

INCREASING SYSTEM SAFETY OF MICROCONTROLLER AND PERIPHERALS WITH SOFTWARE LIBRARY

Software Qualification Test Document

Dissertation by

SUBHRA SAR

2023HT65019

Dissertation work carried out at

Bosch Global Software and Technologies, Bangalore

Master of Technology in Automotive Electronics Degree Program

Under the Supervision of

M SATISH KUMAR

Bosch Global Software and Technologies, Bangalore



BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE

PILANI (RAJASTHAN)

March, 2025

Table of Contents

Introduction	3
Add.s	4
Substract_Multiply.s.....	6
Move.s.....	8
Store.s	10
Bitwise.s	12
Shift_Rotate.s.....	14
Compare.s	16
Stack.s	18
Floating.s	20
Control_Flow.s.....	22
Branching.s.....	24
Conclusion.....	26

Introduction

This document serves as a reference for conducting the qualification test of the Proof-of-Concept (PoC) code developed to validate various instruction sets of the Cortex-M processor series. It provides a concise overview of the instruction sets evaluated through the implementation of supplementary logic within the code. Additionally, it outlines the expected test outcomes for both positive and negative scenarios, with a focus on the verification of register values.

The results obtained from simulations performed using Keil µVision 5 are included as supporting evidence, demonstrating the validity and effectiveness of the implemented code under positive circumstances.

The structure of this document is organized according to the names of the assembly files, each corresponding to a specific group of instruction sets.

Note

The Software test code has been updated in the Git Hub Link:

[SARSHRA/Cortex_M4: Verification of assembly code for Cortex_M4](#)

Add.s

Description

ADD: Adds two registers or a register and immediate.

ADC: Adds with carry — useful for multi-word addition.

ADR: Loads the current or nearby program counter-relative address into a register.

Expected Test Result

Check the value of R3:

0x01: success

0xFF: fail

Obtained Test Result

Compilation:

The screenshot shows the Keil µVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. Below the menu is a toolbar with various icons. The main workspace has tabs for startup_ARMCM4.c, system_ARMCM4.c, and Add.s. The Add.s tab is active, displaying the following assembly code:

```
1 .syntax unified
2 .thumb
3
4 .global main
5 main:
6
7 // ---- ADD Test ----
8 MOV R0, #0x10      // R0 = 0x10
9 MOV R1, #0x20      // R1 = 0x20
10 ADD R2, R0, R1    // R2 = 0x10 + 0x20 = 0x30
11
12 // Optional check
13 LDR R3, =0x30
14 CMP R2, R3
15 BNE fail          // Branch to fail if mismatch
16
17 // ---- ADC Test ----
18 MOV R4, #0x01      // R4 = 1
19 MOV R5, #0x01      // R5 = 1
20
21 ADDS R4, R4, R5   // R4 = 1 + 1 = 2, and resets carry flag
22 ADC R6, R5, #0     // R6 = R5 + carry = 1 + 0 = 1
23
24 LDR R3, =0x01
25 CMP R6, R3
26 BNE fail[         // Branch to fail if mismatch
27
```

The code editor has syntax highlighting for assembly instructions and comments. The bottom pane shows the "Build Output" window with the following text:

```
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_V5\ARM\ARMCLANG\Bin'
assembling Add.s...
"Add.s" - 0 Error(s), 0 Warning(s).
```

The status bar at the bottom right shows "L26 C17 CAP NUM SCRL OVR R/W".

Build:

C:\Users\hp\OneDrive\Documents\My_Assembly_project\My_Assembly_project.uvproj - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Project Target_1 startup_ARMCM4.c system_ARMCM4.c Add.s

110 };
111
112 #if defined (__GNUC__)
113 #pragma GCC diagnostic pop
114 #endif
115
116 /*-----
117 | Reset Handler called on controller reset
118 | *-----*/
119 NO_RETURN void Reset_Handler(void)
120 {
121 | SystemInit(); /* CMSIS System Initialization */
122 | _PROGRAM_START(); /* Enter PreMain (C library entry point) */
123 }
124
125
126 #if defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
127 #pragma clang diagnostic push
128 #pragma clang diagnostic ignored "-Wmissing-noreturn"
129 #endif
130
131 /*-----
132 | Hard Fault Handler
133 | *-----*/
134 void HardFault_Handler(void)
135 {
136 | while(1);

Build Output

Build started: Project: My_Assembly_project
** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
Build target: 'Target_1'
linking...
.RTE\Device\ARMCM4\ARMCM4.ac6.sct(19): warning: L6314W: No section matches pattern *(.bss.noinit).
Program Size: Code=892 RO-Data=976 RW-Data=4 ZI-Data=3584
Finished: 0 information, 1 warning and 0 error messages.
"\Objects\My_Assembly_project.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed: 00:00:00

Simulation L121 C1 CAP NUM SCRL OVR R/W

Execution:

C:\Users\hp\OneDrive\Documents\My_Assembly_project\My_Assembly_project.uvproj - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers Disassembly

Register	Value
Core	
R0	0x00000010
R1	0x00000020
R2	0x00000030
R3	0x00000001
R4	0x00000002
R5	0x00000001
R6	0x00000001
R7	0x00000048
R8	0x00000000
R9	0x00000000
R10	0x0000074C
R11	0x0000074C
R12	0x00000000
R13 (SP)	0x20040000
R14 (LR)	0x00000471
R15 (PC)	0x000004C4
xPSR	0x21000000

Banked System Internal

Mode Thread Privileged
Stack MSP States 278 Sec 0.00002317

FPU

Project Registers

startu ARMCM4.c system_ARMCM4.c Add.s

117 | Reset Handler called on controller reset
118 | *-----*/
119 NO_RETURN void Reset_Handler(void)
120 {
121 | SystemInit(); /* CMSIS System Initialization */
122 | _PROGRAM_START(); /* Enter PreMain (C library entry point) */
123 }
124
125
126 #if defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
127 #pragma clang diagnostic push
128 #pragma clang diagnostic ignored "-Wmissing-noreturn"
129 #endif
130
131 /*-----

