# CIS11 Course Project Part 2: LC-3 Program

In Part 2, we will create an LC-3 program that apply learned concepts in the course. This project is designed to capture the primary concepts of the course, which consist of computer architecture, processing and assembly programming. The program must fulfill the technical requirements. Refer to below project requirements.

## Tasks:

1. Review program requirements, pseudocode (from part 1) and flowchart (from part 1).
2. Write an LC-3 program that meets the requirements based on the given project options.
3. Test and analyze the program outcome to assess program limitations, performance, efficiency and future improvements.
4. Submit LC-3 program (.asm file).

## Project Program Requirements:

### Option A: Bubble Sort

**What is Bubble Sort?**

Bubble Sort is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of Ο(n2) where n is the number of items.

Bubble sort starts with very first two elements, comparing them to check which one is greater.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 14 | 33 | 27 | 35 | 10 |

In this case, value 33 is greater than 14, so it is already in sorted locations. Next, we compare 33 with 27.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 14 | 33 | 27 | 35 | 10 |

We find that 27 is smaller than 33 and these two values must be swapped.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 14 | 27 | 33 | 35 | 10 |

Next we compare 33 and 35. We find that both are in already sorted positions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 14 | 27 | 33 | 35 | 10 |

It continues to compare the next pair, exchange positions until the values are listed by ascending order.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 14 | 27 | 33 | 35 |

**Program:**

Create an LC-3 program that implements Bubble Sort for the following values:

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

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

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

1. Contain appropriate addresses: origination, fill, array, input and output. (20 points)
2. Display sorted values in console. (20 points)
3. Use appropriate labels and comments. (20 points)
4. Contain appropriate instructions for arithmetic, data movement and conditional operations. (40 points)
5. Comprise of 2 or more subroutines and implement subroutine calls. (20 points)
6. Use branching for control: conditional and iterative. (30 points)
7. Manage overflow and storage allocation. (20 points)
8. Manage stack: include PUSH-POP operation on stack. (20 points)
9. Include save-restore operations. (30 points)
10. Include pointer (20 points)
11. Implement ASCII conversion operations (30 points)
12. Use appropriate system call directives. (10 point)
13. Testing (20 points): Test the program using the below values (green).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 8 | 2 | 17 | 6 | 4 | 3 | 21 |

The result of the sort needs to be in the ascending order (small to high).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 6 | 8 | 11 | 17 | 21 |

Building and testing the program is for Project Part 3 (Total 300 points).

### Option B: Test Score Calculator

**Program**

Create an LC-3 program that displays the *minimum, maximum and average grade* of 5 test scores and display the letter grade associated with the test scores.

**Input:** User is prompt to input the test scores.

**Output:** Display maximum, minimum, average score and letter grade equivalence (0 – 50 = F, 60 – 69 = D, 70 – 79 = C, 80 – 89 = B, 90 – 100 = A) on the console.

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

1. Contain appropriate addresses: origination, fill, array, input and output. (20 points)
2. Display minimum, max, average values/grades in console. (20 points)
3. Use appropriate labels and comments. (20 points)
4. Contain appropriate instructions for arithmetic, data movement and conditional operations. (40 points)
5. Comprise of 2 or more subroutines and implement subroutine calls. (20 points)
6. Use branching for control: conditional and iterative. (30 points)
7. Manage overflow and storage allocation. (20 points)
8. Manage stack: include PUSH-POP operation on stack. (20 points)
9. Include save-restore operations. (30 points)
10. Include pointer (20 points)
11. Implement ASCII conversion operations (30 points)
12. Use appropriate system call directives. (10 point)
13. Testing (20 points): Test the program using the below values

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 52 | 87 | 96 | 79 | 61 |

Building and testing the program is for Project Part 3 (Total 300 points).

### Option C: Character Counter for Names

**Program**

Create an LC-3 program that displays the frequency of characters of user full name.

**Input:** User is prompt to input his or her full name.

**Output:** Numerical value the represent the frequency of a character in the user fullname.

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

1. Contain appropriate addresses: origination, fill, array, input and output. (20 points)
2. Display counted values in console. (20 points)
3. Use appropriate labels and comments. (20 points)
4. Contain appropriate instructions for arithmetic, data movement and conditional operations. (40 points)
5. Comprise of 2 or more subroutines and implement subroutine calls. (20 points)
6. Use branching for control: conditional and iterative. (30 points)
7. Manage overflow and storage allocation. (20 points)
8. Manage stack: include PUSH-POP operation on stack. (20 points)
9. Include save-restore operations. (30 points)
10. Include pointer (20 points)
11. Implement ASCII conversion operations (30 points)
12. Use appropriate system call directives. (10 point)
13. Testing (20 points): Test the program using the team names.

Building and testing the program is for Project Part 3 (Total 300 points).