

Lab sheet 05

Title: Exercises for Comparison and Loop in ARM Assembly

Aims:

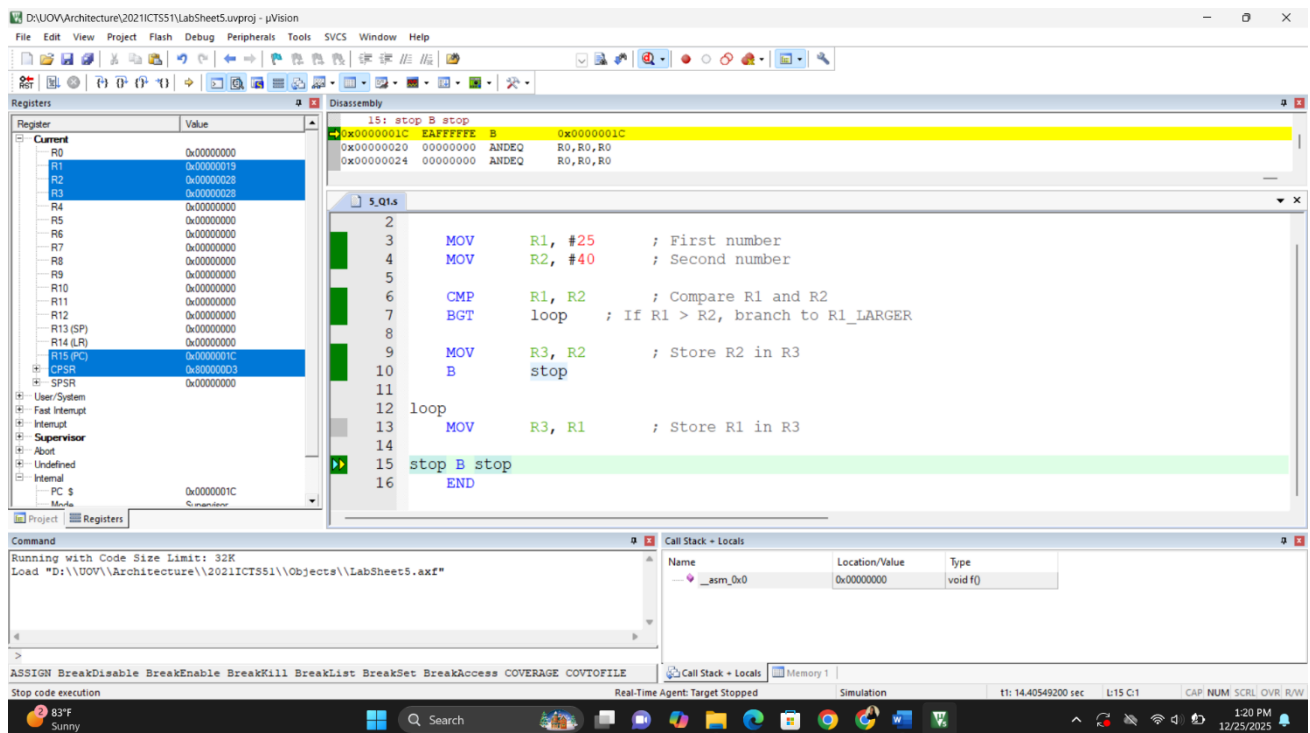
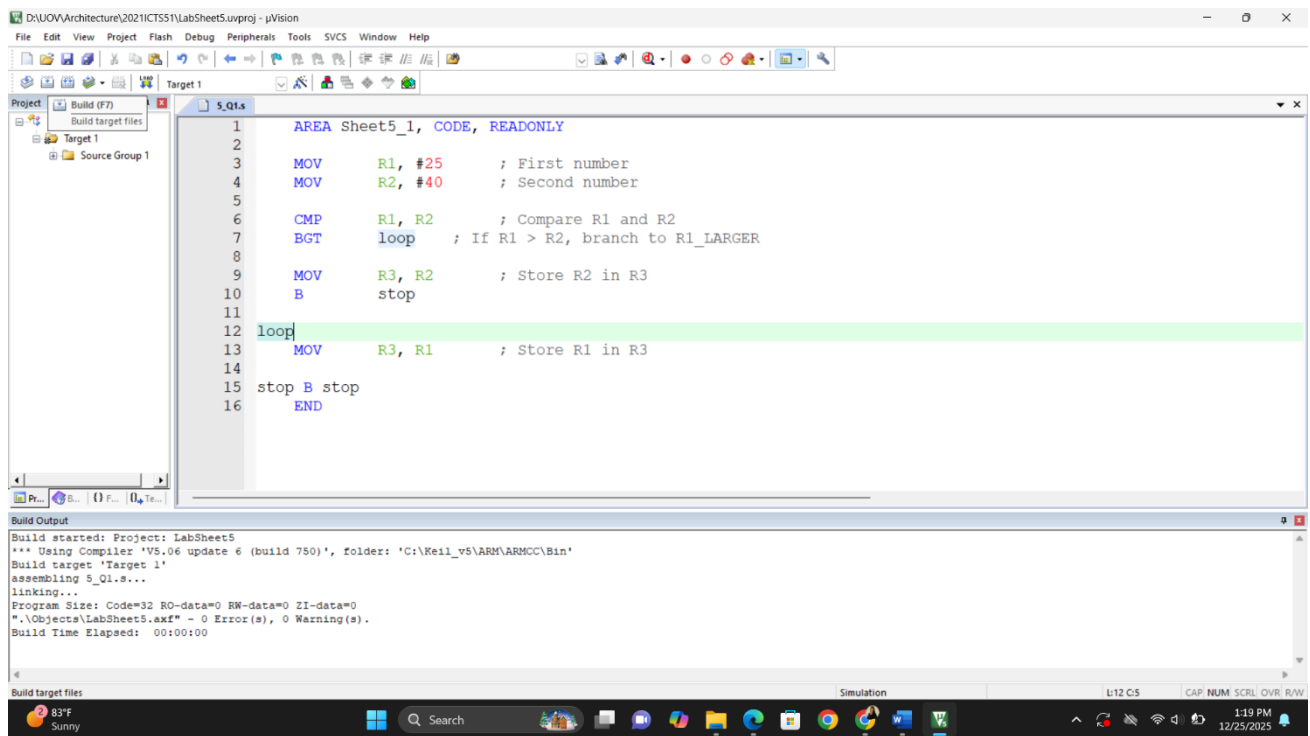
1. To understand how comparison operations are performed using ARM Assembly instructions.
2. To learn how loops are created using conditional and unconditional branch instructions.
3. To develop skills in writing, executing, and debugging loop-based and comparison- based ARM Assembly programs in Keil μ Vision.