Call Stack + Locals

Name	Location/Value	Type
0x00000000		

Call Stack + Locals Memory 1

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE

Simulation t1: 0.00002317 sec L121 C1 CAP NUM SCRL OVR R/W

Subtract_Multiply.s

Description

SUB: Subtract

SBC: Subtract with carry

RSB: Reverse subtracts

MUL: Multiply

Expected Test Result

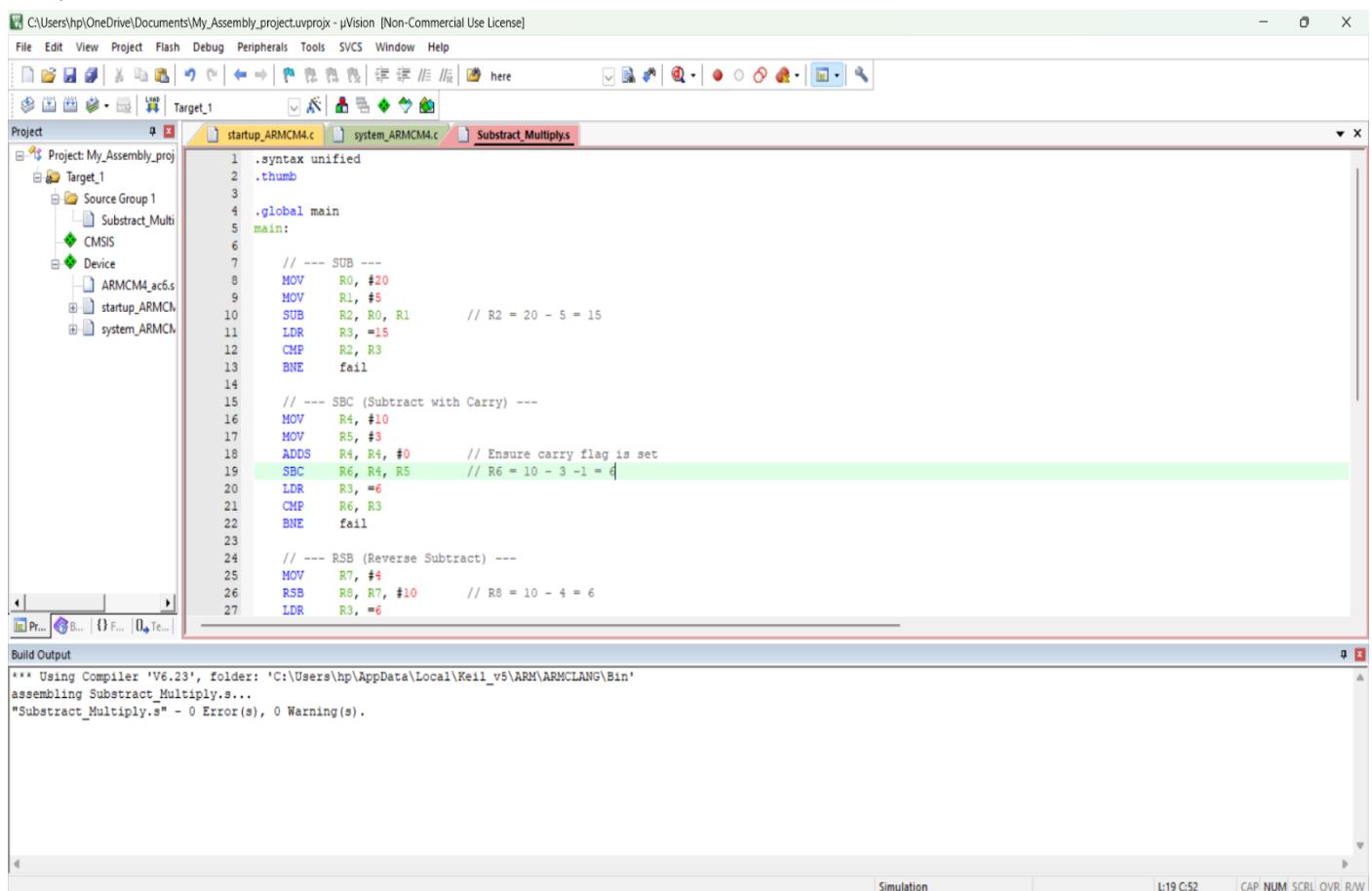
Check the value of R0:

0x01: success

0xFF: fail

Obtained Test Result

Compilation:



The screenshot shows the Keil uVision IDE interface with the assembly file `Subtract_Multiply.s` open. The assembly code is as follows:

```
1 .syntax unified
2 .thumb
3
4 .global main
5 main:
6
7 // --- SUB ---
8 MOV R0, #20
9 MOV R1, #5
10 SUB R2, R0, R1      // R2 = 20 - 5 = 15
11 LDR R3, =15
12 CMP R2, R3
13 BNE fail
14
15 // --- SBC (Subtract with Carry) ---
16 MOV R4, #10
17 MOV R5, #3
18 ADDS R4, R4, #0      // Ensure carry flag is set
19 SBC R6, R4, R5      // R6 = 10 - 3 - 1 = 4
20 LDR R3, =6
21 CMP R6, R3
22 BNE fail
23
24 // --- RSB (Reverse Subtract) ---
25 MOV R7, #4
26 RSB R8, R7, #10     // R8 = 10 - 4 = 6
27 LDR R3, =6
```

The assembly code performs three operations: subtraction (SUB), subtract with carry (SBC), and reverse subtraction (RSB). The assembly code is color-coded for readability, with comments in green.

