CIS 11 Course Project
Part 1: Documenting the Project

Group E:
Grace Kim
Mehak Lohchan

Introduction

1.1 Purpose

The Test Score Calculator project focuses on containing appropriate addresses, which include origination, and the use of fill, array, input, and output. These addresses will be essential because they will allow the manipulation of data within the program. The primary objective of this program is to display the minimum, maximum, and average grade of five test scores as well as the letter grades associated with the test scores. This will be done by deriving arithmetic operations for each test score with a set of instructions that will allow the program to perform calculations and manipulate data. In addition, overflow and storage allocations must be handled to ensure that the data does not exceed the capacity of allocated memory, while efficiently allocating storage space. To promote modularity, the program will consist of subroutines as well as branching for control, in both conditional and iterative contexts. The program is also expected to manage the stack, which will include the implementation of push and pop operations. This is crucial for managing program execution flow. Save-restore operations will be included to ensure data integrity as well as preventing any errors within the program. The inclusion of pointers will give access to data by holding memory locations. The program will include ASCII conversion operations which allow for the conversion of characters or strings to their corresponding values. System-call directives will be implemented to enable the program to interact with the operating system and perform tasks. Finally, to validate the program's functionality, testing is important.

1.2 Intended Audience and Users

The intended audience and users are towards instructors and students.

1.3 Product Scope

The intention of this program is to showcase proficiency in various programming concepts and techniques which include handling addresses, data manipulation, control flow, subroutine implementation, managing a stack, overflow and storage allocation management, save-restore operations, pointer usage, ASCII conversion, and system call directives. By fulfilling these criteria, the Test Score Calculator program will demonstrate the ability to handle and process data efficiently, perform calculations, make decisions based on the programs required tasks, utilize pointers for efficient data access, and manage memory and storage efficiently. The overall intention of this program is to create a functional program that fulfills the criteria and objectives of the Test Score Calculator program.

1.4 Reference

Source Document for the Program Requirements and Specification

The specifications of the desired input and output of this program are the following:

a. For the input, the user is prompted to input the test scores, which will store this information and use it to display the output.
b. The output will consist of displaying the minimum, maximum, and the average scores and letter grade equivalence onto the console.

2. Overall Description

2.1 Product Perspective

The product is a program developed by LC3 assembly language that is designed to calculate and display the minimum, maximum, and average grade of five test scores with the associated letter grade of the score. The program will aid the user in providing an efficient solution for calculating and showcasing test score statistics.

LC3 supports numeric and character data types, so the program will use these types to efficiently compute and display the statistics. Numeric is used in order to store and process the test scores. Character data type used in order to store and display the letter grade that is associated with each test score.

2.2 Product Functions

The program's primary function is to calculate and display the minimum, maximum, and average grade of five test scores along with its associated letter grade.

The main task is to input and output data.

- In inputting data, the user is prompted to enter five test scores. The program will receive and store the test scores in its memory component.
- In outputting data, the program will display the minimum, maximum, and average grade along with the letter grade on the output screen otherwise known as the console.

There are four different subtasks.

- Minimum Calculation
    - In this subtask, a variable will be assigned to hold the lowest score. This will initially be any given value out of the five test scores. Each test score will be compared to the current score and update the score when an even lower score is found.
- Maximum Calculation
    - In this subtask, a variable will be assigned to hold the highest score. This will initially be any given value out of the five test scores. Each score will be compared to the current score and update the score when an even higher score is found.
- Average Calculation
    - In this subtask, the sum of all test scores will be calculated and stored into a variable. The variable that holds the sum will then be divided by five in order to calculate the average.
- Letter Grade
    - In this subtask, grade ranges will be defined. This will be used in order to determine the letter grade that will be associated with each test score. Once determined, a letter grade will be assigned to the score.

<u>Technical Functionality</u>

- Input Subroutine
    - Users are prompted to enter test scores into the program. Test scores are read and stored.
- Minimum Subroutine
    - LDI will be used to load one of the test scores for comparison to the others in order to determine minimum test score.

    - A loop will be used to compare the test scores.
    - LDR will be used to load the test score from memory.
    - ADD and BR will be used to update the minimum grade after comparing if needed.

- Maximum Subroutine
    - LDI will be used to load one of the test scores for comparison to the others in order to determine maximum test score.
    - A loop will be used to compare the test scores.
    - LDR will be used to load the test score from memory.
    - ADD and BR will be used to update the maximum grade after comparing if needed.
- Average Subroutine
    - LDI will be used to initialize the sum of test scores.
    - A loop is used to iterate through test scores to add the next test score to the sum variable.
    - LDR will be used to load a single test score to start the sum variable.
    - ADD will be used to add the current test score to the sum variable.
    - BR will be used to branch back to the loop for more test scores to be added to the sum variable.
    - DIV will be used to divide the sum variable by 5 (total test scores).
- Letter Grade Subroutine
    - BR will be used to compare current test scores with implemented grade ranges.
- Output Subroutine
    - Minimum, maximum, average scores along with letter grades will be outputted onto the console.

