CIS-11 Project Documentation

**Team Name**

**Sebashtian Solis Torres, Gabriella Haines, Tommy Deth**

**Option C – Character Counter for Names**

**5-13-25**

**Advisor: Kasey Nguyen, PhD**

# Part I – Application Overview

### Our main goal for this project is to develop a functional and user-friendly LC-3 assembly program that accurately calculates and displays the frequency of each alphabetic character (A-Z) in a user-provided name string. Success will be measured by the program's ability to produce correct output for various test inputs, including edge cases (empty input, non-alphabetic characters, long names).

## Objectives

### We plan to improve the efficiency of character frequency analysis compared to manual methods. This could be measured by comparing the execution time of the program against the time it takes a human to manually count character frequencies. Additionally, the aim is to create a well-documented and maintainable codebase that adheres to coding best practices—which will be evaluated through code reviews and the clarity of the comments and documentation. Fundamental concepts of LC-3 assembly programming, including subroutine calls, stack management, and memory allocation will be applied to this project. All of which are assessed through code correctness and demonstration of mastery of these concepts.

#### Why are we doing this?

### To elicit the objectives, ask the business expert, the development manager, and the project sponsor the following questions:

* **What business objectives of the company will this project help achieve?**

Possible objectives might be reducing costs, improving the customer service, simplifying the workflow, replacing obsolete technology, piloting a new technology, and many others. Also, make sure you understand exactly how the proposed project will help accomplish the stated objective.

This project aims to demonstrate how useful various applications, such as linguistic analysis, data validation, or even simple educational tools. Doing this now allows us to develop a foundation for more complex text processing tools later.

* **Why are we doing this project now? What will happen if we do it later? What if we do not do it at all?**

The project is being done now to explore alternative programming paradigms and assess their suitability for resource-constrained environments. If delayed, the opportunity to gain experience with assembly language and its potential benefits may be missed.

* **Who will benefit from this project? Do the people who will benefit from it consider it the most important improvement that can possibly be made at this time? Should we be doing a different project instead?**

Anyone who needs to analyze character frequencies in text strings will benefit, from students learning about character encoding to researchers working with textual data. Whether it's the most important improvement at this time is subjective and depends on the priorities of the users, but the project's value lies in its foundational role and potential for future expansion. The development team and the project sponsor will benefit from this project. It allows the team to expand their skill set and the sponsor to evaluate the technology's applicability.

## Business Process

### The current business process likely involves manual character counting or using external tools. This application streamlines this process by automating character frequency counting and providing numerical output directly to the user. The future business process will incorporate this application, making character frequency analysis more efficient and precise.

### The application will be used in a standalone manner. The user will be prompted to enter their full name. The program will then analyze the input string, count the frequency of each character, and display the results in the console. This process does not directly integrate with any existing business systems. The value of this application lies in its educational and exploratory purpose, demonstrating the capabilities of assembly language for basic data processing.

## User Roles and Responsibilities

### Casual User: This user needs a simple, easy-to-use tool for quickly determining the character frequencies in a name. Their interaction is limited to entering a name and interpreting the numerical output. They don't need to understand the underlying algorithms or technical details. Their main responsibility is providing accurate input. Error handling should be robust to accommodate potential mistakes (e.g., extra spaces, non-alphanumeric characters).

### Data Analyst: This user might use the application as part of a larger data analysis task. They require more detailed output (potentially in a format suitable for further processing), possibly with options for handling different character sets or encoding schemes. They understand the underlying concept of character frequency analysis and may use the output for further statistical analysis or reporting. Their responsibilities include selecting the appropriate options, interpreting the results in context, and validating the accuracy of the output against other data sources.

### Developer/Maintainer: This user is responsible for maintaining and potentially expanding the application. They need access to the source code, documentation, and testing procedures. Their responsibilities encompass debugging, adding new features, optimizing performance, and ensuring the application's stability and security.

## Production Rollout Considerations

### The rollout would involve testing the application with various names to ensure accuracy and handling of edge cases (e.g., special characters, empty input). Data population is minimal, as the application only needs to manage the character frequency array. The expected data and transaction volume will be low to moderate depending on usage.

## Terminology

### Character Frequency: The number of occurrences of each unique character within a given text string. This is a fundamental concept in text analysis and is used in various applications, from simple word frequency counting to more advanced natural language processing techniques.

### ASCII (American Standard Code for Information Interchange): A character encoding standard that assigns unique numerical values (0-127) to characters, including letters, numbers, punctuation marks, and control characters. The LC-3 architecture uses ASCII to represent characters in memory. Understanding ASCII is crucial for working with text data in LC-3 assembly programming.

### Alphabetic Character: In this context, it refers specifically to uppercase letters (A-Z). The program could be extended to include lowercase letters or other character sets, but the current scope limits it to uppercase for simplicity.

### Subroutine: A self-contained block of code that performs a specific task and can be called from other parts of the program. Subroutines improve code modularity, readability, and reusability!

# Part II – Functional Requirements

## Statement of Functionality

### The following project will:

### • Prompt the user to input their full name

### • Convert each character from ASCII to an indexable format

### • Count how many times each character appears

### • Display each character and its corresponding count to the console

### • Handle uppercase and lowercase characters, and negate spaces

### • Use subroutines for input, character counting, and output display

### • Implement stack operations (PUSH/POP) for subroutine context saving

### • Apply branching for looping through characters and checking counts

### • Use memory efficiently to store input and frequency data

## Scope

### This program will:

### Be developed in a single phase

### Deliver a fully functional LC-3 program that meets all requirements (e.g., subroutines,

### branching, stack usage, ASCII conversion).

### Focus entirely on character frequency counting from a single user-input string.

## Performance

### The program must handle input strings up to 40 characters

### It must correctly count and display character frequency with no delay or crash under normal input

### Stack operations must preserve register values during subroutine calls

## Usability

### The interface is text-based in the LC-3 simulator

### User interaction consists of:

### A prompt for input

### Clean display of output with each character and its frequency

# Documenting Requests for Enhancements

There does come a time when the requirements for the initial release of your application are frozen. Usually, it happens after the system acceptance test which is the last chance for the users to lobby for some changes to be introduced in the upcoming release.

Currently, you need to begin maintaining the list of requested enhancements. Below is a template for tracking requests for enhancements.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Enhancement** | **Requested by** | **Notes** | **Priority** | **Release No/ Status** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Part III – Appendices

*Program Pseudocode*

Display output message on screen “Please input a name:”

Prompt user for string input

Get input from user

* Initialize character array (to store string - each char in the string)
* Initialize char frequency variable

Get length of string and store in variable

Create iterative/conditional subroutine & then call it

Branch to iterative loop (for loop)

* Iterates through each char in the array
* Utilize string length var -> int i = 0; i < length; i++

Nested conditional loop within iterative loop

* If loop - conditional
* Counts frequency of letters
* If char found more than once, increment frequency counter

Create program output subroutine and call it

Display program output - frequency of each letter (counted values in console)

\*\***Implement**

* ASCII conversion
* Stack: PUSH-POP operations
* At least one pointer - possibly to the array itself
* At least 2 subroutines + subroutine call

*Program Flowchart*

A diagram of a flowchart

AI-generated content may be incorrect.