Build Output:

```
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
assembling Subtract_Multiply.s...
"Subtract_Multiply.s" - 0 Error(s), 0 Warning(s).
```

Build:

The screenshot shows the Keil uVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, build, and simulation. The project tree on the left shows 'Project: My_Assembly_proj' with 'Target_1' selected, containing 'Source Group 1' with 'Subtract_Multi' and 'CMSIS', and 'Device' with 'ARMCM4_ac6.s', 'startup_ARMCM', and 'system_ARMCM'. The main editor window displays assembly code for 'Subtract_Multiply.s' (highlighted in red). The code implements subtraction and reverse subtraction using ARM instructions like MOV, SUB, CMP, BNE, ADDS, SBC, RSB, and LDR. The bottom status bar shows 'Build Output' and the build log.

```
syntax unified
.thumb
.global main
main:
    // --- SUB ---
    MOV R0, #20
    MOV R1, #5
    SUB R2, R0, R1      // R2 = 20 - 5 = 15
    LDR R3, =15
    CMP R2, R3
    BNE fail
    // --- SBC (Subtract with Carry) ---
    MOV R4, #10
    MOV R5, #3
    ADDS R4, R4, #0      // Ensure carry flag is set
    SBC R6, R4, R5      // R6 = 10 - 3 -1 = 6
    LDR R3, =6
    CMP R6, R3
    BNE fail
    // --- RSB (Reverse Subtract) ---
    MOV R7, #4
    RSB R8, R7, #10      // R8 = 10 - 4 = 6
    LDR R3, =6
```

Execution:

The screenshot shows the µVision IDE interface with several windows open:

- Registers**: Shows the ARM寄存器 (R0-R15, PC, PSR) values.
- Disassembly**: Displays assembly code for the startup_ARMCM4.c file, including the semihosting library function.
- Code Editor**: Shows the startup_ARMCM4.c source code, which includes CMSIS System Initialization code and a Reset Handler.
- Command Window**: Contains assembly commands like BS and ASSIGN.
- Call Stack + Locals**: Shows the current call stack and local variable information.

Move.s

Description

MOV: Move immediate value to register
MOVW: Move 16-bit constant (lower half)
MOVT: Move 16-bit constant (upper half)

Expected Test Result

Check the value of R3:

0x01: success

0xFF: fail

Obtained Test Result

Compilation:

The screenshot shows the Keil µVision IDE interface. The main window displays the assembly code for the file 'Moves.s'. The code includes comments explaining the use of MOV, MOVW, MOVT, and MOV with registers. It also includes a branch based on the result of a comparison between R2 and R4, setting R3 to 0x01 if successful or 0xFF if failed. The 'Build Output' window at the bottom shows the compiler command and the successful compilation with no errors or warnings.

```
1 .syntax unified
2 .thumb
3
4 .global main
5 main:
6     // MOV test (immediate)
7     MOV    R0, #0x12            // R0 = 0x00000012
8
9     // MOVW test (write lower 16 bits)
10    MOVW   R1, #0x5678         // R1 = 0x00005678
11
12    // MOVT test (write upper 16 bits)
13    MOVT   R1, #0x1234         // R1 = 0x12345678
14
15    // MOV with register
16    MOV    R2, R1              // R2 = 0x12345678
17
18    LDR    R4, =0x12345678    // Load full 32-bit constant into R4
19    CMP    R2, R4              // Now compare R2 with R4
20
21    // Optionally, branch based on result
22    BEQ    success
23    B      fail
24
25 success:
26     // Set a known value to indicate success
27     MOV    R3, #0x01
```

Build Output

```
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
assembling Move.s...
"Move.s" - 0 Error(s), 0 Warning(s).
```

Build:

The screenshot shows the Keil uVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Build. The Project Explorer on the left shows a hierarchy for 'My_Assembly_project' with 'Target_1' selected, containing 'Source Group 1' with 'Moves.s', 'CMSIS', 'Device', and 'ARMCM4_ac6.s'. The main editor window displays assembly code for 'startup_ARMCM4.ac6' with tabs for 'Moves.s' and 'system_ARMCM4.c'. The code includes instructions like MOV, MOVT, and CMP. Below the editor is the 'Build Output' window, which logs the build process, including linking and final program size information. The bottom status bar shows simulation and memory details.

```
1 .syntax unified
2 .thumb
3
4 .global main
5 main:
6     // MOV test (immediate)
7     MOV    R0, #0x12          // R0 = 0x00000012
8
9     // MOVW test (write lower 16 bits)
10    MOVW   R1, #0x5678        // R1 = 0x00005678
11
12    // MOVT test (write upper 16 bits)
13    MOVT   R1, #0x1234        // R1 = 0x12345678
14
15    // MOV with register
16    MOV    R2, R1            // R2 = 0x12345678
17
18    LDR    R4, =0x12345678    // Load full 32-bit constant into R4
19    CMP    R2, R4            // Now compare R2 with R4
20
21    // Optionally, branch based on result
22    BEQ    success
23    B      fail
24
25 success:
26     // Set a known value to indicate success
27     MOV    R3, #0x01
```

Build Output

```
Build started: Project: My_Assembly_project
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
Build target 'Target_1'
linking...
.RTE\Device\ARMCM4\ARMCM4_ac6.sct(19): warning: L6314W: No section matches pattern *(.bss.noinit).
Program Size: Code=868 RO-data=976 RW-data=4 ZI-data=3584
Finished: 0 information, 1 warning and 0 error messages.
".\Objects\My_Assembly_project.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed: 00:00:02
```

Execution:

The screenshot shows the Keil uVision IDE during execution. The top menu bar and toolbar are identical to the build screen. The 'Registers' window on the left shows the state of various ARM registers, including R0-R15 and xPSR. The 'Disassembly' window in the center shows the assembly code being executed, with the instruction at address 0x000004A8 highlighted. The assembly code includes CMSIS System Initialization and the entry point. The 'Memory 1' window at the bottom allows viewing memory contents at a specific address. The bottom status bar provides real-time simulation and memory usage information.