2.3 User Classes and Characteristics

- Business Personnel

    People like Curriculum Developers or Education Administrators will be intrigued in this program as it would monitor academic performance and help to create certain decisions based on the data they get from the program. This program could also be used by these business personnel to analyze test scores to assess the effectiveness of a certain instructional material, or methods. Tasks from them would include the assessment of the program's output to gain insight on academic performance and make informed decisions based on statistics that the program will provide.

- Technical Personnel

  People like software developers will be involved in the development of the program. Their tasks are to create, debug and maintain the program. They will write all the code individually or as a team and perform tests to ensure the program runs smoothly. Maintaining the program over time is also done by the development team.

2.4 Operating Environment

The system that the application will be operated on can be any system that is able to support LC3 such as Windows, Linux, and MacOS.

Any operating system that consists of a web application to open the LC3 program can be used.

The development platform used to develop and run the LC3 program is LC3 Simulator. We used this in order to write, assemble, and execute the program.

To operate the program, the latest version of LC3 simulator should be used for optimal performance.

Application Usage:
Users must install the LC3 Simulator onto their operating device. It should work for most systems (Windows, Linux, and MacOS). Then, download the Test Score Calculation Program and save it in order to deliver the file into the LC3 Simulator. Once delivered into the simulator, assemble the program in the simulator and run it. The program should execute properly. Users will be prompted to enter the test scores. When finished, scores with letter grades will be output, completing the execution.

2.5 Design and Implementation Constraints

This program is limited in its capabilities as it does use LC3 which is for assembly level programming. It has a limited memory size as well as a very restricted instruction set.

Limitations that users will experience is the inability to calculate a number of test scores any less or greater than 5 as this program can only handle exactly 5 test scores. Grade ranges are also strictly in place. They are hardcoded into the program, so modified grading ranges cannot be done.

2.6 Assumptions and Dependencies

This application is dependent on a network connection and a browser. It does not require additional applications for compilation, other than what was listed. If this application were to be tested on WIndows, then an LC-3 Simulator application could be downloaded which would open up an additional simulator to work with each register of the program. However, downloading this application is not required as it can be simply tested on the LC-3 simulation website using a browser.

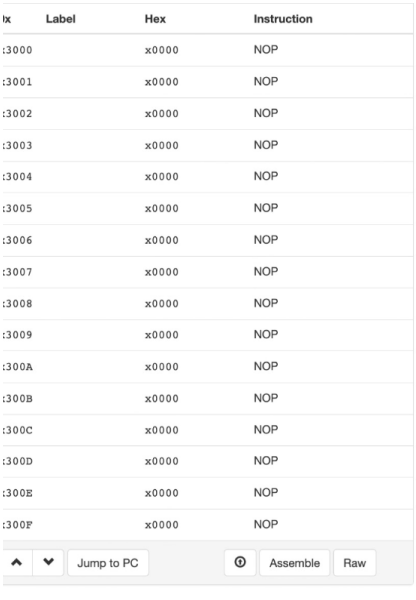
The program is also independent of any exetermal services. This is a contained program that assumes test scores will be provided by the user. It depends on the user for input as it cannot source material or data from outside sources.

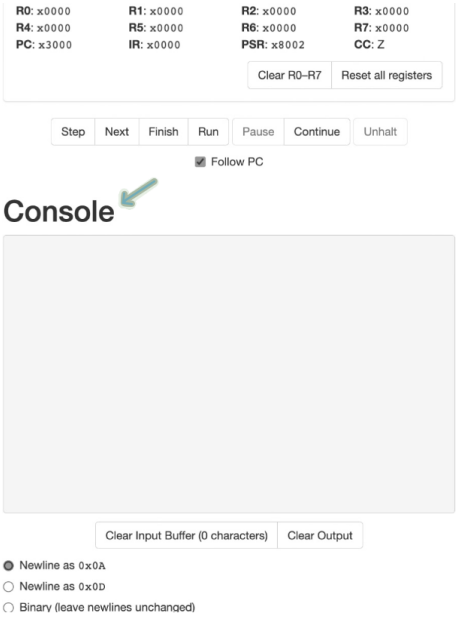
The program assumes that the user has enough memory in their system in order to store the LC3's data.

