Programming Assignment, Pgm2

1. Problem Statement:

The second programming assignment, Pgm2 for the course Computer Science 2 (CS5330.501) is a MIPS Assembly Language program that is intended to open/read a text file and display some basic statistics about the file. The statistics to be displayed are the number of uppercase letters, lowercase letters, number symbols, other symbols, lines of text, and signed numbers. The file name must not be hardcoded in the program; therefore, we require user input for the file name.

2. Approach:

The MIPS code for Pgm2 can be divided into two sections. The first section of the program is the main procedure where we get the file name from the user and do necessary changes to remove the end of line character and make it useful for reading the file. We then call several procedures to perform file operations like opening, reading, and closing. Apart from these we also have procedures to find the required file statistics like count of uppercase, lowercase, number symbols, other symbols, number of lines, and signed numbers. All the procedures that are being called by the main function are categorized as the second section of the program.

In the main procedure, the contents of the file are read into a variable which is then iterated from 0 to the end of the file. The iteration is done byte-by-byte and each character in the file is then analyzed using the procedures from section two to categorize into the relevant statistics. For example, if the character found in the first iteration of the file is ‘H’, then this is categorized as an uppercase character. Individual registers initialized as 0 are used to count the number of characters in each category during each iteration, ultimately, at the end of the whole iteration, we will have the file stats ready.

To identify the characters in the file and categorize them, we compare their hexadecimal ASCII values. The range of required ASCII values for this program is mentioned in Table 1. We use these values to construct our conditional statements.

Table : ASCII Values for File Statistics

|  |  |  |
| --- | --- | --- |
| **Category** | **Symbol Range** | **Hexadecimal Range** |
| Uppercase | A to Z | 0x41 to 0x5a |
| Lowercase | a to z | 0x61 to 0x7a |
| Number Symbols | 0 to 9 | 0x30 to 0x39 |
| Other Symbols | All others except above 3 | All others except above 3 |
| Line | \n | 0x0a |
| Signed numbers | + or - | 0x2b or 0x2d |

If a character has a value less than 0x41 (lower bound of uppercase alphabets) or more than 0x5a (upper bound of uppercase alphabets), then the character is not an uppercase alphabet. If the above condition is false then, it is an uppercase alphabet. Hence, the count will be increased by 1 and the flow returns to the main procedure. The same approach is used for both lowercase alphabets and number symbols by comparing the character with the respective lower and upper bounds. The general flow of this approach is shown in Figure 1.

Less than lower bound

More than upper bound

Return to main procedure

Increment counter

Character from Iteration in Main procedure

True

True

False

False

Figure : Flow of Operation for Uppercase, Lowercase and Number Symbols

The number of lines of text can be found by comparing the iteration character to the ASCII value of the line feed which is 0x0a. If the character is not equal to 0x0a, then return directly to the main procedure, else increment the counter and then return to the main procedure. The flow of this operation is shown in Figure 2.

Value != 0x0a

Return to Main procedure

True

Character from Iteration in Main procedure

Increment counter

False

Figure : Flow of Operation for number of lines

The signed numbers are interpreted that they either have a + (plus) or a – (minus) sign in front of them. Therefore, first, we check for plus or minus signs, if not found return directly to the main procedure, else go into a nested conditional statement. In this nested conditional statement, we now check if the next character is a number, if it is not a number return directly to the main procedure, else increment the counter and then return to the main procedure. This flow is depicted in Figure 3.

Return to main procedure

Character != +

Character != -

Return to main procedure

Increment counter

Character from Iteration in Main procedure

True

True

False

False

Is next character not a number?

False

True

Figure : Flow of Operation for number of signed numbers

Once the iteration in the main procedure is complete, the iteration counter of the iteration will have the value of the total number or characters in the whole file. Now to calculate the number of other symbols, we simply subtract the number of uppercase symbols, lowercase symbols, and number symbols from the total number of characters in the file. This count will include all characters that don’t belong in both alphabets and numbers.

3. Solution:

The above-mentioned approach has been successfully coded and executed in MIPS assembly language. Figure 4 shows the successful completion of the assembling operation.

