2021/ICTS/51

TICT3113(P) – Computer Architecture & Organization

Lab sheet 06

Title: Character Handling in ARM Assembly

Aims:

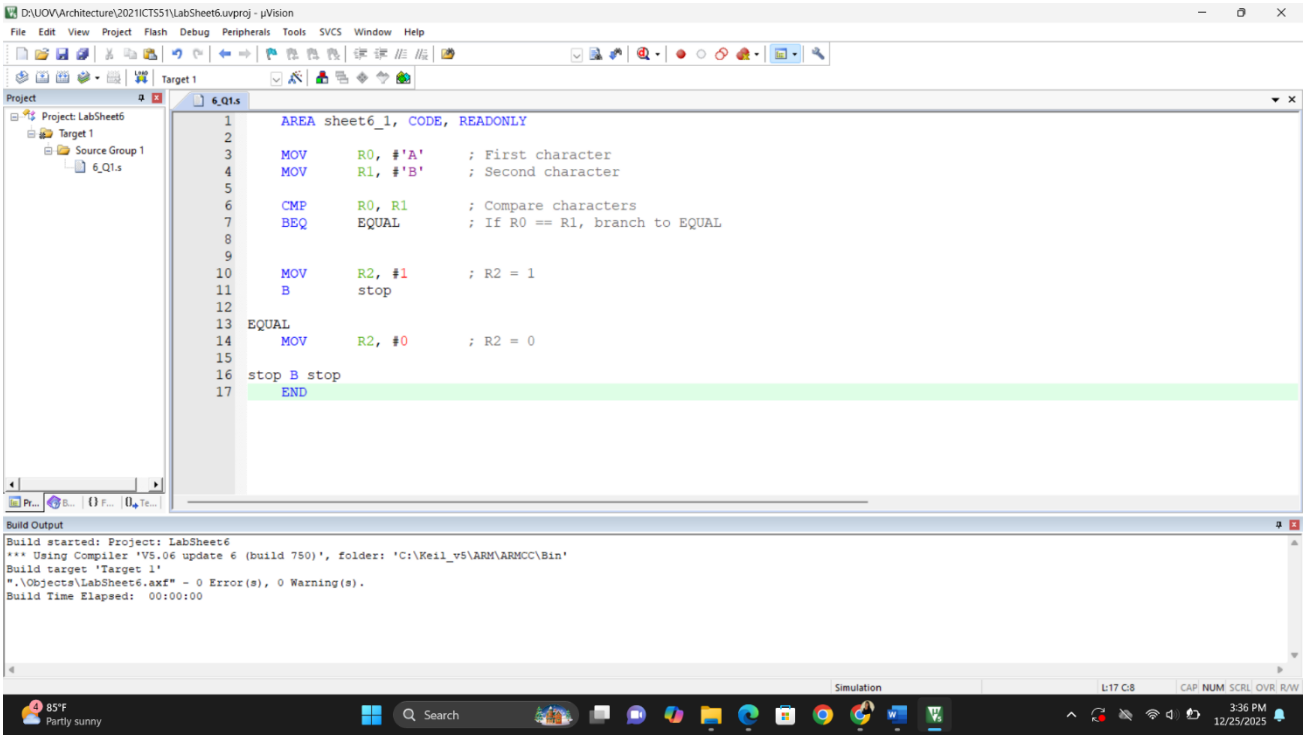
1. To understand how characters are represented in ARM Assembly using ASCII values and how they are loaded into registers.
2. To learn how to perform character classification, such as checking whether a character is uppercase, lowercase, or neither.
3. To observe how comparison instructions (CMP) update condition flags and how conditional branching is used to make decisions.

