CIS-11 Project Documentation Template

**Team Push N Pop**

**Tawny Hoemke, Gilbert Sosa**

**Test Score Calculator**

**5/19/2024**

**Advisor: Kasey Nguyen, PhD**

# Part I – Application Overview

## Objectives

### 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 prompted to input the test scores.

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



#### Why are we doing this?

* **What business objectives of the company will this project help achieve?** This project will help simplify the workflow of teachers, professors, and any personnel that is required to enter a grade.
* **Why are we doing this project now? What will happen if we do it later? What if we do not do it at all?** We are doing this project to build our skills in assembly/ LC3.If we do this later or not at all, our grade would be negatively affected.
* **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?** Professors and teachers will benefit from this project. This is an important improvement because it minimizes their workload by automating the processes for calculating the average, minimum, and maximum test scores.

## Business Process

### Previously, professors would have to manually calculate averages for lecture halls filled with hundreds of students, after having to grade each of their tests.

### With this program, professors will no longer have to worry about crunching those numbers. By simply inputting numbers into this calculator, an average will immediately be produced.

## User Roles and Responsibilities

### Professors must first grade their class' test. After having them all graded, they are to input their values into this calculator.

## Terminology

### LC3: Little Computer 3

Assembly: A low level program

Array: A collection of values

ASCII: American Standard Code for Information Interchange

Binary: A decimal value

Register: A storage location

Pointer: Stores memory address location

# Part II – Functional Requirements

## Statement of Functionality

### Prompt the user to input five grades. Each 3-digit grade will be inputted one digit at a time.

### The program converts each digit from the ASCII character value to its appropriate binary value.

### Once converted, the first digit of the grade is multiplied by 100, and the second digit is multiplied by 10.

### After this multiplication, all three digits are added together and stored into a register.

### The grade is then placed into an array. Each time a grade is placed into the array, a pointer is used to move to the next location of the array.

* The program finds the maximum, minimum, average and its associated letter grade.
* Maximum, minimum, average and associated grades are displayed.

## Scope

* PROMPT1: Gives directions on how to input digits.
* PROMPT2: Prompts input of each digit of grade.
* TSCORES: Array.
* \MULT: performs appropriate multiplication on first two digits.
* CHECKMIN: Determines minimum grade.
* CHECKMAX: Determines maximum grade.
* AVGSUM: Determines average grade.
* DISPLAYALL: Outputs minimum, maximum, average, and letter grades

## Technical Requirements

* Contain appropriate addresses.
* Display minimum, maximum, average, and associated letter grades on console.
* Use appropriate comments and labels.
* Contain appropriate instructions for arithmetic, data movement and conditional operations.
* Comprise of two or more subroutines and implement subroutine calls.
* Use branching for control: conditional and iterative.
* Manage overflow and storage allocation.
* Manage stack.
* Include save-restore operations.
* Include pointer.
* Implement ASCII conversion operations.
* Use appropriate system call directives.
* Test program.

### 

# Part III – Appendices

## Flow chart or pseudo-code.

Include branching, iteration, subroutines/functions in flowchart or pseudocode.

Code is to implement:

Array with each digit; every 3 elements is one test score

Pointer pointing to array elements

Stack may be used in place of array; appropriate PUSH POP subroutines will be made

***Input/ Output***

**LEA** R0, IN\_PROMPT ;Output prompt to enter in test score

**PUTS** ;Display PROMPT

**LEA** R1 TSCORES ;R1 Pointer for test scores array

**AND** R5, R5, #0 ;R5 is loop counter

**PROMT LOOP {**

PROMPTING USER

;//////////////////////////////////////////////////////////////////

;// Getting Digits //

;/////////////////////////////////////////////////////////////////

;//Let R0 be used for prompting user and accepting input

;//Let R1, R2, and R3 be used for multiplication

;//R2 will be used to multiply digits 1 and 2 by 100 and 10, respectively

;//Get digit1, multiply by 100

LEA R0, PROMPT1

GETC ;//Get ASCII Value (Character)

