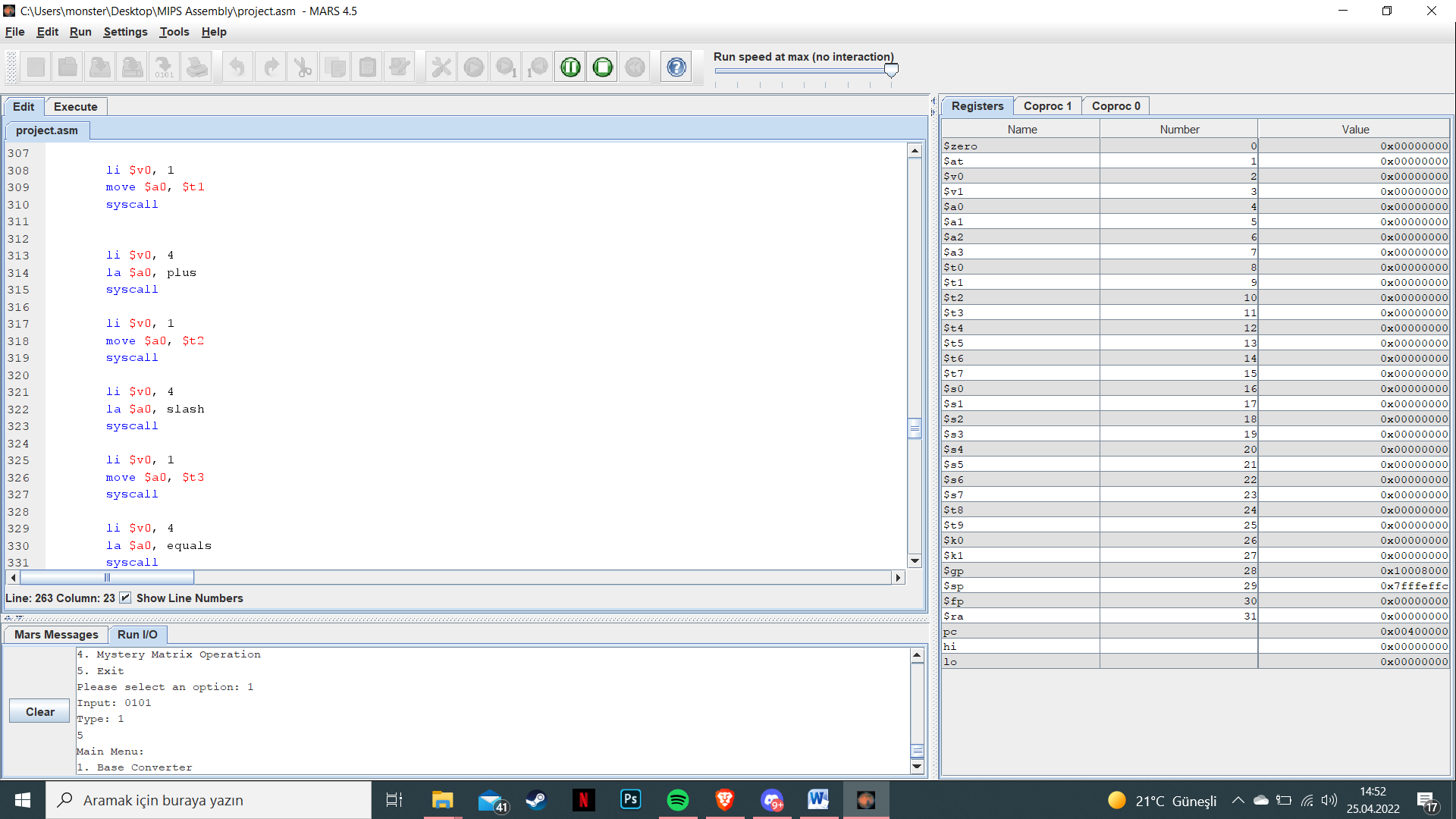