Registers

Register	Value
R0	0x00000012
R1	0x12345678
R2	0x12345678
R3	0x00000001
R4	0x12345678
R5	0x00000001
R6	0x0000001C
R7	0x00000723
R8	0x00000000
R9	0x00000000
R10	0x00000734
R11	0x00000734
R12	0x00000000
R13 (SP)	0x20040000
R14 (LR)	0x00000471
R15 (PC)	0x000004AB
xPSR	0x61000000

Disassembly

```
0x000004A8 E7FE B 0x000004A8
0x000004A9 0000 MOVS r0,r0
0x000004AC 5678 DCW 0x1234
0x000004AE 1234 DCW 0xB510
0x000004B0 B510 PUSH {r4,lr}
0x000004B2 4604 MOV r4,r0
0x000004B4 F3AF8000 NOP.W
0x000004B8 4620 MOV r0,r4
0x000004BA E8BD4010 POP {r4,lr}
0x000004BE F7FFFBDB B.W 0x00000478 __rt_exit
0x000004C2 4770 BX lr
0x000004C4 4901 LDR r1,[pc,#4] : @0x000004CC
0x000004C6 2018 MOVS r0,#0x18
0x000004C8 BEAB BKPT 0xAB
0x000004CA E7FE B 0x000004CA
0x000004CC 0026 DCW 0x0002
0x000004CE 0002 DCW 0x4602
0x000004D0 4602 MOV r2,r0
0x000004D2 1CC9 ADDS r1,r1,#3
0x000004D4 F0210103 BIC r1,r1,#0x03
0x000004D8 E9D20300 LDRD r0,r3,[r2,#0]
0x000004DC 1A1B SUBS r3,r3,r0
0x000004DE 428B CMP r3,r1
0x000004E0 D201 BCS 0x000004E8
```

startup_ARMCM4.c system_ARMCM4.c Moves.s

```
120 [
121 | SystemInit(); /* CMSIS System Initialization */
122 | __PROGRAM_START(); /* Enter PreMain (C library entry point) */
123 }
124
125 ]
```

Memory 1

Address: []

Call Stack + locals | Watch 1 | Memory 1

Simulation t1: 20.29945808 sec L121 C1 CAP NUM SCRL OVR R/W

Store.s

Description

STR: Store Register to Memory

LDR: Load Register from Memory

Expected Test Result

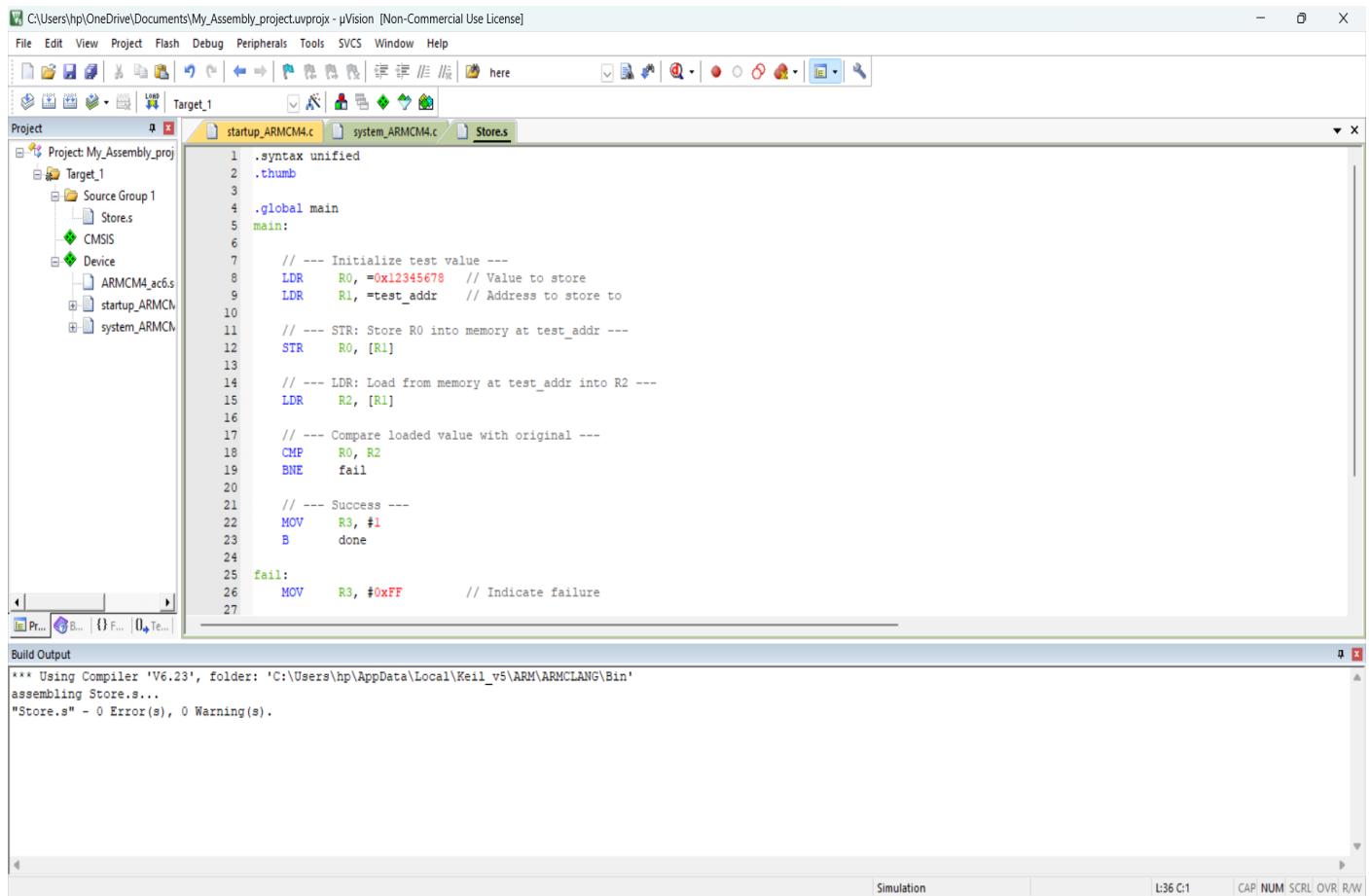
Check the value of R3:

