

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

D:\UOV\Architecture\2021ICTS51\LabSheet6.uvproj - μVision

```

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
File Explorer Project Target 1 Search
Project LabSheet6 Target 1 Source Group 1 6_Q1.s
1 AREA sheet6_1, CODE, READONLY
2
3 MOV R0, #'A' ; First character
4 MOV R1, #'B' ; Second character
5
6 CMP R0, R1 ; Compare characters
7 BEQ EQUAL ; If R0 == R1, branch to EQUAL
8
9
10 MOV R2, #1 ; R2 = 1
11 B stop
12
13 EQUAL
14 MOV R2, #0 ; R2 = 0
15
16 stop B stop
17 END

```

Build Output

```

Build started: Project: LabSheet6
*** Using Compiler 'V5.06 update 6 (build 750)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Build target 'Target 1'
".\Objects\LabSheet6.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:00

```

85°F Partly sunny

D:\UOV\Architecture\2021ICTS51\LabSheet6.uvproj - μVision

| Register | Value |
|----------|------------|
| R0 | 0x00000041 |
| R1 | 0x00000042 |
| R2 | 0x00000001 |
| R3 | 0x00000000 |
| R4 | 0x00000000 |
| R5 | 0x00000000 |
| R6 | 0x00000000 |
| R7 | 0x00000000 |
| R8 | 0x00000000 |
| R9 | 0x00000000 |
| R10 | 0x00000000 |
| R11 | 0x00000000 |
| R12 | 0x00000000 |
| R13 (SP) | 0x00000000 |
| R14 (LR) | 0x00000000 |
| R15 (PC) | 0x0000001C |
| CPSR | 0x80000003 |
| SPSR | 0x00000000 |

Registers

Disassembly

```

1: stop B stop
0x0000001C EAFFFFF B 0x00000001C
0x00000020 00000000 ANDEQ R0,R0,R0
0x00000024 00000000 ANDEQ R0,R0,R0

```

6_Q1.s

```

1 AREA sheet6_1, CODE, READONLY
2
3 MOV R0, #'A' ; First character
4 MOV R1, #'B' ; Second character
5
6 CMP R0, R1 ; Compare characters
7 BEQ EQUAL ; If R0 == R1, branch to EQUAL
8
9
10 MOV R2, #1 ; R2 = 1
11 B stop
12
13 EQUAL
14 MOV R2, #0 ; R2 = 0
15
16 stop B stop
17 END

```

Command

```

Running with Code Size Limit: 32K
Load "D:\UOV\Architecture\2021ICTS51\Objects\LabSheet6.axf"

```

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE

Call Stack + Locals

| Name | Location/Value | Type |
|-----------|----------------|----------|
| __asm_0x0 | 0x00000000 | void f() |

Real-Time Agent: Target Stopped

3:36 PM 12/25/2025

85°F Partly sunny

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

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

Screenshot of the µVision IDE showing assembly code for a character digit check. The code compares a character in R0 against '0' and '9'. If less than '0', it's not a digit; if greater than '9', it's also not a digit. Otherwise, it moves #1 to R1.

```

1 AREA Sheet6_2, CODE, READONLY
2
3 MOV R0, #'5'      ; Character to check
4
5
6 CMP R0, #'0'
7 BLT NOT_DIGIT    ; If less than '0', it is not a digit
8
9
10 CMP R0, #'9'
11 BGT NOT_DIGIT   ; If greater than '9', it is not a digit
12
13
14 MOV R1, #1
15 B stop
16
17 NOT_DIGIT
18 MOV R1, #0
19
20 stop B          stop
21 END

```

Build Output:

```

Build started: Project: LabSheet6
*** Using Compiler 'V5.06 update 6 (build 750)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Build target "Target 1"
assembling 6_Q2.s...
linking...
Program Size: Code=36 RO-data=0 RW-data=0 ZI-data=0
".\Objects\LabSheet6.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:00

```

Windows Taskbar:

Screenshot of the µVision IDE showing the Registers and Disassembly panes. The Registers pane shows R0=0x00000035 and R1=0x00000001. The Disassembly pane shows the same assembly code as the previous screenshot.