ST R0, DIGIT1 ;//R0 = ASCII for DIGIT1

R0 -= 48 ;//Convert to Binary

ADD R1, R0, #0 ;//R1 = Digit 1

R2 = 100 ;//Multiple ADD lines

JSR MULT ;//R1 \* R2 = R3

R4 += R3

AND R0, R0, #0 ;//Clear R0

AND R2, R2, #0 ;//Clear R2

;//Get digit2, multiply by 10, add to R3

LEA R0, PROMT2

GETC

ST R0, DIGIT2

R0 -= 48

ADD R1, R0, #0

R2 = 10

JSR MULT

R4 += R3

AND R0, R0, #0

AND R2, R2, #0

;//Get digit3

LEA R0, PROMT3

GETC

ST R0, DIGIT3 ;//Store

R0 -= 48 ;//Convert to Binary

ADD R1, R0, #0 ;R1 = Digit3

R3 += R1 ;//R3 = Test Score

;//////////////////////////////////////////////////////////////////

;// Checking MIN MAX //

;//////////////////////////////////////////////////////////////////

LEA R1, MIN

JSR CHECKMIN

LEA R1, MAX

JSR CHECKMAX

R5--

if (R5 = 0)

BRz EXITLOOP ;//May have different name

}

TSCORES[R5] = R3

R5++

if (R5 = 5)

BR EXITLOOP ;//May have different name

}

;/////////////////////////////////////////////////////////////////////////////

;// Average Summing Loop //

;/////////////////////////////////////////////////////////////////////////////

R1 = &TSCORES1 ;//Pointer points at address of first array element

R2 = 0 ;//R2 is a counter

AVGSUM{

R3+= \*R1 ;// R3 = Sum

R1 += (Element size)

R2++

if (R2 != 5)

BR AVGSUM ;//Continue loop

}

JSR DIV ;//R4 = R3 / R2 = SUM/5

ST R4, AVG

;//////////////////////////////////////////////////////////////////

;// Display values //

;//////////////////////////////////////////////////////////////////

DISPLAYALL{

LEA R0, DISPLAYMIN

PUTS

LEA R0, TSCORES[Min element]

PUTS

R3 = MIN

JSR GIVELETTER

LEA R0, DISPLAYMAX

PUTS

LEA R0, TSCORES[Max element]

PUTS

R3 = MAX

JSR GIVELETTER

LEA R0, DISPLAYAVG

PUTS

AVGTOASCII

LEA R0, AVGASCII

PUTS

R3 = AVG

JSR GIVELETTER

}

AVGTOASCII{

First Digit:

99/ 100

However many times 100 goes into 99:

DIGIT1 = 1

Second Digit:

99/10

Digit2 = 9

Digit 3 = Remainder

RET

}

**IN\_PROMPT** .STRINGZ “Enter test score 1 digit at a time. If score is not 100, enter 0

for first the digit.”

**PROMPT1** .STRINGZ “1st digit: “

**PROMPT2** .STRINGZ “2nd digit: “

**PROMPT3** .STRINGZ “3rd digit: “

SubRoutines

;//Let R3 have your inputted grade

;//Let R1 be pointer to current minimum MIN

**CHECKMIN**

;//Save OG values of R2, R1

ST R3, SAVER3

ST R1, SAVER1

if (R3 < \*R1)

ST R3, MIN

;///Restore values of R3, R4

LD R3, SAVER3

LD R1, SAVER1

RET

;//Let R3 have your inputted grade

;//Let R1 be current maximum MAX

**CHECKMAX**

;//Save OG values of R3, R1

ST R3, SAVER3

ST R1, SAVER4

if (R3 > \*R4)

ST R3, MAX

;///Restore values of R3, R4

LD R3, SAVERR3

LD R4, SAVERR4

RET

**DIV** ;R4 = R3/R2, R5 = R3%R2

ST R2, SAVE1

ST R3, SAVE2

AND R4, R4, x0

NOT R2, R2

ADD R2, R2, #1

DLOOP

AND R5, R5, x0

ADD R4, R4, #1

ADD R5, R3, x0

ADD R3, R3, R2

BRzp DLOOP

ADD R4, R4, #-1

RET

;//Let R3 contain grade

;//Let R0 contain appropriate letter grade

**GIVELETTER**

ST R0, SAVER0

ST R3, SAVER3

if (R3 - 59 <= 0)

R0 = ASCII for 'F'

Display R0

else if (R3 - 69 <= 0)

R0 = ASCII for 'D'

Display R0

else if (R3 - 79 <= 0)

R0 = ASCII for 'C'

Display R0

else if (R3 - 89 <= 0)

R0 = ASCII for 'B'

Display R0

else

R0 = ASCII for 'A'

Display R0

LD R0, SAVER0

ST R3, SAVER

RET

;//R1 \* R2 = R3

**MULT**

;//Save register values

ST R1, SAVER1

ST R2, SAVER2

ST R3, SAVER3

AND R3, R3, #0

MLOOP

ADD R3, R3, R1

ADD R2, R2, #-1

BRp MLOOP

LD R1, SAVER1

LD R2, SAVER2

LD R3, SAVER3

RET