

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 - Theory:

1. What is a Character in ARM?
 - Characters are stored as ASCII values (0–127).
Example: 'A' = 65, 'a' = 97, '0' = 48
 - In ARM Assembly, characters are loaded using:
`MOV R0, #'A'` or `MOV R0, 'A'`
2. ASCII Value Ranges
 - Uppercase letters (A–Z): 65–90
 - Lowercase letters (a–z): 97–122
 - Digits ('0'–'9'): 48–57
3. Loading Characters into Registers
 - Characters stored as immediate ASCII values: `MOV R1, #'A'`
 - Also possible using hex values: `MOV R1, #0x41`

Exercises:

1) Compare two characters (equal or not)

- If Equal MOV R2,#0
- Else MOV R2,#1

The screenshot shows the μVision IDE interface with the following details:

- Registers:** Shows the current register values, including R0=0x00000041, R1=0x00000042, and R2=0x00000001.
- Disassembly:** Displays the assembly code for the character comparison exercise. The code uses R0 and R1 to hold characters 'A' and 'B' respectively, compares them, and branches based on equality.
- Call Stack + Locals:** Shows a single entry in the call stack: `_asm_0x0` at location 0x00000000.
- Command Line:** Displays the command line with the current project loaded.

```

16: stop b stop
0x00000024 EAFFFFFE B 0x00000024
0x00000028 00000000 ANDEQ R0,R0,R0
0x0000002C 00000000 ANDEQ R0,R0,R0

sheet6_Que1.s
1     area sheet6_1,code,readonly
2     MOV R0,#'A'      ;first character
3     MOV R1,#'B'      ;second character
4
5     CMP R0,R1        ;compare characters
6     BEQ Equal        ;if R0 == R1, Branch to Equal
7     BNE NotEqual    ;if R0 != R1, Branch not to Equal
8
9     Equal
10    MOV R2,#0        ;R2 = 0
11    B stop
12
13    NotEqual
14    MOV R2,#1        ;R2 = 1
15    b stop
16
17 stop b stop
18 end

```

2) Check if a character is a digit ('0'-'9')

- If digit mov R1,#1
- If not a digit R1,#0

The screenshot shows the μVision IDE interface with the following details:

- Registers:** Shows the current register values, including R0=0x00000035, R1=0x00000001.
- Disassembly:** Displays the assembly code for digit checking. It compares the character in R0 against the ASCII values of '0' and '9'. If R0 is less than '0', it sets R1 to #0. If R0 is greater than '9', it also sets R1 to #0. Otherwise, it sets R1 to #1.
- Call Stack + Locals:** Shows a single entry in the call stack: `_asm_0x0` at location 0x00000000.
- Command Line:** Displays the command line with the current project loaded.

```

17: stop b stop
0x00000024 EAFFFFFE B 0x00000024
0x00000028 00000000 ANDEQ R0,R0,R0
0x0000002C 00000000 ANDEQ R0,R0,R0

sheet6_Que2.s
1     area sheet6_1,code,readonly
2     MOV R0,#'5'      ;character to check
3     MOV R1,#0
4
5     CMP R0,#'0'
6     BLT NOT_DIGIT   ;If less than '0'
7
8     CMP R0,#'9'
9     BGT NOT_DIGIT   ;If greater than '9'
10
11    MOV R1,#1        ;If is a digit
12    b stop
13
14    NOT_DIGIT
15    MOV R1,#0        ;Not a digit
16
17 stop b stop
18 end

```

3) Check if a character is uppercase or lowercase:

- If uppercase → R1 = 1
- If lowercase → R1 = 2
- Otherwise → R1 = 0

The screenshot shows the Keil uVision IDE interface with the following details:

- Registers:** Shows the current register values, including R0=0x00000064, R1=0x00000002, and PC=\$0x00000038.
- Disassembly:** The assembly code for "sheet6_Que3.s" is displayed. It initializes R0 to 'd', checks if it's uppercase ('A-Z'), and if not, checks if it's lowercase ('a-z'). If neither, R1 is set to 0. Then, it branches based on the result: if uppercase, R1 is set to 1; if lowercase, R1 is set to 2. Finally, it stops the program.
- Call Stack + Locals:** Shows a single entry: __asm_0x0 at 0x00000000.
- Project:** Shows the project structure with "sheet6_Que3.s" selected.
- Build Output:** Shows the build log: "Build started: Project: LabSheet6", "Build target 'Target 1'", ".\Objects\LabSheet6.axf" - 0 Error(s), 0 Warning(s), and "Build Time Elapsed: 00:00:00".

4) Check if a character is a vowel

- If vowel mov R1,#1
- If not a vowel R1,#2

The screenshot shows the Keil uVision IDE interface with the following details:

- Project:** Shows the project structure with "sheet6_Que4.s" selected.
- Disassembly:** The assembly code for "sheet6_Que4.s" is displayed. It initializes R0 to 'C' and R1 to 0. It then loops through all lowercase vowels ('a', 'e', 'i', 'o', 'u') and sets R1 to 1 if the character matches. If none match, R1 is set to 2. Finally, it stops the program.
- Call Stack + Locals:** Shows a single entry: __asm_0x0 at 0x00000000.
- Build Output:** Shows the build log: "Build started: Project: LabSheet6", "Build target 'Target 1'", ".\Objects\LabSheet6.axf" - 0 Error(s), 0 Warning(s), and "Build Time Elapsed: 00:00:00".

The screenshot shows the Keil µVision IDE interface with the following windows:

- Registers**: Shows the current register values. The R0 register is highlighted in yellow, containing the value 0x00000043.
- Disassembly**: Displays the assembly code for the file "sheet6_Que4.s". The code checks if a character is a vowel or not, using CMP and BEQ instructions to compare R0 with ASCII values for 'a', 'e', 'i', 'o', and 'u'. If not a vowel, it sets R1 to 1 and stops. The PC register is at address 0x00000064.
- Call Stack + Locals**: Shows a single local variable: __asm_0x0 with a value of 0x00000000.
- Project**: Shows the project configuration.
- Command**: Displays build logs: "Running with Code Size Limit: 32K" and "Load "D:\\C_Architecture\\Objects\\LabSheet6.axf"".
- Real-Time Agent**: Shows "Target Stopped" with a timestamp of "t1: 9.03761525 sec".
- Simulation**: Shows "L31 C1 CAP NUM SCRL OVR R/W".

Discussion:

This lab focused on character handling in ARM Assembly using ASCII values. Programs were written to compare characters, check for digits, uppercase or lowercase letters, and vowels. The CMP instruction and conditional branches were used to make decisions, and the results were stored in indicator registers. Keil debugger was used to observe how register values and condition flags change during execution.

Reference:

- Lecturer's Notes.