0x01: success

0xFF: fail

Obtained Test Result

Compilation:



The screenshot shows the Keil uVision IDE interface. The main window displays the assembly code for the file 'Store.s'. The code implements a simple test routine using ARM assembly instructions. It initializes R0 with the value 0x12345678, stores it to memory at address test_addr, loads it back into R2, compares it with R0, and then sets R3 to 0x01 if they are equal, or 0xFF if they are not. The assembly code is as follows:

```
1 .syntax unified
2 .thumb
3
4 .global main
5 main:
6
7     // --- Initialize test value ---
8     LDR   R0, =0x12345678    // Value to store
9     LDR   R1, =test_addr      // Address to store to
10
11    // --- STR: Store R0 into memory at test_addr ---
12    STR   R0, [R1]
13
14    // --- LDR: Load from memory at test_addr into R2 ---
15    LDR   R2, [R1]
16
17    // --- Compare loaded value with original ---
18    CMP   R0, R2
19    BNE   fail
20
21    // --- Success ---
22    MOV   R3, #1
23    B    done
24
25 fail:
26    MOV   R3, #0xFF           // Indicate failure
27
```

The 'Build Output' window at the bottom shows the compilation process:

```
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
assembling Store.s...
"Store.s" - 0 Error(s), 0 Warning(s).
```

Build:

The screenshot shows the Keil µVision IDE interface during the build process of a project named 'My_Assembly_project'. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Build. The Project Explorer on the left shows the project structure with Target_1, Source Group 1, Stores, CMSIS, Device, and startup_ARMCM4_ac6.s. The main code editor displays assembly code for 'startup_ARMCM4.c' with comments explaining memory operations and comparisons. The bottom pane shows the 'Build Output' window with the following log:

```
Build started: Project: My_Assembly_project
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
Build target 'Target_1'
linking...
.RTE\Device\ARMCM4\ARMCM4.ac6.sct(19): warning: L6314W: No section matches pattern *(.bss.noinit).
Program Size: Code=860 RO-data=976 RW-data=8 ZI-data=3584
Finished: 0 information, 1 warning and 0 error messages.
".\Objects\My_Assembly_project.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed: 00:00:00
```

The status bar at the bottom right indicates L36 C1, CAP NUM SCRL OVR R/W.

Execution:

The screenshot shows the Keil µVision IDE during execution. The top menu bar and toolbar are identical to the build screen. The left sidebar displays the 'Registers' window, which lists various CPU registers with their current values. The main area features a 'Disassembly' window showing assembly instructions and their corresponding memory addresses. Below it is the source code editor for 'startup_ARMCM4.c', where the assembly code is mapped back to C language. A green arrow points to line 121, indicating the current instruction being executed. The bottom command line shows the command 'BS \My_Assembly_project\RTE\Device\ARMCM4\startup_ARMCM4.c\l21'. The status bar at the bottom right shows simulation parameters: t1: 26.08737125 sec, L121 C1, CAP NUM SCRL OVR R/W.

Bitwise.s

Description

AND: Bitwise AND

ORR: Bitwise OR

EOR: Bitwise Exclusive OR (XOR)

BIC: Bit Clear (AND with NOT)

MVN: Move NOT

Expected Test Result

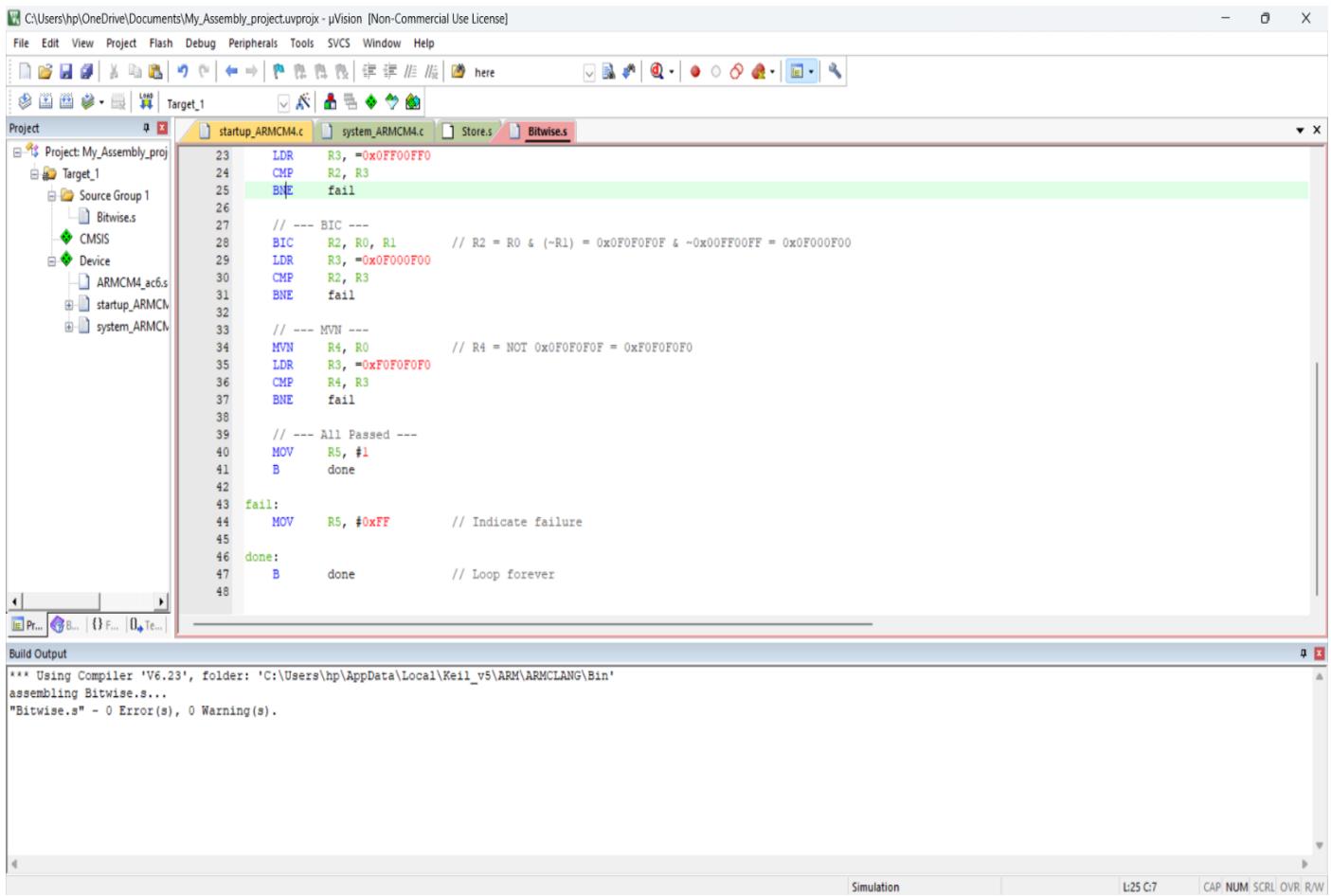
Check the value of R5:

0x01: success

0xFF: fail

Obtained Test Result

Compilation:



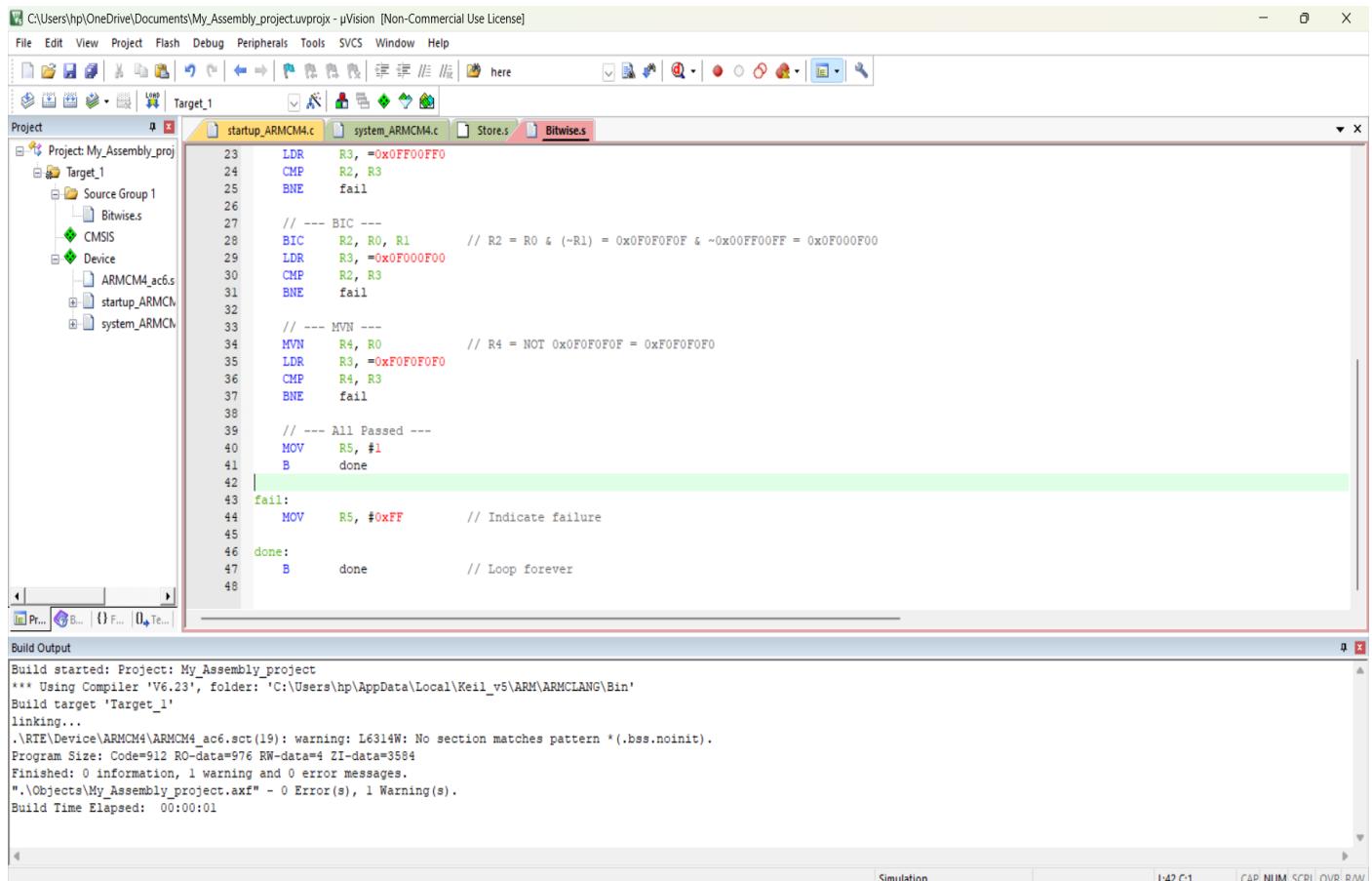
The screenshot shows the Keil µVision IDE interface with the assembly file 'Bitwise.s' open. The assembly code is as follows:

```
23 LDR R3, =0xFFFF00
24 CMP R2, R3
25 BNE fail
26
27 // --- BIC ---
28 BIC R2, R0, R1      // R2 = R0 & (~R1) = 0x0F0FOFOF & ~0x0FF00FF = 0x0F000F00
29 LDR R3, =0x0F000F00
30 CMP R2, R3
31 BNE fail
32
33 // --- MVN ---
34 MVN R4, R0          // R4 = NOT 0x0F0FOFOF = 0xF0F0F0FO
35 LDR R3, =0xF0F0F0FO
36 CMP R4, R3
37 BNE fail
38
39 // --- All Passed ---
40 MOV R5, #1
41 B done
42
43 fail:
44 MOV R5, #0xFF        // Indicate failure
45
46 done:
47 B done              // Loop forever
48
```