Registers:

| Register | Value |
|----------|------------|
| R0 | 0x00000035 |
| R1 | 0x00000001 |
| R2 | 0x00000000 |
| R3 | 0x00000000 |
| R4 | 0x00000000 |
| R5 | 0x00000000 |
| R6 | 0x00000000 |
| R7 | 0x00000000 |
| R8 | 0x00000000 |
| R9 | 0x00000000 |
| R10 | 0x00000000 |
| R11 | 0x00000000 |
| R12 | 0x00000000 |
| R13 (SP) | 0x00000000 |
| R14 (LR) | 0x00000000 |
| R15 (PC) | 0x00000020 |
| CPSR | 0x80000003 |
| SPSR | 0x00000000 |

Disassembly:

```

20: stop B          stop
0x00000020_EAFFFFFF_B      0x00000020
0x00000024_00000000_ANDEQ  R0,R0,R0
0x00000028_00000000_ANDEQ  R0,R0,R0

```

Registers pane (continued):

| Register | Value |
|----------|-------------|
| PC \$ | 0x00000020 |
| Mode | Supervisor |
| States | 52003608 |
| Sec | 13.00090200 |

Call Stack + Locals:

| Name | Location/Value | Type |
|----------|----------------|----------|
| _asm_0x0 | 0x00000000 | void f() |

3) Check if a character is uppercase or lowercase:

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

The screenshot shows the µVision IDE interface with the assembly code for character classification. The code uses R0 as the character register and R1 as the indicator register. It checks if the character is uppercase ('A'-'Z') or lowercase ('a'-'z'). If uppercase, R1 is set to 1; if lowercase, R1 is set to 2; otherwise, R1 is set to 0. The assembly code is as follows:

```

AREA Sheet6_3, CODE, READONLY
1    MOV   R0, #'a'      ; Character to check
2
3    CMP   R0, #'A'      ; If < 'A', cannot be uppercase
4    BLT   CHECK_LOWER
5    CMP   R0, #'Z'
6    BLE   IS_UPPER     ; If >= 'A' AND <= 'Z', it is uppercase
7
8    CMP   R0, #'a'      ; If < 'a', cannot be lowercase
9    BLT   OTHER
10   CMP   R0, #'z'
11   BLE   IS_LOWER    ; If >= 'a' AND <= 'z', it is lowercase
12
13   OTHER
14   MOV   R1, #0        ; Neither upper nor lower
15   B    stop
16
17   IS_UPPER
18   MOV   R1, #1        ; Uppercase indicator
19   B    stop
20
21   IS_LOWER
22   MOV   R1, #2        ; Lowercase indicator
23   B    stop
24
25   stop B             stop
26
27
28 stop B             stop
29 END

```

The build output window shows the linking process completed successfully.

This screenshot shows the µVision IDE with the Registers and Assembly windows open. The Registers window displays the current state of the processor registers, including R0, R1, and the PC at address 0x00000038. The Assembly window shows the same assembly code as the previous screenshot. The status bar indicates the target is stopped at address 0x00000038.

| Register | Value |
|----------|------------|
| R0 | 0x00000061 |
| R1 | 0x00000002 |
| R2 | 0x00000000 |
| R3 | 0x00000000 |
| R4 | 0x00000000 |
| R5 | 0x00000000 |
| R6 | 0x00000000 |
| R7 | 0x00000000 |
| R8 | 0x00000000 |
| R9 | 0x00000000 |
| R10 | 0x00000000 |
| R11 | 0x00000000 |
| R12 | 0x00000000 |
| R13 (SP) | 0x00000000 |
| R14 (LR) | 0x00000000 |
| R15 (PC) | 0x00000038 |
| CPSR | 0x00000003 |
| SPSR | 0x00000000 |

