Lab4: Sakiko's Delivery System

1 Task

Sakiko was once the daughter of a wealthy family, but due to a family misfortune, her father became despondent and turned to alcohol. As a result, Sakiko had to start working as a delivery driver to support the household. She begins all deliveries from a restaurant located at coordinate $(0, 0)$ in a city with grid-like streets. Due to traffic regulations, she can only travel **right** (increasing the column coordinate) or **down** (increasing the row coordinate).

To improve delivery efficiency, Sakiko designed a system to compute a recommendation value for each potential order. Orders with higher recommendation values should be prioritized. The recommendation is calculated as follows:

1. Sakiko always starts from $(0, 0)$.
2. She may only move right (increase column index) or down (increase row index).
3. $Steps(i, j)$ is the minimum number of steps to reach (i, j).
4. $Routes(i, j)$ is the total number of unique routes to reach (i, j).
5. A final recommendation value combines route flexibility (based on $Routes(i, j)$) and travel cost (based on $Steps(i, j)$).
6. Your task is to compute the recommendation value for the destination (N, M).

For any location (i, j):

$$Steps(i, j) = i + j, \quad i \geq 0, j \geq 0 \tag{1}$$

$$Routes(i, j) = \begin{cases} 1, & i = 0 \text{ or } j = 0 \\ Routes(i-1, j) + Routes(i, j-1), & i > 0 \text{ and } j > 0 \end{cases} \tag{2}$$

$$Recommendation(i, j) = Routes(i, j) \times 5 - Steps(i, j) \tag{3}$$

The higher the recommendation value, the more worthwhile the order is to take. A lower value suggests avoiding it.

2 Your Job

The input coordinates N $(0 \leq N \leq 5)$ and M $(0 \leq M \leq 5)$ are stored in memory at addresses `x3100` and `x3101` respectively.

You must implement a **recursive** LC-3 assembly program to calculate the recommendation value for (N, M) and store the result at memory location `x3200`.

Assume all general-purpose registers (R0–R7) are initialized to zero. Your program begins at `x3000`.

For your convenience, your code may be written as:

```
.ORIG x3000
    LDI R0, INPUT_N      ; Load N into R0
    LDI R1, INPUT_M      ; Load M into R1
    ; Your code starts here
    ; ...
    ; ...
    ; Your code ends here
    TRAP x25             ; Halt program

STACK    .FILL x6000     ; Starting address of the stack
INPUT_N  .FILL x3100     ; Address for input N
INPUT_M  .FILL x3101     ; Address for input M
RESULT   .FILL x3200     ; Address to store the result
.END
```

3 Grading

Correctness for 50% and the report for other 50%.

4 Submission

Here are some notifications:

- Write your program in **LC-3 assembly**.
- Begin with `.ORIG x3000` and end with `.END`.
- The last instruction must be `TRAP x25` (HALT).
- Include **comments** in your code where necessary for clarification.

Submit a single ZIP file named `StudentID_Name.zip`:

```
PB********_Name.zip
|
+-- PB********_Name_report.pdf
|
+-- lab4.asm
```

5 Report

Your report should include:

- Purpose.
- Process: Development steps, encountered bugs, and solutions.
- Results: Demonstrate correctness with test outputs.
- Discussions: Answer the discussion questions below.

6 Discussion Questions

- It is easy to see that this program can be rewritten as a simple iterative program, and the iterative method seems like more efficient compared to the basic recursive method. Why?
- Based on the reasons you mentioned, how can you improve the efficiency of this program (no code required)?