### Raybert Salazar Project testing TestScoreCalculator.asm

### 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.

Input

52 , 87, 96, 79, 61

Output

(the letter grades for each test score in order (52, 87, 96, 79, 61)A screenshot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

The max, min and avg grade

A screenshot of a computer program

AI-generated content may be incorrect.

**Program Explanation: Test Score Calculator**

This LC-3 program calculates and displays the **minimum**, **maximum**, and **average** of **five user-input test scores**, along with their corresponding **letter grades** based on a fixed grading scale (A–F). The program is organized into modular subroutines for clarity, reusability, and maintainability.

**Program Structure**

The program is broken down into **clear, functional components**, each handling a specific task:

1. **Main Loop**
   * Iteratively reads 5 test scores from the user.
   * Stores each score in a memory array using a pointer-based method.
   * Accumulates the sum for later average computation.
   * Converts the score to a letter grade using CLASSIFY\_GRADE.
   * Displays the letter grade using OUT.
2. **READ\_3DIGIT\_NUMBER Subroutine**
   * Prompts and reads **three digits** as ASCII input.
   * Converts the ASCII digits into an integer (0–999).
   * Supports only valid numerical characters (no validation built-in).
3. **STORE\_BY\_INDEX Subroutine**
   * Uses the index and array base to store each score into a memory location.
   * Maintains and updates the index using a dedicated memory label.
4. **DIV & MULT Subroutines**
   * Performs division and multiplication without using native LC-3 support (since LC-3 has no DIV or MUL).
   * Implements **repeated addition/subtraction** for basic arithmetic.
5. **DISPLAY\_NUMBER Subroutine**
   * Converts an integer into its ASCII decimal representation.
   * Uses a **stack-based PUSH/POP** approach to reverse digit order before printing.
6. **FIND\_MIN\_MAX Subroutine**
   * Iterates through the array to find the **minimum** and **maximum** values using comparison and pointer logic.
7. **CLASSIFY\_GRADE Subroutine**
   * Maps numeric scores to ASCII letter grades based on score ranges (90–100 = A, ..., <60 = F).
8. **Stack Support (PUSH, POP, ISEMPTY)**
   * Implements a basic stack mechanism using register R6 to reverse numbers for output and support modular conversion operations.

**LC-3 Limitations**

The LC-3 architecture has some critical limitations that directly impact the structure and design of this program:

* **9-bit PC-relative addressing**:  
  Instructions like LD, ST, BR, and LEA can only access **±256** memory locations from the current instruction. This means:
  + All subroutines, data labels, and jump targets must be **placed close together**.
  + In this program, labels like X, Y, and XDIVY are defined near the main loop to prevent offset too large errors.
* **No hardware division or multiplication**:  
  The LC-3 does not support MUL or DIV. These had to be implemented using:
  + Repeated subtraction for division.
  + Repeated addition for multiplication.
* **No character validation or advanced input**:  
  The LC-3 supports basic GETC and OUT calls but has no string parsing or real input validation, which limits interactivity.
* **Limited general-purpose registers (R0–R7)**:  
  Careful save/restore operations are necessary in each subroutine to avoid register conflicts.