3. External Interface Requirements

3.1 User Interfaces

The user will interface with the program using the console on the LC-3 website. The output will be displayed onto the console where the user will be allowed to input a value or a character dependent on the output.





3.2 Hardware Interfaces

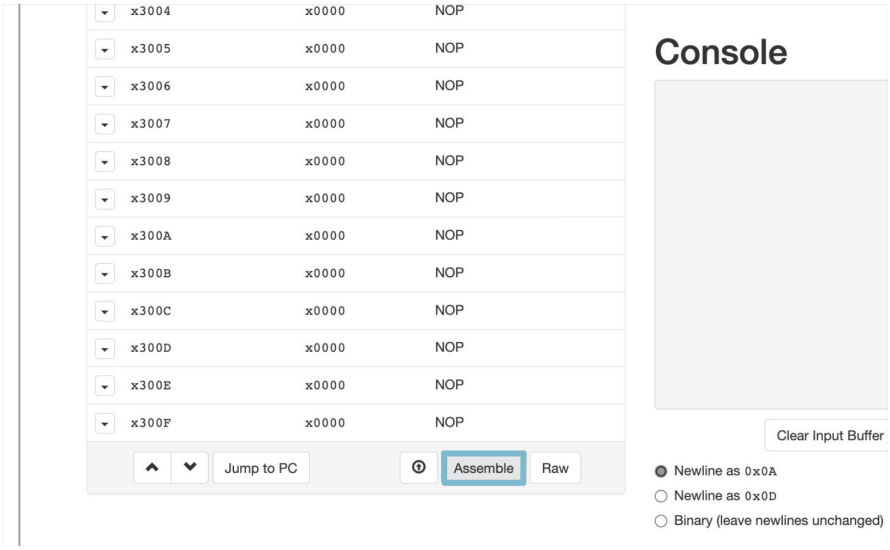
The hardware interface that will be included is a computer mouse if the user were to be testing the program on a PC. Devices such as a keyboard and a mouse can be used for input. Whereas a device such as a monitor is used to display the output to the screen. For the user to be able to test the program, their keyboard will need to be attached to a PC, or the user will need to be using a laptop. This program can run on any type of computer system, including Windows, Linux, macOS.

3.3 Software Interfaces

Within the LC-3 website, there are many components that the software interface includes to be able to perform operations and function. For example, the user interfaces are visual and interactive components of a software that allow the user to interact with the system. These components consist of clicking a button or navigation elements that allow users to move between different sections/functionalities of the application. Input validation is also an important component of the software interface as it allows the user to enter and submit data which will allow the user to navigate through the web application. For example, on the LC-3 website, the user will first click the assemble button to enter the program, then press enter to load the program onto the simulator and set any values (if needed), and lastly click run to display the output.

3.4 Communications Interface

This application will require web, Internet, and network connectivity. Given that the program will be using a website simulator, these three are essential to ensure that the program is able to compile properly. This program will be able to run on all browsers such as Internet Explorer, Google Chrome, Safari, Firefox, and websites similar to these. Since the program requires a network connection and will be using a website simulator, the network connection needed is Wi-Fi, mobile, and the web. To be able to test the program, the user will need to first assemble the program using the assemble button as shown as below.



4. Detailed Description of Functional Requirements

4.1 Type of Requirement (summarizing from section 2.2)

The functional requirements of this program is to calculate and showcase the minimum, maximum, and average grade with the associated letter grade to the user. Its purpose is to provide a statistic given five test scores. The inputs for this program will be the five test scores from the user, and it will output the minimum, maximum, and average grade with their respective letter grade. The data from this program are the test scores, minimum, maximum, and average grade. It is all stored in the internal memory of the LC3 program.

4.2 Performance requirements

The program should be efficient in execution and provide the correct, calculated statistics within a reasonable time (milliseconds). It should also not take up much given the small dataset and minimal calculation involved. The performance of this program aligns with the capabilities of LC3. The only performance requirement from the operating system is that it can run LC3. This means any computer that has a web browser will be able to utilize this program efficiently.

