# CIS11 Course Project Part 1: Documenting the Project

Fill in the following areas (purple).

**Introduction**

* 1. **Purpose**

The purpose of this project is to create an LC-3 assembly program that implements bubble sort algorithm which is comparison-based algorithm in which each pair of adjacent elements is compared, and the elements are swapped if they are not in order.

* 1. **Intended Audience and Users**

This program can be used by individuals interested in low-level programming and computer architecture.

* 1. **Product Scope**

The intention of this program is to provide a practical example of sorting algorithms implemented in LC-3 assembly language. The program's scope includes user input handling, sorting algorithm implementation, and displaying the sorted results.

* 1. **Reference**

**Source Documents for the Program Requirements and Specification**

**The program must fulfill the following criteria:**

1. Contain appropriate addresses: origination, fill, array, input, and output.
2. Display sorted values in console.
3. Use appropriate labels and comments.
4. Contain appropriate instructions for arithmetic, data movement and conditional operations.
5. Comprise of 2 or more subroutines and implement subroutine calls.
6. Use branching for control: conditional and iterative.
7. Manage overflow and storage allocation.
8. Manage stack: include PUSH-POP operation on stack.
9. Include save-restore operations.
10. Include pointer.
11. Implement ASCII conversion operations.
12. Use appropriate system call directives.

**Input:** User input 8 numbers, ranging from 0 – 100.

**Output:** Display sorted values in ascending order in console.

**2. Overall Description**

**2.1 Product Perspective**

The program will read user input, sort the input using the Bubble Sort algorithm, and display the sorted output. The data type used will be integers within the range of 0 to 100.

* 1. **Product Functions**

**The overall description of functionality:**

* Input Handling: Read 8 integers from the user. Convert ASCII input to binary and store in memory.
  + Purpose: To gather the data that will be sorted.
  + Inputs: Characters from the keyboard.
  + Outputs: Integers stored in memory.
  + Data: ASCII characters converted to binary integers.
  + Data Storage: Internal to the application, stored in the array in memory.
* Sorting: Implement Bubble Sort to sort the integers in ascending order.
  + Purpose: To arrange the integers in ascending order.
  + Inputs: Unsorted integers stored in memory.
  + Outputs: Sorted integers in memory.
  + Data: Binary integers.
  + Data Storage: Internal to the application, sorted in the same array.
* Output Handling: Convert binary integers to ASCII and display them on the console.
  + Purpose: To present the sorted integers to the user.
  + Inputs: Sorted integers from memory.
  + Outputs: ASCII characters displayed on the console.
  + Data: Binary integers converted to ASCII characters.
  + Data Storage: Displayed directly on the console.

**Technical functionality:**

Subroutine for Input Handling, Bubble Sort Algorithm, Subroutine for Output.

* 1. **User Classes and Characteristics**

Users that are learning about assembly language and sorting algorithms.

* 1. **Operating Environment**

System Type: The program will be operated on LC-3 simulators and can be used on any operating system that can run the LC-3 simulator.

Development Platform: The program will be developed and tested using an LC-3 assembly language simulator.

Simulator Version: Any version of the LC-3 simulator that supports the standard LC-3 instruction set.

* 1. **Design and Implementation Constraints**

Memory Limitations: The LC-3 has a limited address space, so efficient use of memory is necessary.

* 1. **Assumptions and Dependencies**

Simulator Dependency: The application is dependent on an LC-3 simulator for execution.

No External Services: The program does not rely on any web services or external applications.

***3*. External Interface Requirements**

* 1. **User Interfaces**

Users will interact with the program via the console, inputting numbers and viewing the sorted output.

* 1. **Hardware Interfaces**

Computer Types: The program will run on any computer capable of running the LC-3 simulator.

* 1. **Software Interfaces**

The program requires the LC-3 simulator to assemble and execute the code.

* 1. **Communications Interface**

The application does not require web, internet, or network connectivity.

**4. Detailed Description of Functional requirements**

**4.1     Type of Requirement (summarize from Section 2.2)**

Read 8 integers from the user. Convert ASCII input to binary and store in memory, Implement Bubble Sort to sort the integers in ascending order, convert binary integers to ASCII and display them on the console.

**4.2 Performance requirements**

The program should sort the numbers in a reasonable time.

**4.3 Flow Chart OR Pseudocode.**

|  |
| --- |
|  |
| 1. **Set up memory locations for input array, temporary storage, and counters.**    * Define the starting address for input array.    * Define temporary storage for swap operations.    * Initialize counter for number of elements (N = 8).   **Input**   1. **Read 8 user inputs.**    * Loop 8 times to read inputs.    * Store the integers in the input array.   **Bubble Sort Algorithm**   * + Outer loop (i) from 0 to N-1.     - Initialize swapped flag to false.     - Inner loop (j) from 0 to N-i-1.       * Compare array[j] and array[j+1].       * If array[j] > array[j+1], swap them and set swapped flag to true.     - If no swaps occurred in the inner loop, break the outer loop early.   **Output**   1. **Display sorted array:**    * Loop through the sorted array.    * Display each number. |  |

|  |
| --- |
|  |
|  |  |