Compute File Statistics using MIPS Assembly Language

1. Problem Statement

Write a MIPS Assembly Language program to produce the following statistics from an input file - upper case letters, lower case letters, number symbols, other symbols, lines of text, and signed numbers.

A sample is attached: do not hardcode the file name in your source code.

# Approach

To write this assignment program in MIPS assembly language, a simulator IDE called MARS (MIPS Assembler and Runtime Simulator) is used. The straightforward approach in solving the given problem is to first open and read the file in MIPS and then iterate through each character in the file checking against the hexadecimal ASCII values to increment the respective counters for uppercase letters, lowercase letters, number symbols, other symbols, lines of text, and signed numbers.

# Solution

ASCII (*which stands for American Standard Code for Information Interchange*) is a character encoding standard for text files in computers and other devices. These symbols consist of letters (both uppercase and lowercase), numbers, punctuation marks, special characters, and control characters. Each symbol in the character set can be represented by a Decimal value ranging from 0 to 127, as well as equivalent Hexadecimal and Octal values. In the following program, we make use of the following hexadecimal ASCII values: uppercase characters (0x41 – 0x5a), lowercase characters (0x61 – 0x7a), numbers (0x30 – 0x39), arithmetic operators (0x2b for plus sign, 0x2d for minus sign), line feed (0x0a), etc.

Several procedures or functions have been defined in the code to accomplish the given task. The main procedure is the entry point to the program which asks the user to enter the filename for which we need to compute statistics. The constrain put forward is to not have a filename of more than **30 characters**. Keeping the filename length static helps us to iterate and remove the line feed character at the end of the filename which makes the opening and reading of the file easier. While reading a file as a buffer of characters, it is always a good practice to have knowledge on the buffer length or buffer size. There is no “industry standard” buffer size to set while reading the file since it’s all dependent on the computers processing power. In this program, a small buffer size of **1000000** is kept reducing the latency and increasing the accuracy of the results.

After we have the file read as a buffer, we proceed to check each character one by one and for each character we try to determine whether it’s an uppercase letter, lowercase letter, number symbol, linefeed, any other symbol, or a signed number. Having a separate procedure for each of these cases with each procedures argument as the current character is a good way to count the total number of characters for the respective case. While it is a straightforward to determine if a character is an uppercase, lowercase, number and linefeed based on the hexadecimal ASCII values mentioned in the first paragraph, we must have some considerations for signed numbers and other symbols. In the code, **signed numbers are arithmetic operators (plus or minus) followed by a single digit number** **(1 – 9)** example +1, +2, -1, -2, etc. **Other symbols are considered as characters which are neither uppercase, lowercase letter and a number.** So, these can include linefeed, spaces, plus signs, hyphen/minus, commas, full stops, asterisks, etc.

Figure 1 shows the MIPS Assembler output of the written program. The program has been assembled successfully without any error. For the program demonstration, let us consider a sample input file (testFile.txt) shown in Figure 2. As the program runs, it asks for the file name to be entered by the user. After entering the sample file name, the program computes, and outputs the file statistics asked in the problem statement. This output is shown in Figure 3.

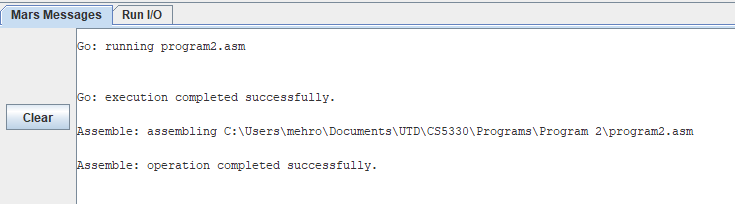


Figure 1: MARS Assembler Output

Graphical user interface, text, application, Word

Description automatically generated

Figure 2: A sample test file

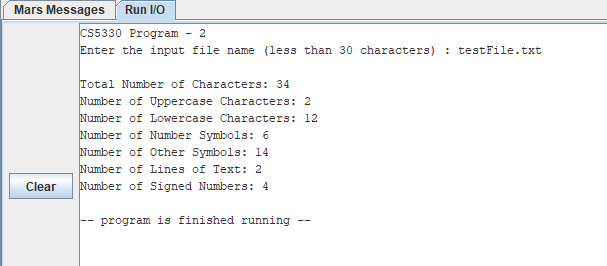


Figure 3: MARS program output