4.3 Pseudocode

```
.ORIG x3000                ; Begin

; Allocate memory locations for storing the test scores
.ORIG x4000

SCORES      .BLKW 5        ; Store 5 test scores

; Prompt user to input the test scores
            LEA R0 , PROMPT         ; Load address of the prompt message
            PUTS                    ; Output the message
            LEA R1, SCORES          ; Load address
            LEA R2, SCORE_PROMPT; Load address
            LD R3, #5               ; Initialize counter

LOOP        ADD R3, R3, #-1     ; Decrement counter
            BRz CALCULATE       ; If counter is zero, go to calculation
            IN                  ; Read input
            STR R0, R1, #0      ; Storing test scores
            ADD R1, R1, #1      ; Increment memory address
            PUTS                ; Output message
            BRnzp LOOP          ; Continue the loop

; Calculation for the minimum, maximum, and average test scores
CALCULATE       LEA R1, SCORES  ; Load address
                LDR R2, R1, #0  ; Load 1st score as the min and max
                LD R3, #0       ; Initialize sum to 0
                LD R4, #0       ; Counter
LOOP_MINMAX     ADD R4, R4, #1  ; Increment counter
                LDR R0, R1, #0  ; Load current score
                ADD R1, R1, #1  ; Increment memory address

; Calculate minimum test score
ADD R5, R0, R2                  ; Add current score to min score
BRp SKIP_MIN                    ; If result is pos, skip updating min score
AND R2, R0, #0                  ; Clear register
ADD R2, R2, R0                  ; Add current score to min score
SKIP_MIN
```

```
; Calculate maximum test score
ADD R5, R0, R3                  ; Add current score to max score
BRn SKIP_MAX                    ; If result is neg, skip updating max score
AND R3, R0, #0                  ; Clear register
ADD R3, R3, R0                  ; Add current score to max score
SKIP_MAX

; Calculate sum of the test scores
        ADD R3, R3, R0                  ; Add current score to sum
        ADD R4, R4, #-1                 ; Decrement the counter for remaining
        BRnzp LOOP_MINMAX               ; Continue looping

; Calculate average
        ADD R0, R3, #0                  ; Move sum to R0
        ADD R1, R4, #0                  ; Move number of scores to R1

        BRnz DIV_LOOP                   ; If # of scores is not 0, go into loop
        BRnzp AVG                       ; If # of scores is 0, average is found

DIV_LOOP                        ; Loop to divide the sum by 5
        DIV R0, R0, R1          ; Divide sum by 5
        ADD R1, R1, #-1         ; Decrement number of scores
        BRp DIV_LOOP            ; R1 is pos, continue loop
        BRz DIV_LOOP            ; R1 is zero, continue loop

AVG     ADD R0, R0, #-10        ; Adjust R0 for ASCII conversion
        ADD R0, R0, #48         ; Convert R0 to ASCII character
        LD R4, ASCII_AVG        ; Load address of ASCII_AVG
        ST R0, R4, #0           ; Store letter in the variable

; Display minimum, maximum, and average scores
LEA R0, RESULT_MIN              ; Load address
PUTS                            ; Output message
LD R0, R2                       ; Load minimum score
OUT                             ; Output the minimum score
LEA R0, RESULT_MAX              ; Load address
PUTS                            ; Output message
LD R0, R3                       ; Load the maximum score
OUT                             ; Output the maximum score
LEA R0, RESULT_AVG              ; Load address
```

```
PUTS                                ; Output message
LD R1, ASCII_AVG                    ; Load address
LD RO, R1, #0                       ; Load address
OUT                                 ; Output the average score

; Calculate letter grade
LEA R1, SCORES                      ; Load address
LEA R2, LETTER                      ; Load address
LD R3, #5                           ; Counter

LOOP_GRADE      LDR R0, R1, #0      ; Load current score
                ADD R1, R1, #1      ; Increment memory address

; Letter grade
ADD R4, R0, #60                     ; Compare with ASCII
BRnzp GRADE_A                       ; If pos or 0, branch to GRADE_A
ADD R4, R0, #-70                    ; Compare with ASCII
BRnp GRADE_F                        ; If neg or 0, branch to GRADE_F
ADD R4, R0, #-65                    ; Adjust range
ADD R4, R4, #7                      ; Get ASCII value

GRADE_A         ADD R4, R4, #-10    ; Adjust ASCII value range
                ADD R4, R4, #48     ; Convert ASCII to character
                STR R4, R2, #0      ; Decrement counter
                ADD R3, R3, #-1     ; Not zero, Repeat and loop
                BRnzp LOOP_GRADE

LEA R0, RESULT_GRADE                ; Display letter grade
PUTS                                ; Output message
LD R2, LETTER                       ; Load address
LD R0, R2, #0                       ; Load address
OUT                                 ; Output

HALT                                ; Halt the program

PROMPT          .STRINGZ “Enter 5 test scores: “
SCORE_PROMPT    .STRINGZ “Enter in a test score:  “
RESULT_MIN      .STRINGZ “Minimum Test Score: “
RESULT_MAX      .STRINGZ “Maximum Test Score:  “
RESULT_AVG      .STRINGZ “Average Test Score: “
```

```
RESULT_GRADE    .STRINGZ “Letter Grade: “
ASCII_AVG       .BLKW 1
LETTER          .BLKW 5

.END            ; End of program
```