The 'Build Output' window at the bottom shows the compilation results:

```
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
assembling Bitwise.s...
"Bitwise.s" - 0 Error(s), 0 Warning(s).
```

Build:



```

23    LDR   R3, =0xFFFF00FF
24    CMP   R2, R3
25    BNE   fail
26
27    // --- BIC ---
28    BIC   R2, R0, R1      // R2 = R0 & (~R1) = 0x0F0FOFOF & ~0x0FFF00FF = 0x0F000F00
29    LDR   R3, =0x0F000F00
30    CMP   R2, R3
31    BNE   fail
32
33    // --- MVN ---
34    MVN   R4, R0          // R4 = NOT 0x0F0FOFOF = 0xF0F0FOFO
35    LDR   R3, =0xF0F0FOFO
36    CMP   R4, R3
37    BNE   fail
38
39    // --- All Passed ---
40    MOV   R5, #1
41    B     done
42
43 fail:
44    MOV   R5, #0xFF        // Indicate failure
45
46 done:
47    B     done            // Loop forever
48

```

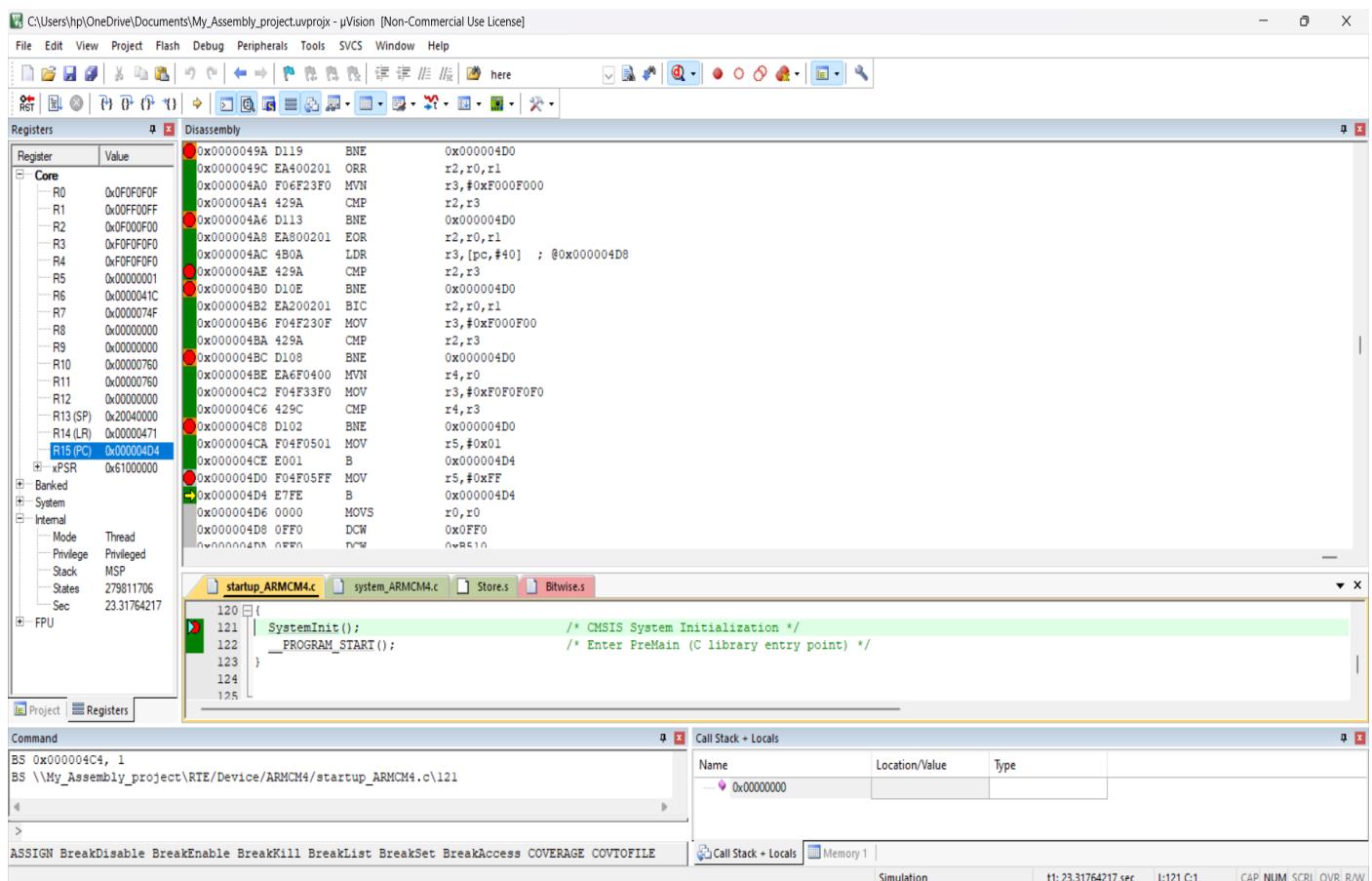
Build Output:

```

Build started: Project: My_Assembly_project
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
Build target 'Target_1'
linking...
.RTE\Device\ARMCM4\ARMCM4.ac6.sct(19): warning: L6314W: No section matches pattern *(.bss.noinit).
Program Size: Code=912 RO-data=976 RW-data=4 ZI-data=3584
Finished: 0 information, 1 warning and 0 error messages.
"\Objects\My_Assembly_project.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed: 00:00:01