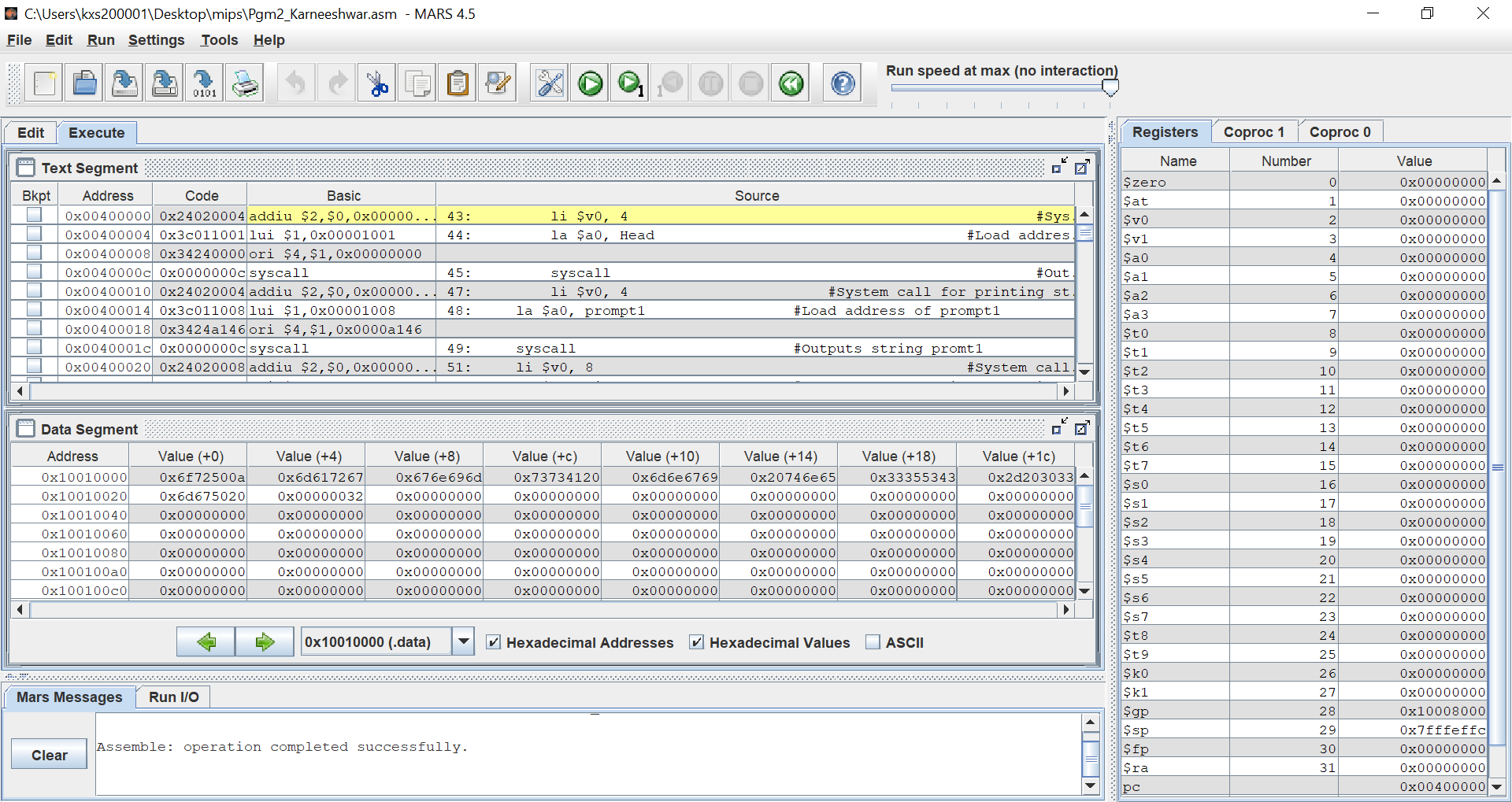


Figure : Screen capture of assembled code

To execute the program and check results, we need a sample text file. The file “test.txt” from the problem statement has been used for this purpose. The results of execution can be seen in Figure 5.

Text, table

Description automatically generated with medium confidence

Figure : Results

4. Conclusion:

The given problem statement has been analyzed, and the required outputs were delivered using the MIPS programming language. The code has been created to accept a file name as user input and to perform several conditional and logical operations to calculate the required file statistics. The results of these operations were displayed back to the user in a clear format.