Tasks:

1. Identify and explain comparison and branch instructions such as CMP, BEQ, BNE, BGT, BLT, and B used in ARM Assembly for decision-making and looping.
2. Analyze the program flow by tracing each step to understand:
 - a. How comparisons affect branching
 - b. How loops repeat based on conditions
 - c. How ARM uses condition flags to control program execution

Activities:

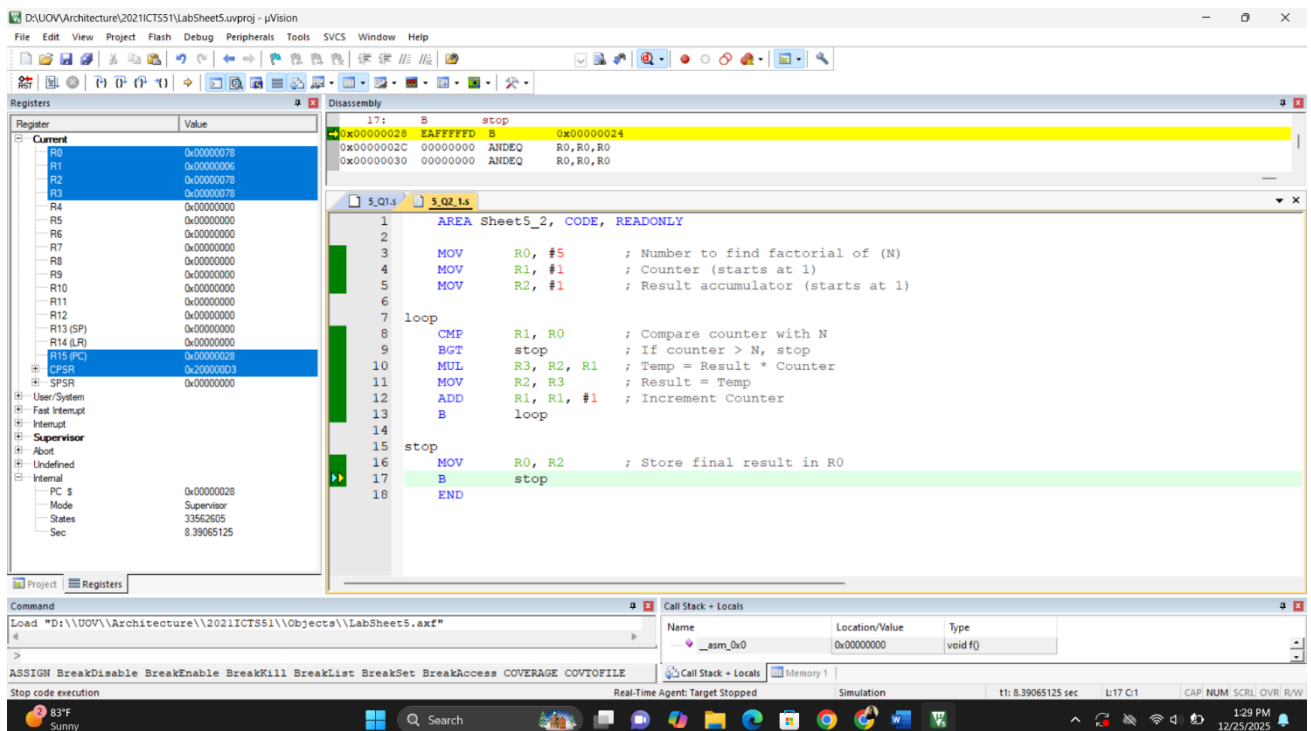
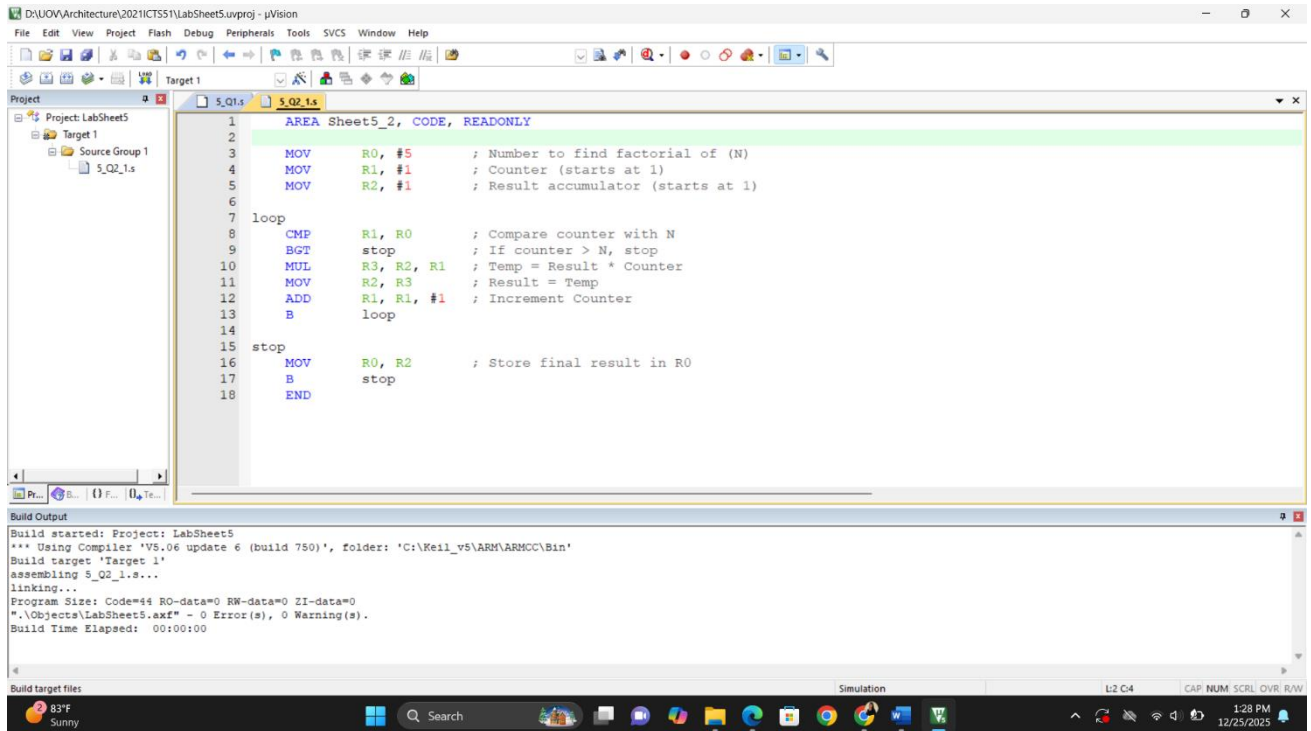
- 1) find the larger number between two numbers and store the output in R3 register



2) Write a program to find the factorial of a number in each way.