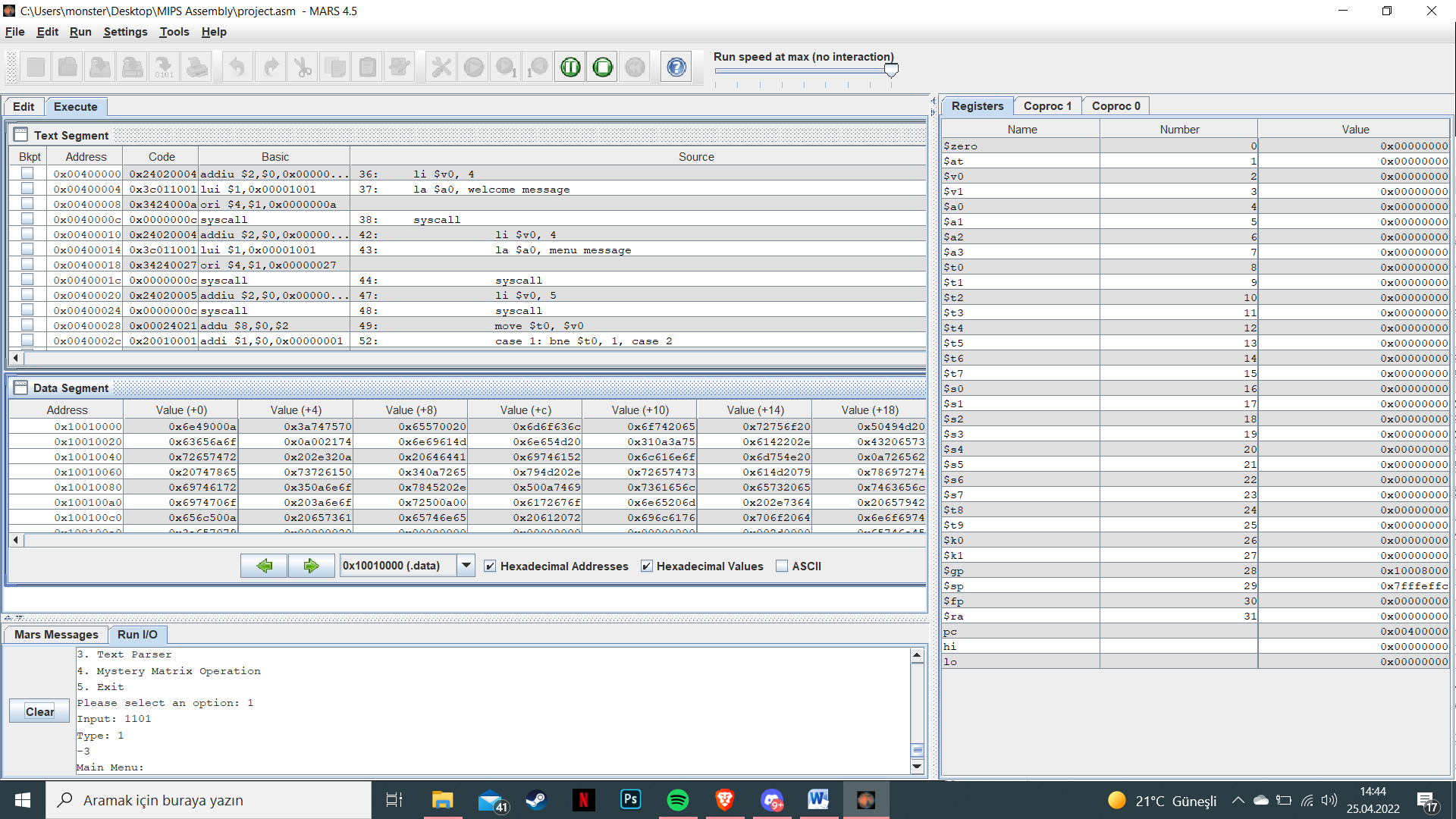
**150119527-Muhammet Emin Yelden**

