CIS-11 Final Project Documentation

**J&V&E**

**Jesse Martinez**

**Victoria Avila**

**Elizabeth Grajeda**

**Project Name: Bubble Sort**

**5/24/25--5/26/25**

**Advisor: Kasey Nguyen, PhD**

# Part I – Application Overview

This LC-3 program is designed to sort user-inputted values using the Bubble Sort algorithm, and output said values in ascending order. This program's intention is to extend out to educational purposes, providing an interactive environment to learning programmers on how to implement algorithms; such as this sorting program.

## Objectives

#### Why are we doing this?

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

The following comments come from the team behind this project:

**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.** Unfortunately, this project does not involve a business expert/development manager, so this question is N/A (not applicable).

**Why are we doing this project now? What will happen if we do it later? Or not at all?** We are completing this project to practice team collaboration, along with enhancing our proficiency in assembly programming. Doing it now will allow us to showcase the skills we have learned and apply them to a team setting that shows we are able to work alongside other developers as is required in the field.

**Who will benefit from this project?** This LC-3 program is designed for individuals in the educational sphere. This program will benefit computer science students / individuals trying to learn assembly language the most. Educators can also use and reference this project as a learning tool for their students, either by showing their students this program or having their students create and/or use bubble sorting algorithms on their own.

## Business Process

Existing business process:

1. Decide on which program to create, as a group, that is an LC-3 sorting algorithm for numbers between 0 -100 that implements Bubble Sort and displays the values in ascending order.
2. Assign everyone to collaborate on the final documentation to help establish what goals and criteria we seek to see in our final code.
3. Plan and map the flowchart that will help decide how we want our code to work. Then write out the code.
4. Test small numbers first that don’t need to be converted to ASCII.
5. Test with larger numbers that require ASCII conversion
6. Monitor the results we get with larger numbers and if needed/required to fix
7. Repeat step three if we need to fix/alter any portion of our code that isn’t giving us the expecting outcome.

Future business process:

In the case of wanting to expand our program to allow it to take numbers that are either negative OR bigger than 100, the business process would mostly be the same, to get everyone on the right page about what values we wish to be able to input into our program and how to map the flowchart that will help us write the subroutines to allow this to expand beyond 0-100.

## User Roles and Responsibilities

1. Victoria Avila

* Responsibilities
  + Coordinating group work and communication
  + Program Documentation
  + LC-3 Programming
  + Flowchart

1. Jesse Martinez

* Responsibilities
  + LC-3 Programming

1. Elizabeth Grajeda

* Responsibilities
  + Program Documentation
  + LC-3 Programming
  + Flowchart

## Production Rollout Considerations

1. Create a timeline
   1. Roadmap that establishes deadlines and milestones for getting parts of the project documentation and then the actual project completed.
2. Rollout plan
   1. Define what users of this program will see and if we get the expected outcome and consider user input error, and accounting for things that could go wrong.
3. Build and Test
   1. Check and ensure that the program is built correctly
4. Communications plan
   1. Get everyone on the same page about any/if any future changes and improvements once the program is released

## Terminology

Lc-3: Also known as Little Computer 3, is an assembly language and educational language for computers.

Push-pop: This is an operation that can be used in lc-3 and is used to sort data. It has the ability to swap data locations. In this project, the push/pop operation will be used to reorganize the 8 values that the user gives to the program, and rearrange them in ascending order.

Algorithms: In lc-3, along with other programming languages, algorithms are instructions that are written with the intention of having a program perform a task(s).

Condition Code: The condition codes are N (negative), Z (zero), and P (positive) that will be used for conditional branching.

Branch: An instruction that changes the flow of execution based on a condition like ‘BRz’, ‘BRn’, ‘BRp’.

Effective Address: The actual memory address calculated during the execution of an instruction using base registers and offsets.

Pass: A single iteration through our given array of numbers in a sorting algorithm, which this bubble sort will make multiple of to sort it.