4) Check if a character is a vowel

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

D:\UOV\Architecture\2021ICTS51\LabSheet6.uvproj - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Project: LabSheet6 Target 1 6.Q1.s 6.Q2.s 6.Q3.s 6.Q4.s

```

1 AREA VowelCheck, CODE, READONLY
2
3 MOV R0, #'e' ; Character to check
4
5 ; Compare against each lowercase vowel
6 CMP R0, #'a'
7 BEQ IS_VOWEL
8 CMP R0, #'e'
9 BEQ IS_VOWEL
10 CMP R0, #'i'
11 BEQ IS_VOWEL
12 CMP R0, #'o'
13 BEQ IS_VOWEL
14 CMP R0, #'u'
15 BEQ IS_VOWEL
16
17 ; Compare against each uppercase vowel
18 CMP R0, #'A'
19 BEQ IS_VOWEL
20 CMP R0, #'E'
21 BEQ IS_VOWEL
22 CMP R0, #'I'
23 BEQ IS_VOWEL
24 CMP R0, #'O'
25 BEQ IS_VOWEL
26 CMP R0, #'U'
27 BEQ IS_VOWEL
28
29 ; If no match found
30 MOV R1, #2 ; Not a vowel
31 B stop
32
33 IS_VOWEL
34 MOV R1, #1 ; Is a vowel
35
36 stop B stop
37 END

```

Build Output

linking...

Program Size: Code=100 RO=data=0 RW=data=0 ZI=data=0
 ".\Objects\LabSheet6.axf" - 0 Error(s), 0 Warning(s).

Build Time Elapsed: 00:00:01

85°F Partly sunny

D:\UOV\Architecture\2021ICTS51\LabSheet6.uvproj - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

| Register | Value |
|----------|------------|
| R0 | 0x00000065 |
| R1 | 0x00000001 |
| R2 | 0x00000000 |
| R3 | 0x00000000 |
| R4 | 0x00000000 |
| R5 | 0x00000000 |
| R6 | 0x00000000 |
| R7 | 0x00000000 |
| R8 | 0x00000000 |
| R9 | 0x00000000 |
| R10 | 0x00000000 |
| R11 | 0x00000000 |
| R12 | 0x00000000 |
| R13 (SP) | 0x00000000 |
| R14 (LR) | 0x00000000 |
| R15 (PC) | 0x00000060 |
| CPSR | 0x60000003 |
| SPSR | 0x00000000 |

Registers

Registers

Dissassembly

```

3: stop B stop
0x00000060 0xFFFFFFF B 0x00000060
0x00000064 00000000 ANDEQ R0,R0,R0
0x00000068 00000000 ANDEQ R0,R0,R0
8: CMP R0, #'e'
9: BEQ IS_VOWEL
10: CMP R0, #'A'
11: BEQ IS_VOWEL
12: CMP R0, #'O'
13: BEQ IS_VOWEL
14: CMP R0, #'U'
15: BEQ IS_VOWEL
16:
17; Compare against each uppercase vowel
18: CMP R0, #'A'
19: BEQ IS_VOWEL
20: CMP R0, #'E'
21: BEQ IS_VOWEL
22: CMP R0, #'I'
23: BEQ IS_VOWEL
24: CMP R0, #'O'
25: BEQ IS_VOWEL
26: CMP R0, #'U'
27: BEQ IS_VOWEL
28:
29; If no match found
30: MOV R1, #2 ; Not a vowel
31: B stop
32:
33: IS_VOWEL
34: MOV R1, #1 ; Is a vowel
35:
36: stop B stop
37: END

```

Call Stack + Locals

| Name | Location/Value | Type |
|----------|----------------|----------|
| _asm_0x0 | 0x00000000 | void f() |

Real-Time Agent: Target Stopped

Simulation

11:55.00489025 sec 1:36 C1 CAP NUM SCBL OVR/R/W

85°F Partly sunny

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