**QUESTION 1:**

In Question1, firstly we take input from user as binary string and ask them to enter which Type they want to see. We control whether the user enter wrote 1 or 2 and direct the proper label(**Type1SolutionQ1**, **Type2SolutionQ1**).

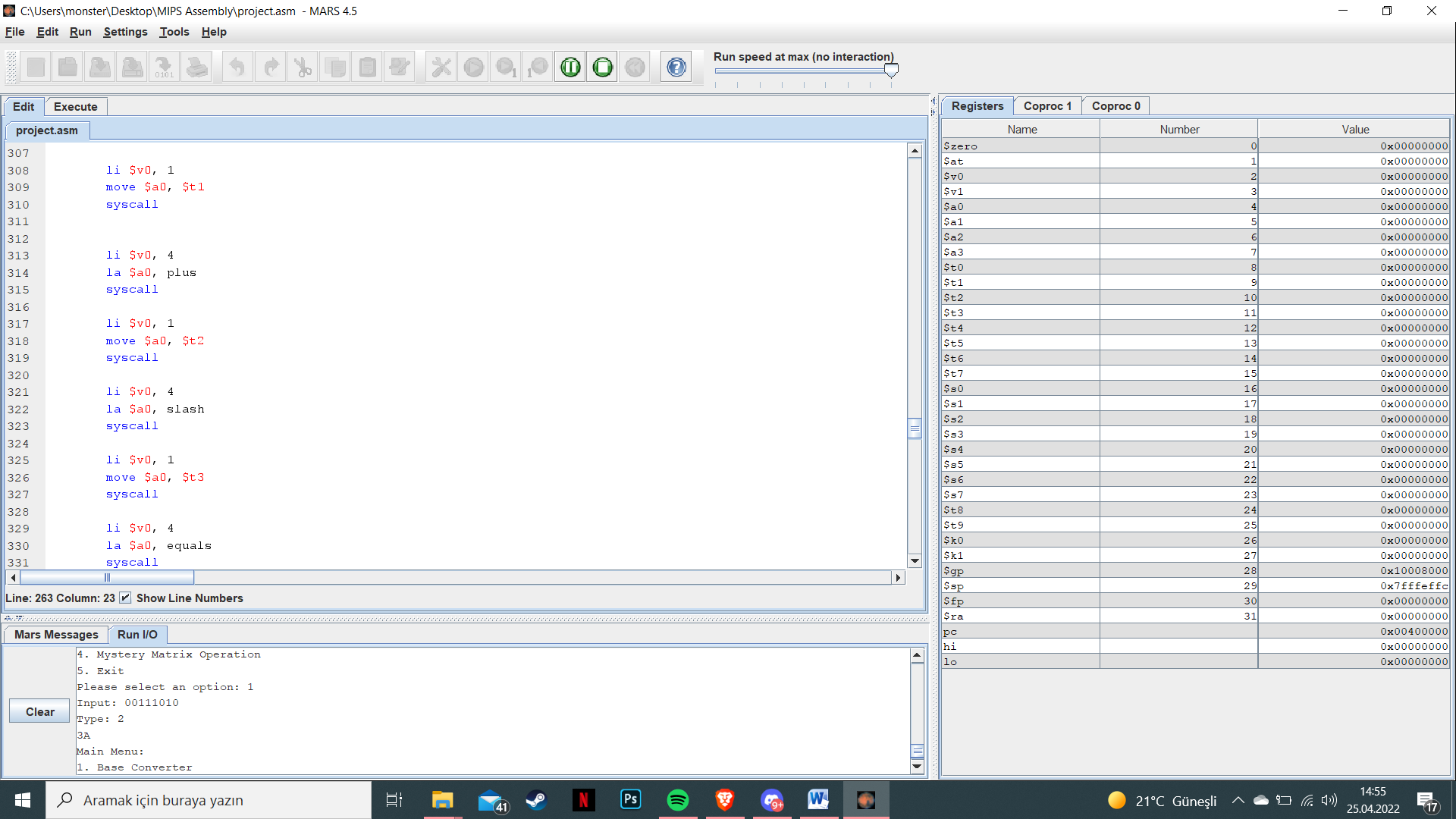
**For Type 1:**

First, we calculate the length of the input by using “**loop**” label and store the result to the $**t5** register. While calculating the length, we also store the bits in to the stack to use them later. Then, we check the first bit of the string to learn whether it is positive or negative. If the first bit is 0, it goes to “**pos**” label. We get the value in stack pointer, by substracting 48, we convert the ascii value to the decimal value(0 or 1). Then, if it is 0, we don’t have take it into calculation, moving the next bit via **“j posloop”** instruction. If it is 1, it goes to **calc2** label. By the way, we keep our position with $t5 and $t3 registers. $t5 is already our original length and at start we use move $t3, $t5 instruction and keep length also in $t3. While moving in bits, we substract them and get our position. By using this position value, while calculating the sum, we multiply the 1s with 2^(position value). For the case of first bit is 1, it goes to “**neg**” label and it calculates the first 1, then it keeps calculating when the current bit is 0 (we made our logic by thinking “converting 0s to 1s and 1s to 0s after meeting first 1” method). Also for the case of being the first bit is 1, we add negativity sign while printing.



**For Type 2:**

First, again we calculate the total length. After that, we use grouping counter to understand whether our current position is the multiple of 4 or not. We go into the **“whileLoopQ2”** label to start calculating. We store and calculate them until we reach the multiple of 4 position or end of the string. For first 4th bit, we calculate and store them into the stack. Then, we keep doing the same process. But, If the total length is not the multiple of 4 (Ex: 11 1010), for 5th and 6th positions we calculate them again and storing them into the stack. Then we print all the result.