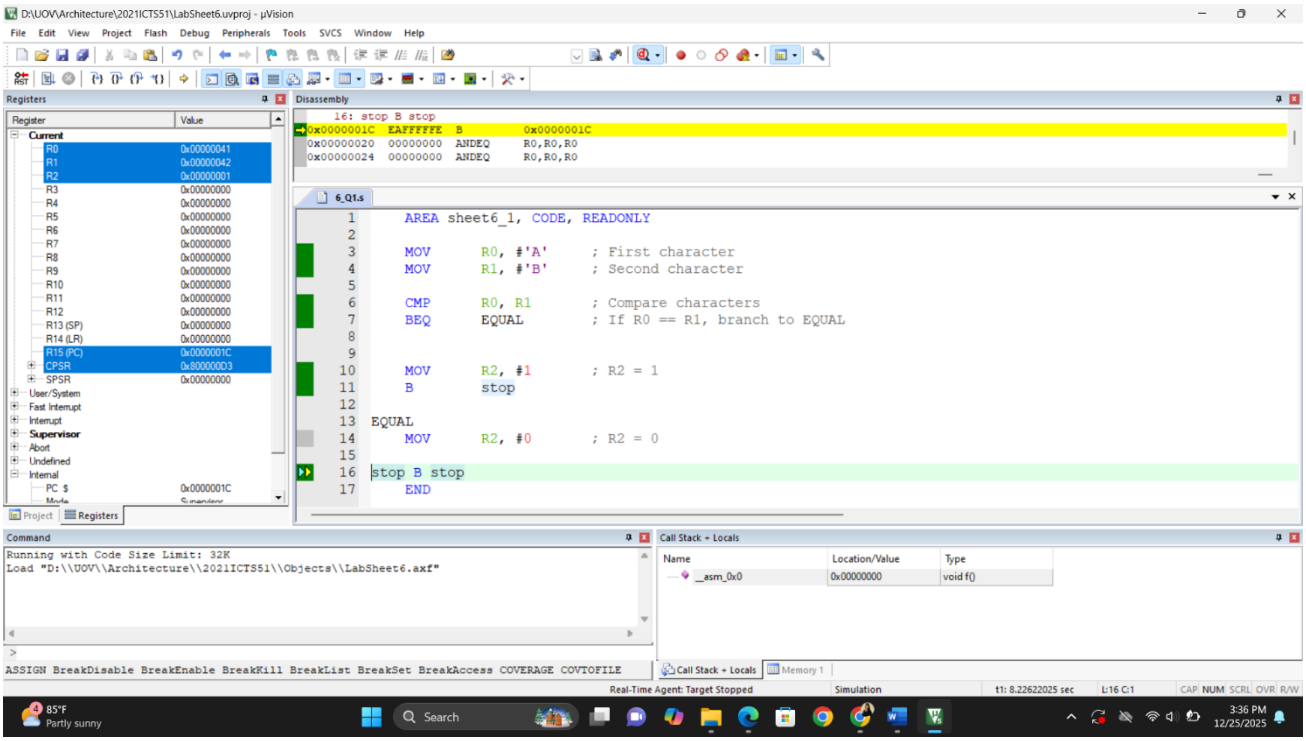
Tasks:

1. Identify the ASCII ranges for uppercase letters, lowercase letters, digits, and other characters.
2. Write an ARM Assembly program to:
• Load a character into a register
• Compare it against ASCII ranges
• Set an indicator register to represent uppercase, lowercase, or other
3. Assemble and build the program using Keil µVision IDE.
4. Run the program in Debug Mode and observe how register values change during each loop iteration.
5. Analyze how the use of ASCII boundaries and branching allows ARM Assembly programs to handle different character types.