Swap: The process of exchanging the values of two variables, which is commonly used in a sorting algorithm such as this one.

Trap: Instruction used to handle system calls or interrupts that invokes system routines like ‘Trap x25’ for HALT.

Directive: Instructions to the assembler used to set up the program structure and data like ‘.ORIG’, ‘.FILL’, and ‘.STRINGZ’.

Load/Store: Instructions used to move data between memory and registers such as ‘LD’, ‘LDR’, and ‘STR’.

***Business Terminology used in previous sections:***

Timeline: A schedule outlining milestones from development through release and maintenance.

Documentation: Create comprehensive documentation (such as this) to help users get started.

Resource Allocation: Ensure that team members are assigned tasks such as coding, documentation, support, and testing.

# Part II – Functional Requirements

## Statement of Functionality

The following list includes the agreed requirements of the program:

1. Appropriate addresses such as: origination, fill, array, input and output.
2. Console ability to display sorted user-inputted values
3. Appropriate labels and comments
4. Appropriate instructions for arithmetic, data movement and conditional operations
5. 2 or more subroutines that are implemented within the program
6. Branching controls: conditional and iterative
7. Overflow and storage allocation management
8. Stack management: push-pop operations
9. Save-restore operations
10. Pointer implementation
11. ASCII conversion operations
12. System call directives
13. Tested with the values 11, 8, 2, 17, 6, 4, 3, 21 prior to submission

## Scope

The intention of this program is to receive a set amount of values, in this case 8 submitted by the user, and then rearrange said values in ascending order. This program is limited to numeric values that range between zero to one hundred(0-100). This is the final phase that will be delivered.

## Performance

Time Execution: ) where n is the number of elements and the larger the array inputted, the more the performance slows. If no swaps are made, the program could terminate early and thus reduce the number of cycles needed to sort.

Memory Usage: Requires a constant amount of extra memory space for sorting like a couple of registers and a small number of temporary variables. Should at most use (10) additional memory locations to account for those variables and pointers.

Code Efficiency: Early exit mechanism where it terminates early if only one element in the list.

## Usability

### In this section you describe any specific usability requirements.

### You need to include this section only if there are any “overarching” usability goals and considerations. For example, the speed of navigation of the UI may be such a goal. As in the previous section, use numeric measures of usability whenever possible.

Not Applicable to this program as there are no extra requirements for running this program aside from the lc-3 editor alongside the lc-3 simulator.

# 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** |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### This portion is N/A ( Non-applicable ) to our program as it will be a one time release (upon submission). There is no official supervisor that overlooks this program who could request enhancements to our team.

# Part III – Appendices

In this section, our team would like to include some references that we used to start this project, alongside this document and flowchart creation.

### Michael Sambol. “Bubble Sort in 2 Minutes.” Youtube, 26 July 2016, <https://www.youtube.com/watch?v=xli_FI7CuzA&t=53s>. Accessed 24 May 2024.

1. “How to Make a Sorting Algorithm in Assembly Code, in LC3.” *Stack Overflow*, stackoverflow.com/questions/43735625/how-to-make-a-sorting-algorithm-in-assembly-code-in-lc3. Accessed 25 May 2024.
2. Kasey Nguyen. “CIS11 Unit Lecture.” Youtube, 19 April 2023, <https://www.youtube.com/watch?v=-8jSbe1VEc4&list=PL2W-cwk5hZgQCG5jh2_VgjpB27P8x8G-Y&index=57>. Accessed 24 May 2024.
3. “S2014/Lc3/Bubblesort.asm at Master · Oc-Cs360/S2014.” GitHub, <https://github.com/oc-cs360/s2014/blob/master/lc3/bubblesort.asm>. Accessed 25 May 2024.

## Flow chart or pseudo-code.

For this program, we chose to create a flowchart to show our guideline. The image of the flowchart is located on the next page.