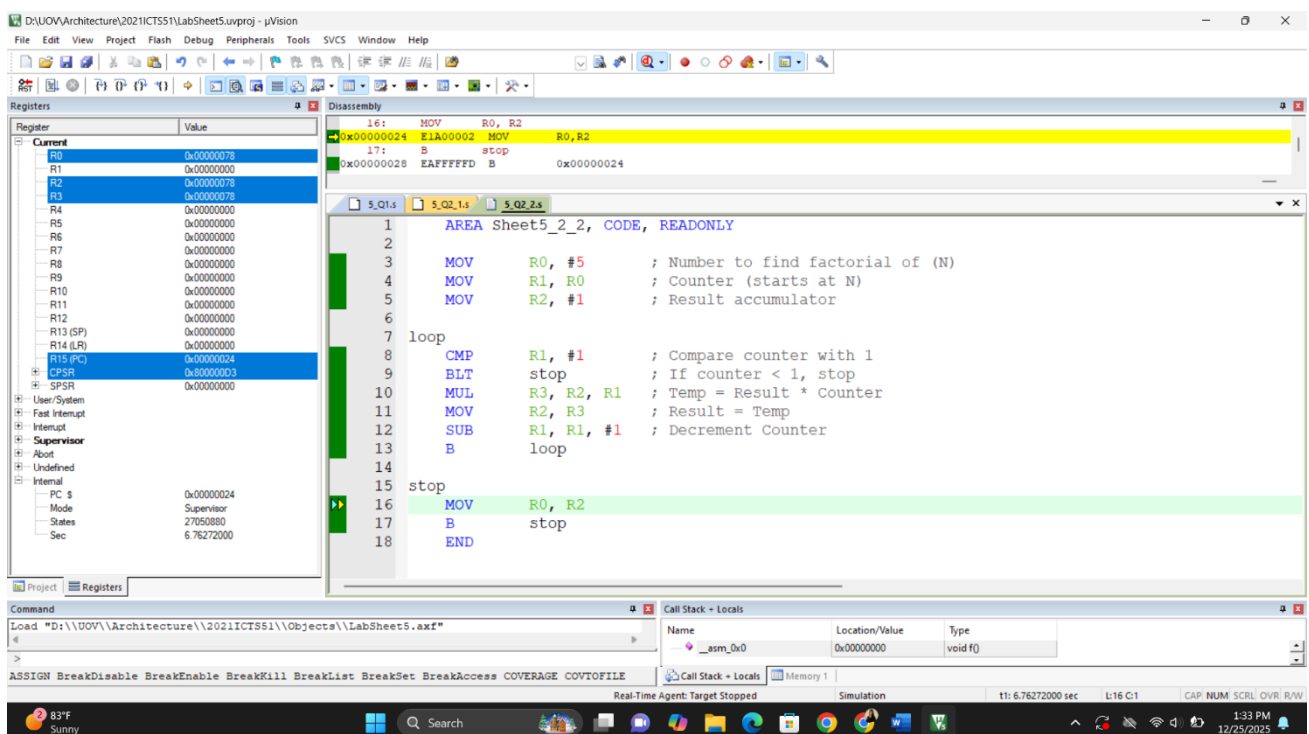
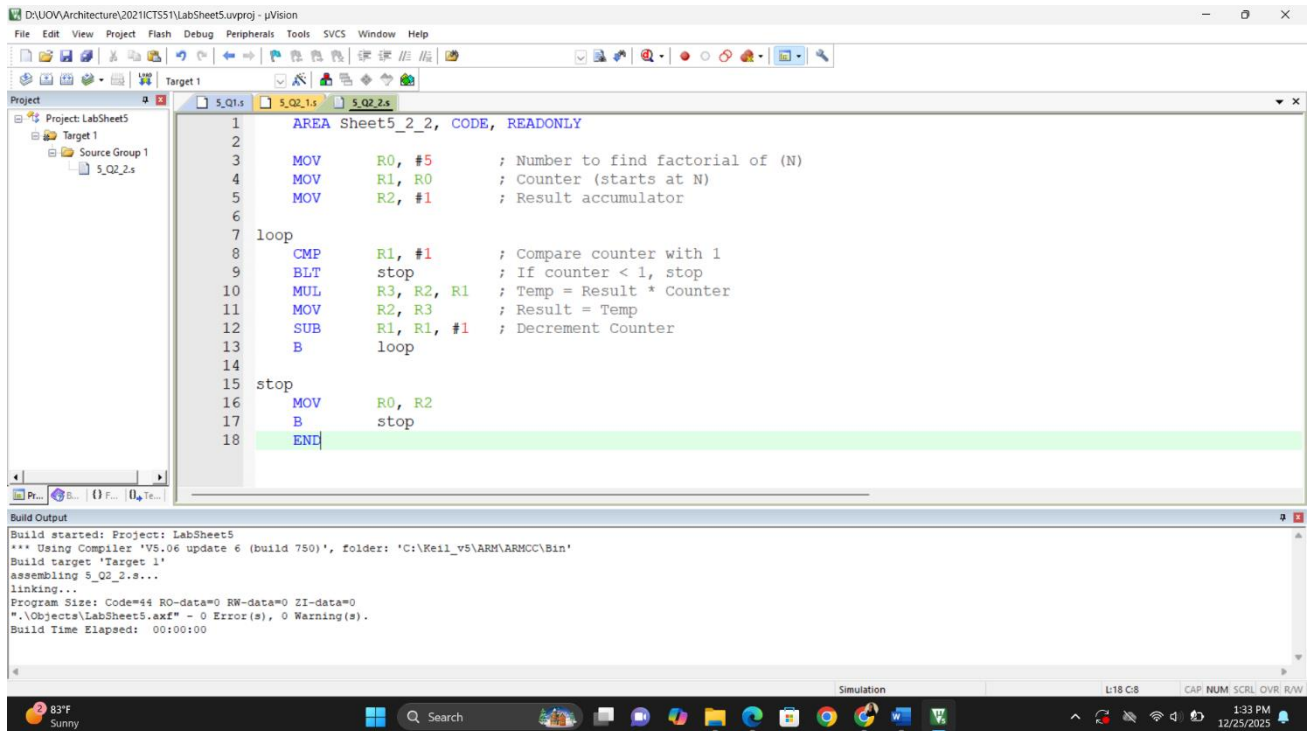
- Increment