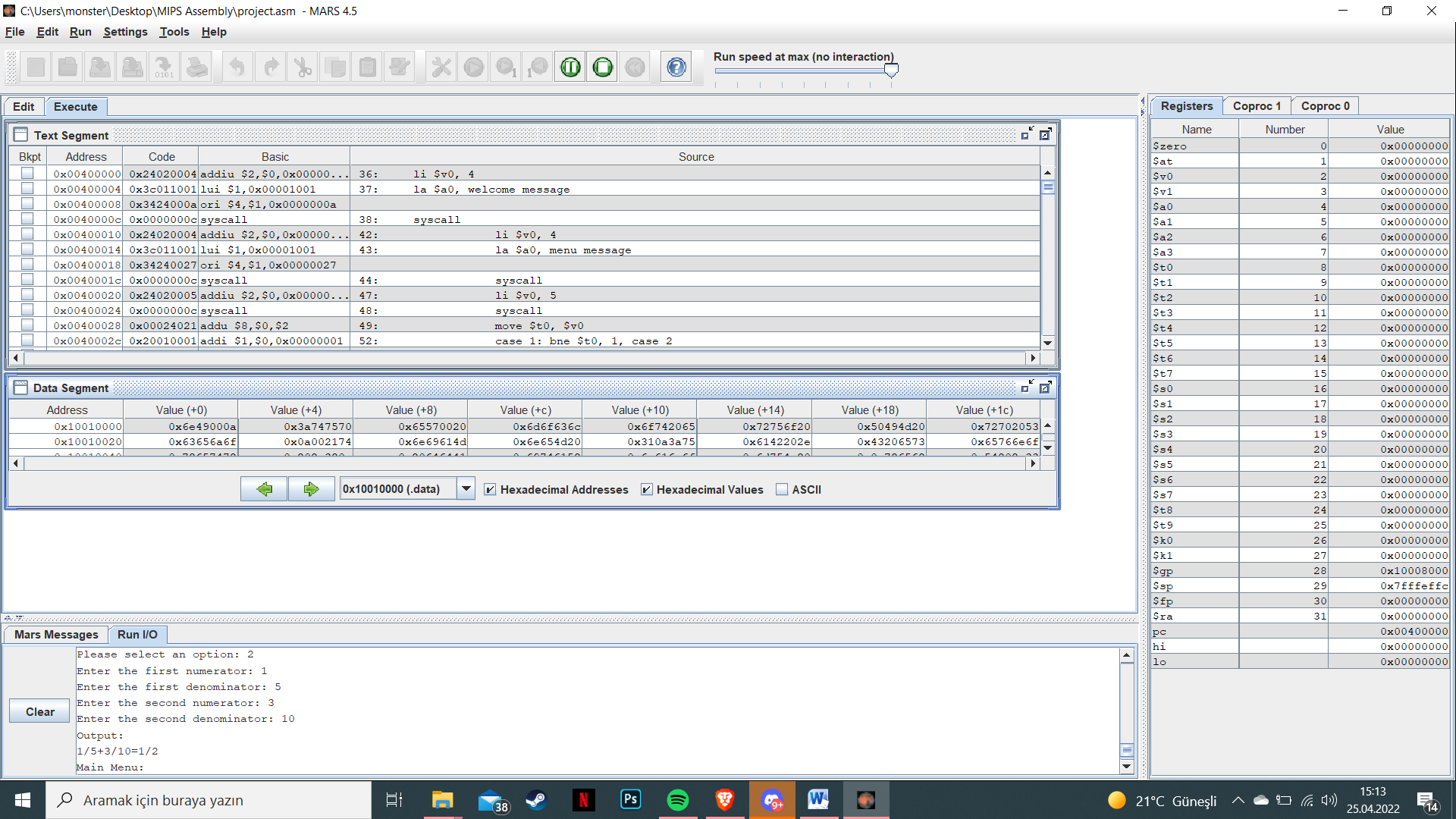
**QUESTION 2:**

In this question, first we get inputs from the user (first numerator, second numerator, first denominator, second denominator) and print them according to project PDF. Then, we multiplied denominators and adding them to the proper numerators. We need to find canonical form of

numerator1\*denominator2 + numerator2\*denominator1 / denominator1 \* denominator2

we stored the “denominator1 \* denominator2” to the **$t7 register** and “numerator1\*denominator2 + numerator2\*denominator1” to the **$t6 register.**