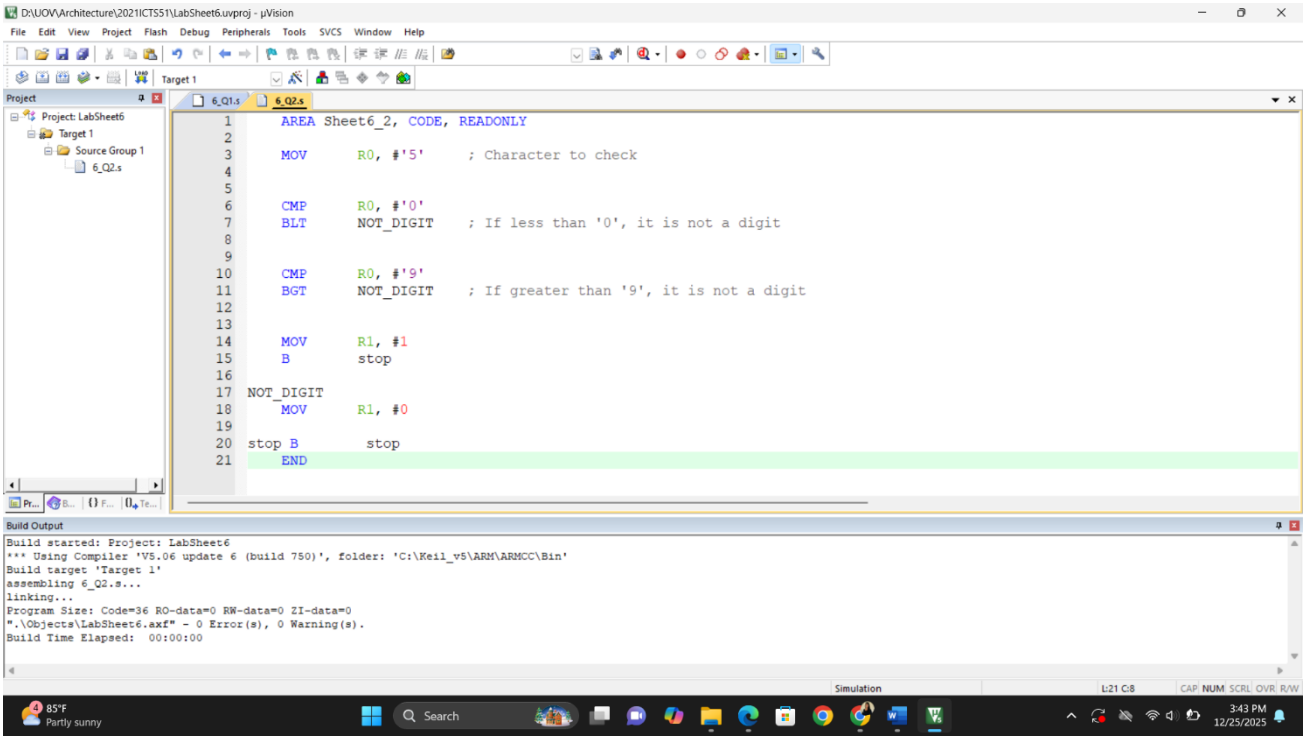
Activities:

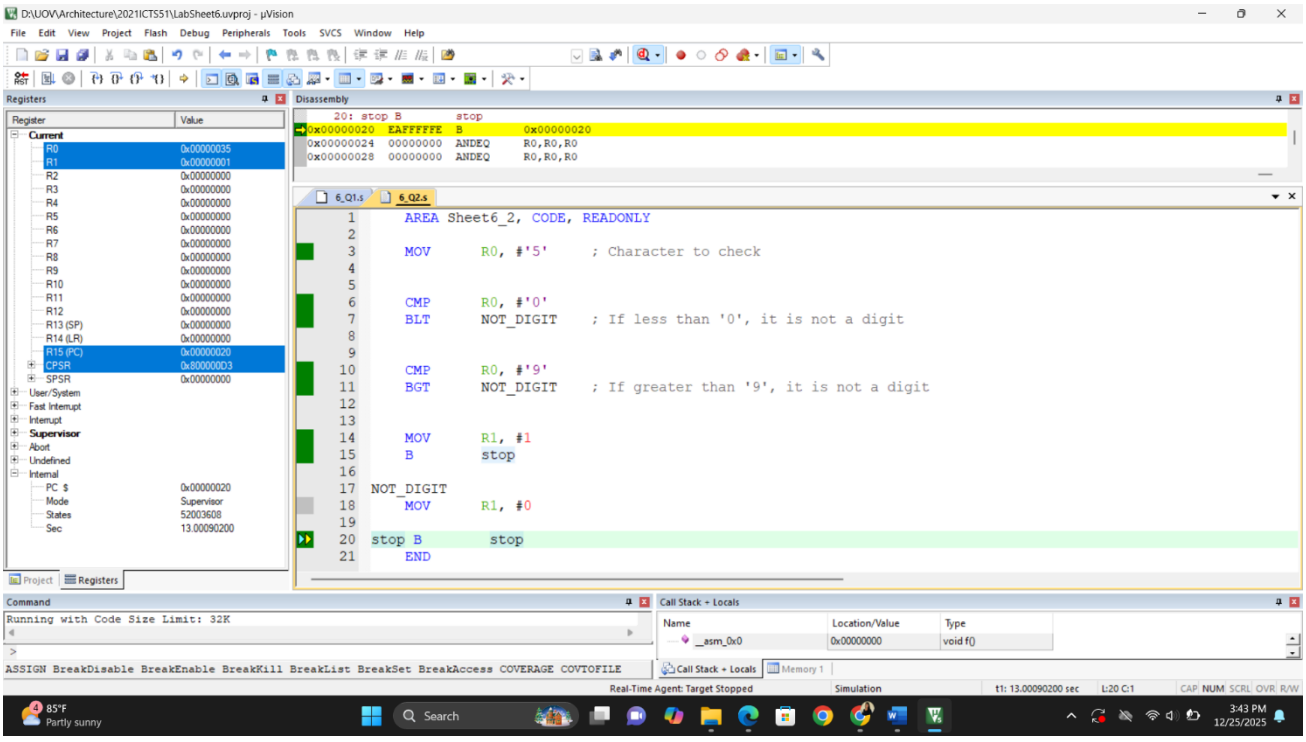
1) Compare two characters (equal or not)
   - If Equal MOV R2,#0
   - Else MOV R2,#1