Hint: $5! = 1 * 2 * 3 * 4 * 5$

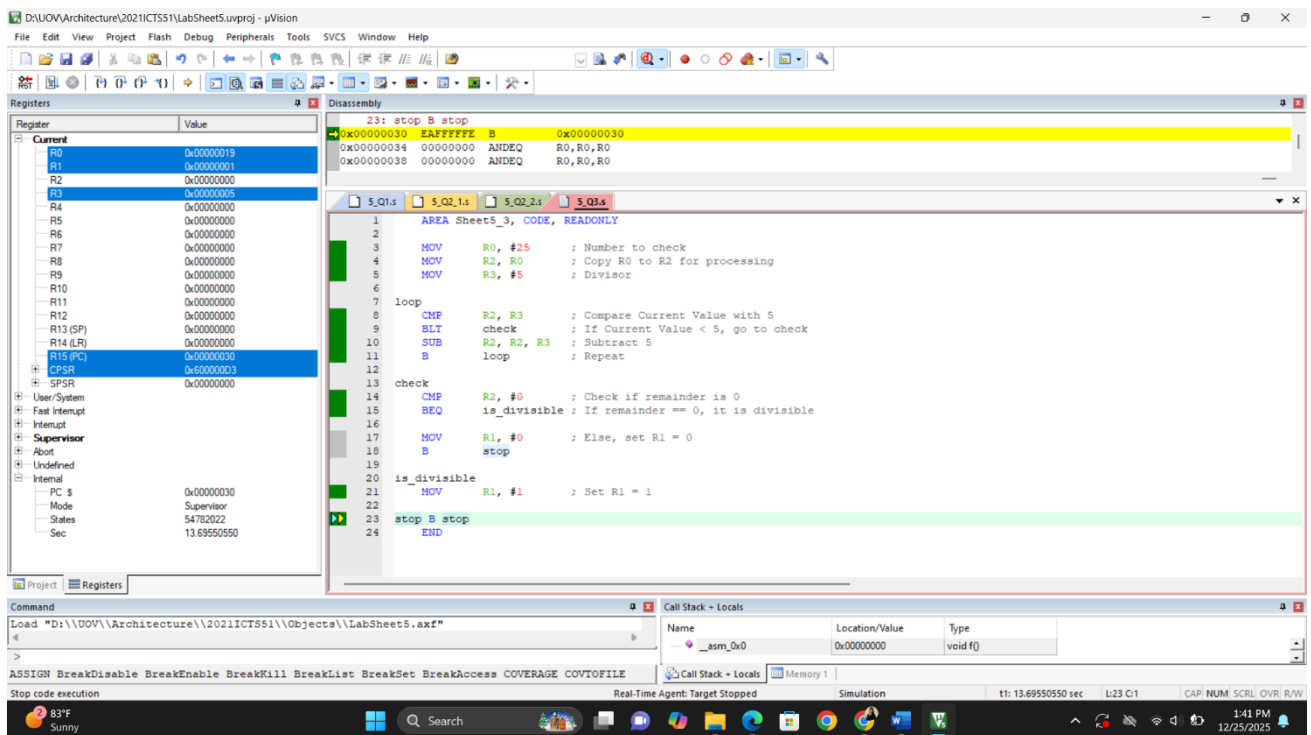
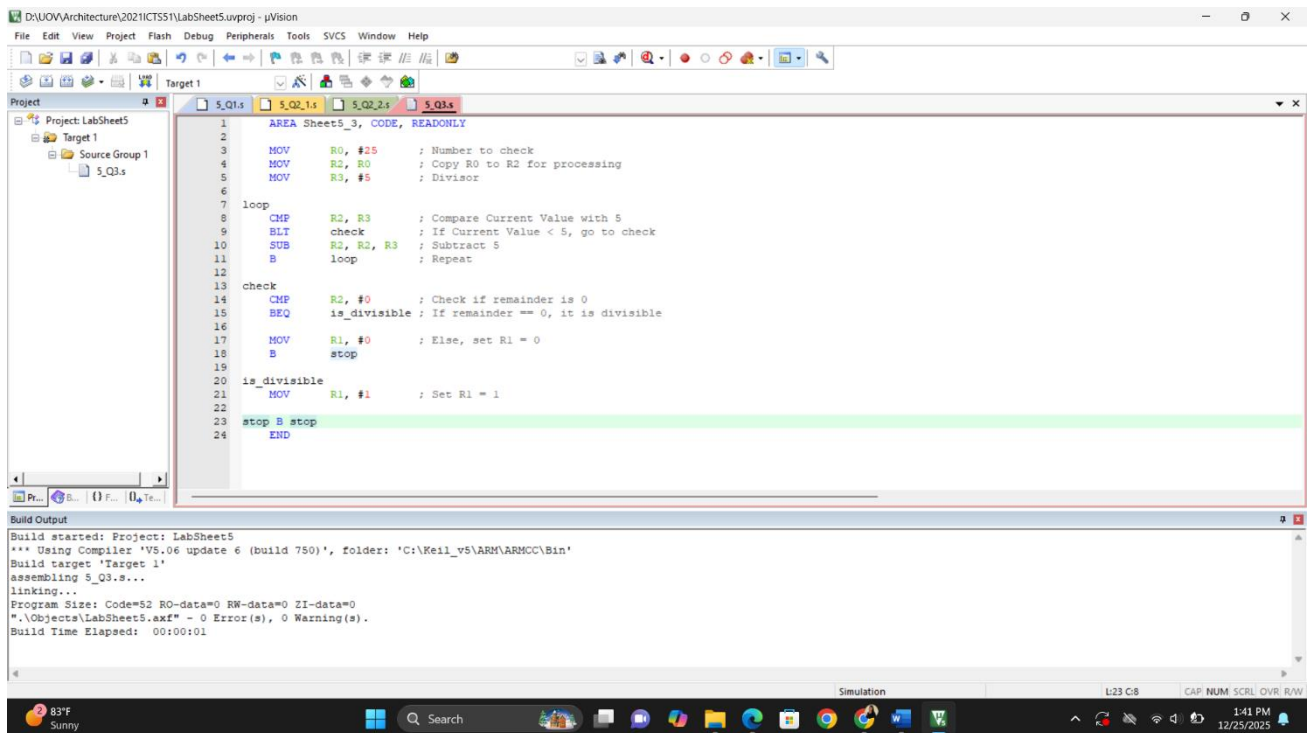