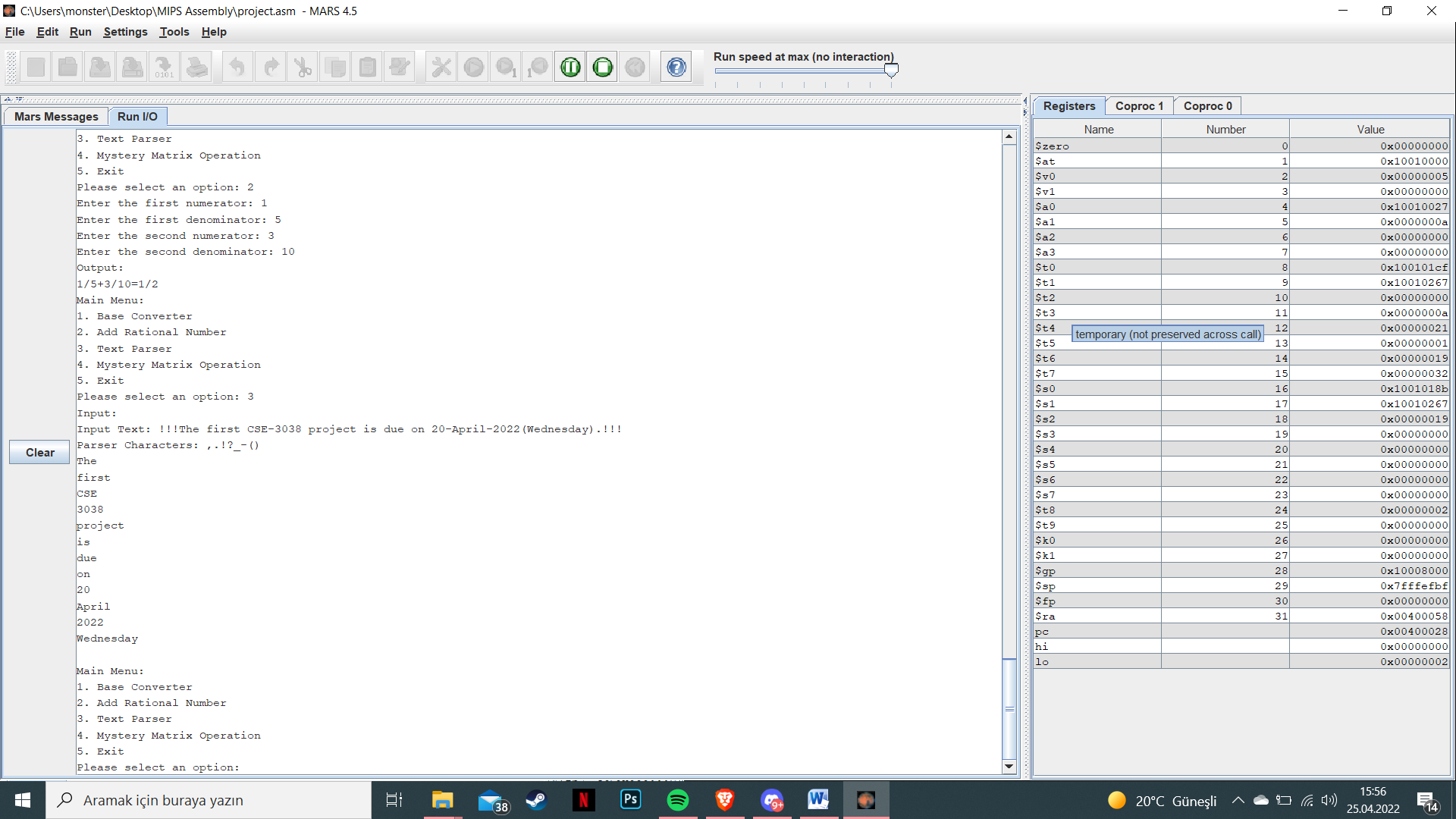
Our logic in this question is that we need to find common divisors for both of the numbers and keep doing the “finding common divisor by starting 1” process until reaching the numbers’ itself. We calculate the GCD of the numbers with this logic in **“forLoopGCD”** label.



**QUESTION 3:**

In this question, our logic is:

Firstly, we take input text and parser characters from the user. We are declaring a substring counter to control stack’s position and length of substring. Then, we are looping the first ch of the input text in the parsers. If they are same (It means that the character is delimeter) and substring counter is greater than 0(We are using this for example if input string is “!!!cse” we dont want to print after seeing first delimeter.) printing from stack. While printing stack, not to print them reversed, first we are moving back by the length of substring count and printing this way. For space character issue, just before controlling delimeter, we control space character with “**beq $t3, 32, forSpace”** and seperating proper places.



**MENU:**

In our menu, there is a main “**while**” label for main loop. We get input from the user for their choice and forward it by using **case labels like case\_1,case\_2** etc. For each question, there is a forward label and if the input is 5, it goes to **exit** label and finishes the program. Also, we control whether the input is valid or not. After the end of each question, we return back to our main loop. This is how our menu works well.