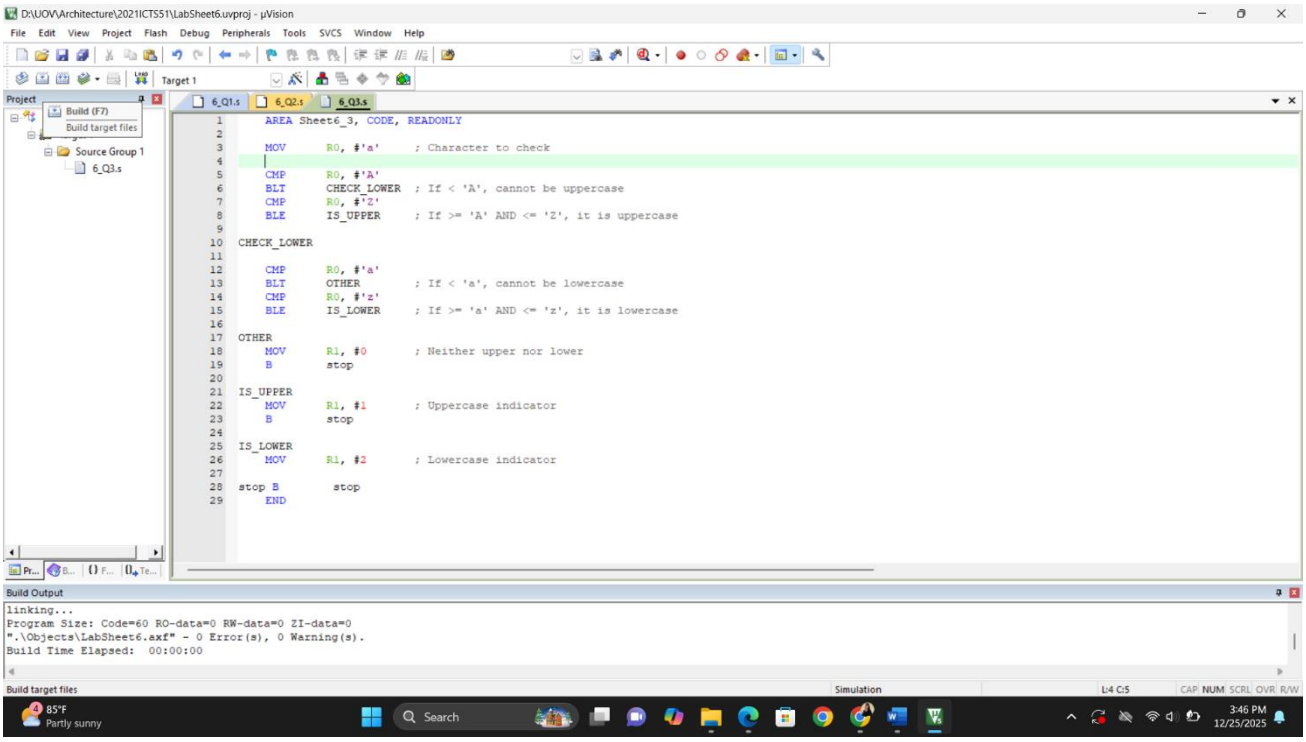


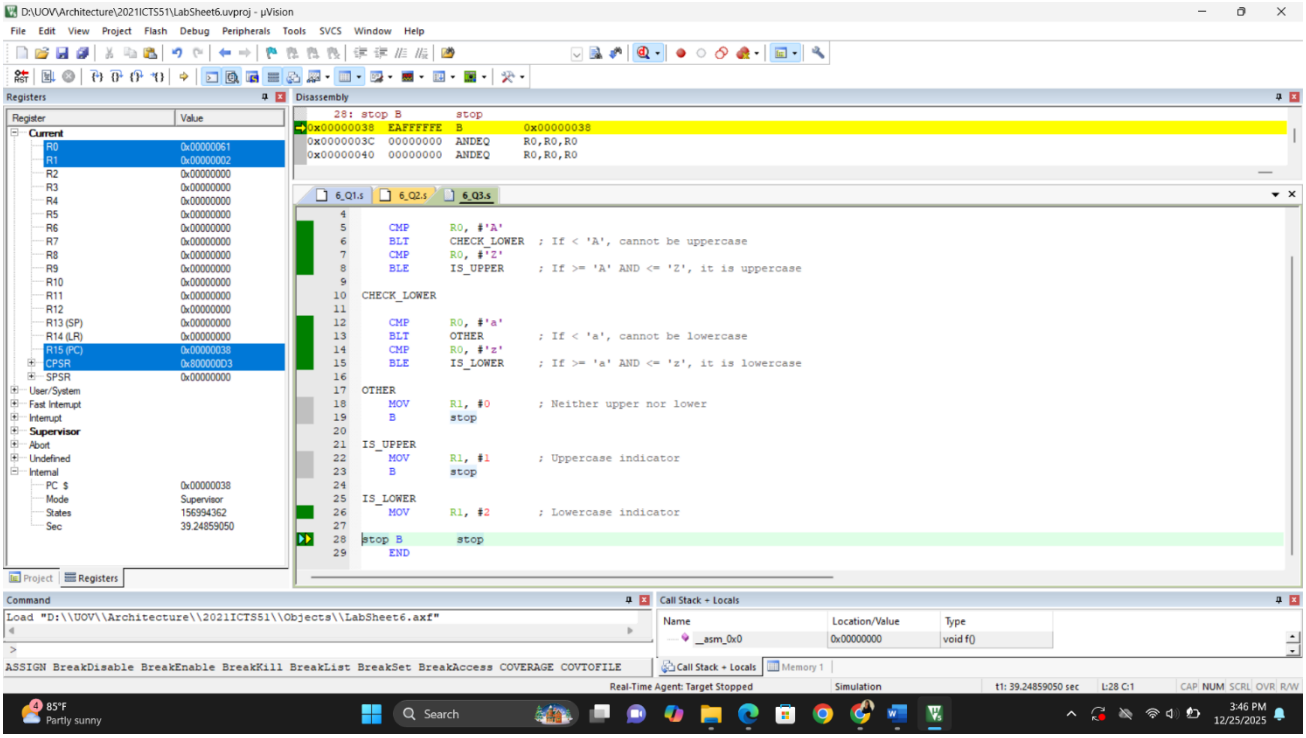
2) Check if a character is a digit (‘0’–’9’)
   - If digit mov R1,#1
   - If not a digit R1,#0