▪ Decrement

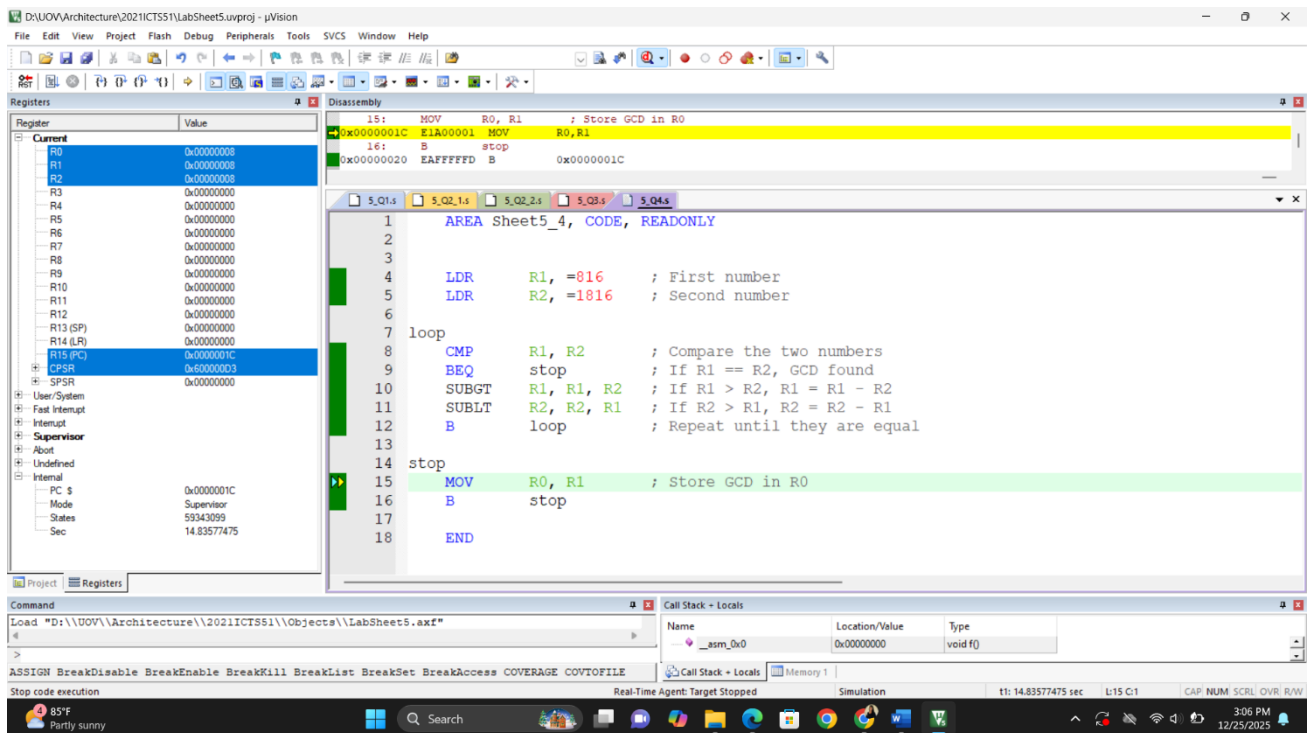
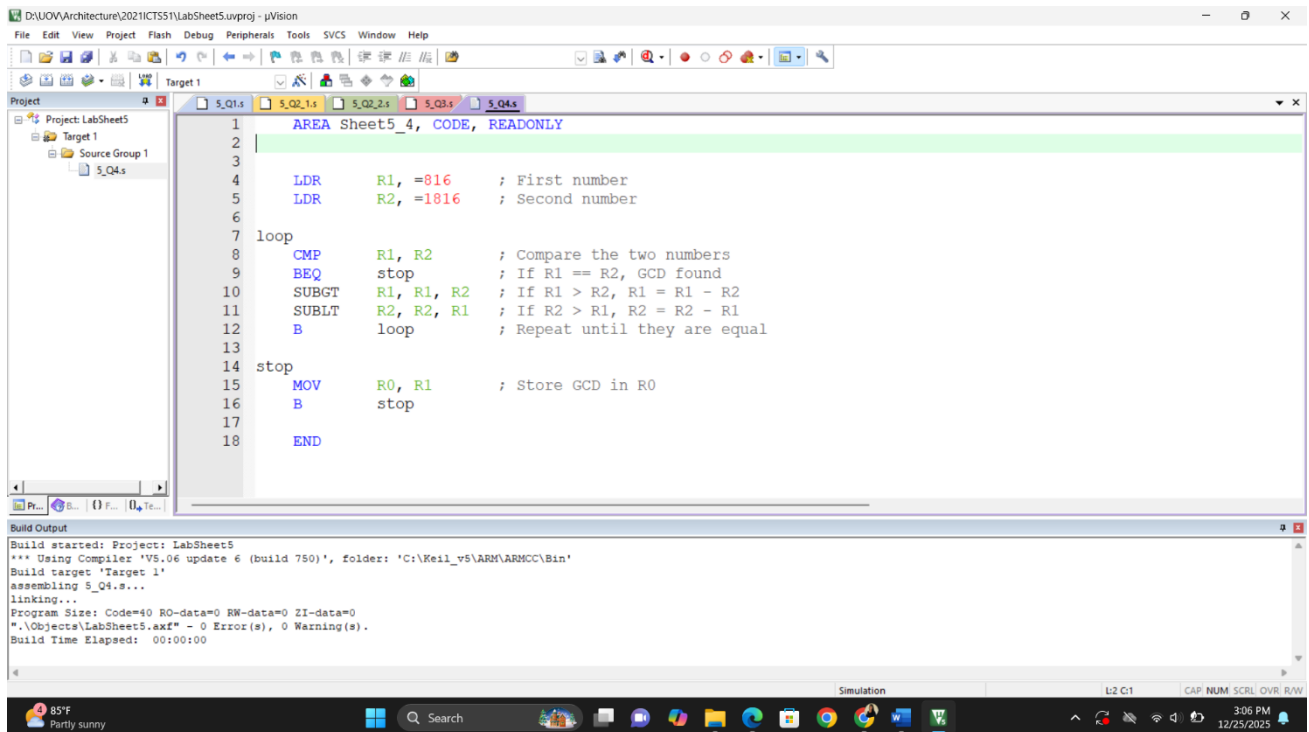
Hint: $5! = 5 * 4 * 3 * 2 * 1$