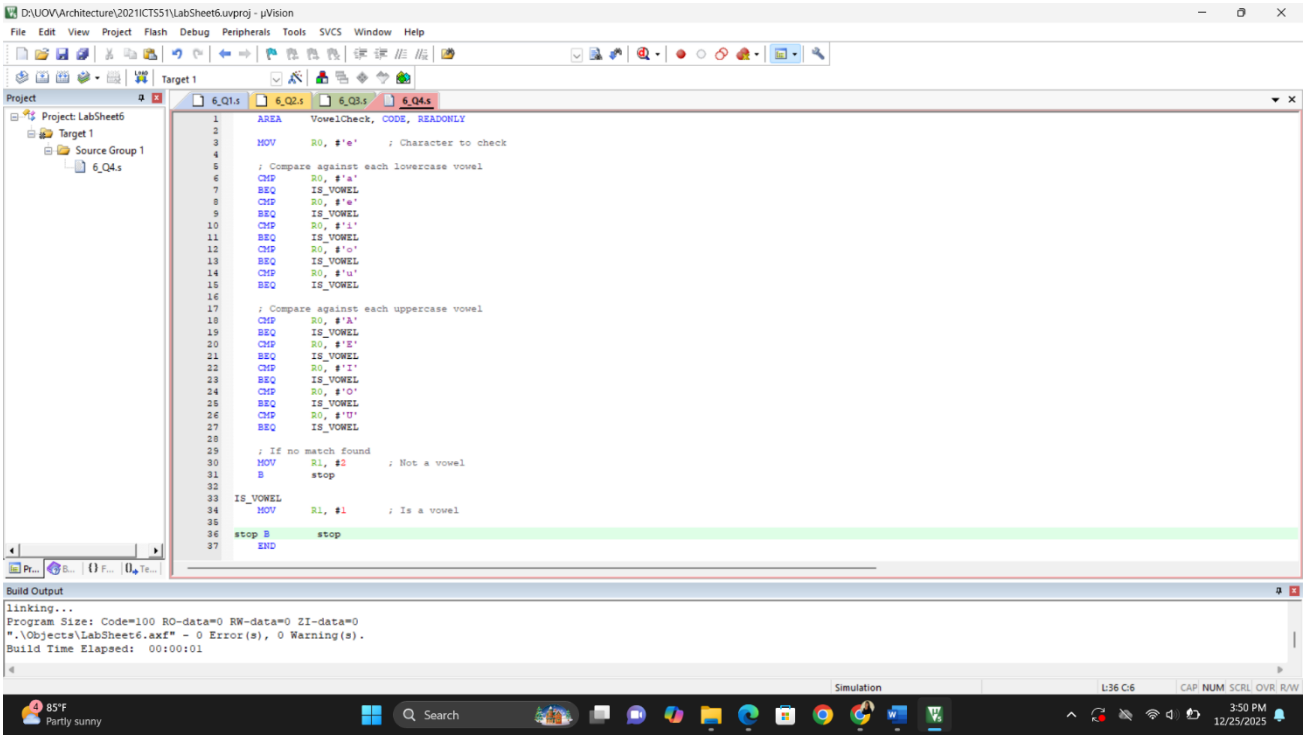


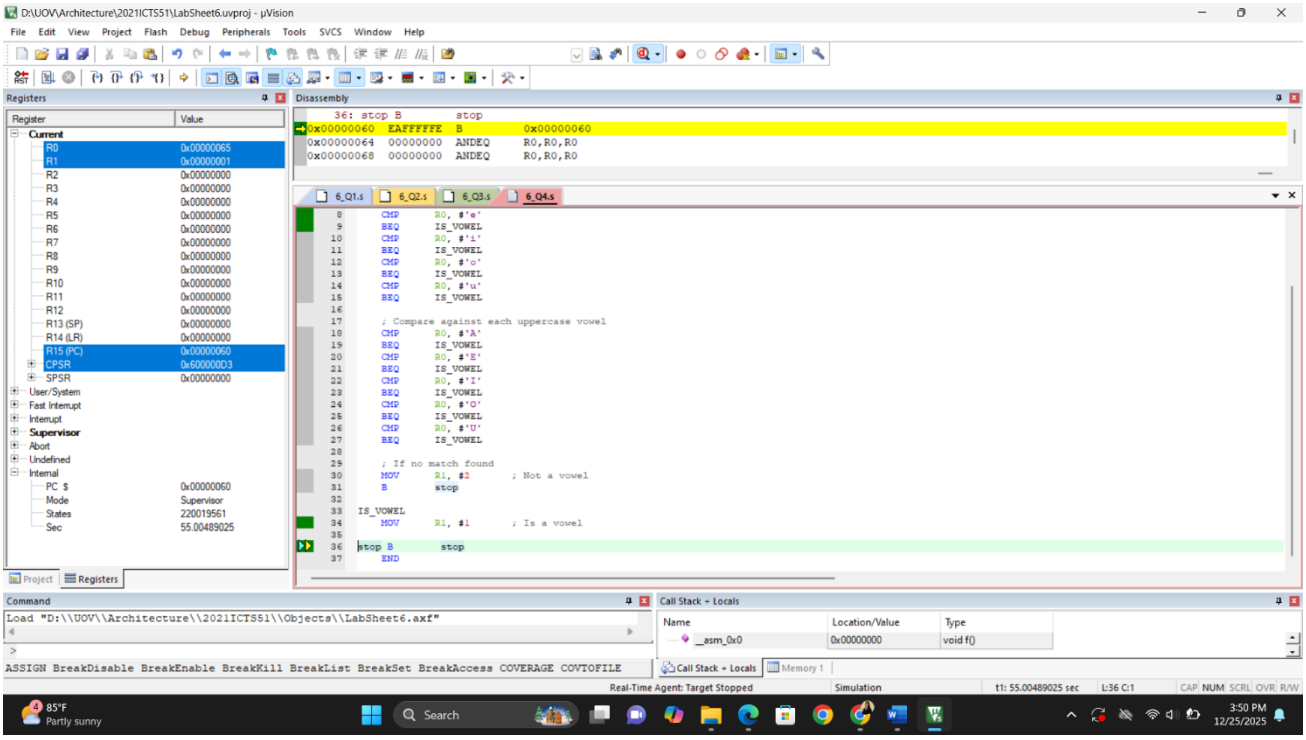
3) Check if a character is uppercase or lowercase:
   - If uppercase → R1 = 1
   - If lowercase → R1 = 2
   - Otherwise → R1 = 0





4) Check if a character is a vowel
   - If vowel mov R1, #1
   - If not a vowel R1, #2





Discussion:

1) **********

- ASCII Comparison: The program treats characters (like 'A' or 'B') as their underlying ASCII numerical values during comparison.
- Flag Usage: The CMP instruction subtracts the value of the second character from the first; if the result is zero, the "Zero" flag is set.
- Conditional Logic: BEQ (Branch if Equal) relies on that Zero flag to decide whether to skip the "Else" logic and set R2 to 0.

2) **********

- Boundary Checking: To identify a digit, the program checks if the character falls within the inclusive range of ASCII '0' (48) to '9' (57).
- Two-Step Filter:
    - First, it eliminates anything mathematically less than '0'.
    - Second, it eliminates anything mathematically greater than '9'.
- Boolean Result: If the character survives both checks, it is confirmed as a digit, and R1 is assigned 1.

3) **********

- Range Segregation: This logic uses multiple ranges because uppercase letters ('A'-'Z') and lowercase letters ('a'-'z') are located in different sections of the ASCII table.
- Fallback Logic: The "Otherwise" condition acts as a catch-all for symbols, punctuation, or numbers, ensuring R1 is reset to 0 if neither alphabetic range matches.
- Efficiency: Once a match is found (e.g., it is confirmed as Uppercase), the program immediately branches to the end to avoid unnecessary checks.

4) **********

- Explicit Matching: Unlike digits or case checking, vowels are not contiguous in the alphabet (they are separated by consonants), so the program checks each possibility individually.
- Case Sensitivity: The program includes checks for both lowercase ('a, e, i, o, u') and uppercase ('A, E, I, O, U') to ensure the check is robust regardless of how the character was typed.
- OR Logic: This is an implementation of "Logical OR"—if any one of these comparisons is true, the character is classified as a vowel.

Reference: Keil Software