- 3) Write a program to check whether a number in R0 is divisible by 5.
If divisible, set R1 = 1; otherwise, R1 = 0



- 4) Write a program to find the Greatest Common Divisor (GCD) of 816 and 1816.
Hint: GCD is the biggest number that can divide both numbers completely



Discussion:

1) *****

- Logic: Uses the CMP instruction to subtract two values internally and update processor flags.
- Conditional Branching: The BGT (Branch if Greater Than) instruction directs the program flow based on the comparison result.
- Storage: If the first number is larger, it jumps to a specific label to move that value to R3; otherwise, it proceeds to move the second number.

2) *****

- Accumulation: Both methods use a register initialized to 1 to store the running product.
- Multiplication: The MUL instruction performs the core calculation in every loop iteration.
- Direction:
 - Increment: Starts the counter at 1 and climbs up to N.
 - Decrement: Starts the counter at \$N\$ and counts down to 1.
- Boundary: Both loops use a condition (BGT or BLT) to stop the process once the counter exceeds the target range.

3) *****

- Modulo via Subtraction: Since direct remainder instructions aren't always available, the program repeatedly subtracts 5 from the original number.
- Remainder Identification: Once the value becomes less than 5, the remaining value is the "remainder."
- Result Logic: A final comparison (CMP) checks if that remainder is exactly 0. If it is, the number is divisible, and the indicator R1 is set to 1.

4) *****

- Euclidean Algorithm: This program uses the subtraction-based Euclidean method to find the GCD.
- Iterative Reduction: The larger of the two numbers is replaced by the difference between the two numbers ($R1 = R1 - R2$ or $R2 = R2 - R1$).
- Convergence: The process repeats until both registers hold the same value. That final equal value is the Greatest Common Divisor.
- Large Constants: Because 816 and 1816 are large, the LDR pseudo-instruction is required instead of MOV to avoid "Immediate value out of range" errors.

Reference: Keil Software