```

Execution:



Registers:

Register	Value
R0	0x0F0FOFOF
R1	0x0FFFOFFF
R2	0x0F0FOFO0
R3	0x0F0FOFO0
R4	0x0F0FOFO0
R5	0x00000001
R6	0x0000041C
R7	0x0000074F
R8	0x00000000
R9	0x00000000
R10	0x00000760
R11	0x00000760
R12	0x00000000
R13 (SP)	0x20040000
R14 (LR)	0x00000471
R15 (PC)	0x000004D4
xPSR	0x61000000

Disassembly:

```

120  {
121  | SystemInit();           /* CMSIS System Initialization */
122  | __PROGRAM_START();      /* Enter PreMain (C library entry point) */
123  }
124
125

```

Call Stack + Locals:

Name	Location/Value	Type
0x00000000		

Shift_Rotate.s

Description

LSL: Logical Shift Left

ASR: Arithmetic Shift Right

ROR: Rotate Right

RRX: Rotate Right with Extend (through carry)

Expected Test Result

Check the value of R5:

0x01: success

0xFF: fail

Obtained Test Result

Compilation:

The screenshot shows the Keil uVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar below has various icons for file operations like Open, Save, and Build. The main workspace shows a project tree for 'My_Assembly_proj' with Target_1 selected. Under Target_1, there are Source Group 1, Shift_Rotates, CMSIS, and Device sections. The Device section contains ARMCM4_ac6.s, startup_ARMCM4, and system_ARMCM4. The assembly code for 'Shift_Rotate.s' is displayed in the center editor window, starting with .syntax unified and .thumb. It includes three sections: LSL, ASR, and ROR, each performing a shift operation on R0 and comparing the result with R2. The build output at the bottom shows the compiler version (V6.23), the assembly folder, and a message indicating 0 errors and 0 warnings.

```
1 .syntax unified
2 .thumb
3
4 .global main
5 main:
6
7     // --- LSL: Logical Shift Left ---
8     MOV    R0, #0x01
9     LSL    R1, R0, #3      // R1 = 0x01 << 3 = 0x08
10    LDR   R2, =0x08
11    CMP    R1, R2
12    BNE    fail
13
14     // --- ASR: Arithmetic Shift Right ---
15    LDR   R0, =0xF0000000  // A negative value (signed)
16    ASR   R1, R0, #4      // Arithmetic shift: preserves sign = 0xFF000000
17    LDR   R2, =0xFF000000
18    CMP    R1, R2
19    BNE    fail
20
21     // --- ROR: Rotate Right ---
22    LDR   R0, =0x80000001
23    ROR   R1, R0, #1      // R1 = 0xC0000000
24    LDR   R2, =0xC0000000
25    CMP    R1, R2
26    BNE    fail
27
```

Build Output:
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
assembling Shift_Rotate.s...
"Shift_Rotate.s" - 0 Error(s), 0 Warning(s).

Build:

C:\Users\hp\OneDrive\Documents\My_Assembly_project.uvproj - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Project Target_1 startup_ARMCM4.c system_ARMCM4.c Shift_Rotate.s

110 };
111
112 #if defined (__GNUC__)
113 #pragma GCC diagnostic pop
114 #endif
115
116 /*-----
117 Reset Handler called on controller reset
118 -----*/
119 _NO_RETURN void Reset_Handler(void)
120 {
121 SystemInit(); /* CMSIS System Initialization */
122 __PROGRAM_START(); /* Enter PreMain (C library entry point) */
123 }
124
125
126 #if defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
127 #pragma clang diagnostic push
128 #pragma clang diagnostic ignored "-Wmissing-noreturn"
129 #endif
130
131 /*-----
132 Hard Fault Handler
133 -----*/
134 void HardFault_Handler(void)
135 {
136 while(1);

Build Output

```
Build started: Project: My_Assembly_project
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
Build target: 'Target_1'
assembling Shift_Rotate.s...
linking...
.RTE\Device\ARMCM4\ARMCM4_ac6.sct(19): warning: L6314W: No section matches pattern *(.bss.noinit).
Program Size: Code=916 RO-data=976 RW-data=4 ZI-data=3584
Finished: 0 information, 1 warning and 0 error messages.
"\Objects\My_Assembly_project.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed: 00:00:04
```

Simulation L121 C1 CAP NUM SCRL OVR R/W

Execution:

C:\Users\hp\OneDrive\Documents\My_Assembly_project.uvproj - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers Disassembly

Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x80000000
R4	0xC0000000
R5	0x00000001
R6	0x00000041C
R7	0x00000753
R8	0x00000000
R9	0x00000000
R10	0x00000764
R11	0x00000764
R12	0x00000000
R13 (SP)	0x20040000
R14 (LR)	0x000004CE
R15 (PC)	0x000000DA
xPSR	0x10000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	624775200
Sec	52.06460000
FPU	

startup_ARMCM4.c system_ARMCM4.c Shift_Rotate.s

```
120 {  
121     SystemInit(); /* CMSIS System Initialization */  
122     __PROGRAM_START(); /* Enter PreMain (C library entry point) */  
123 }  
124  
125
```

Call Stack + Locals

Name	Location/Value	Type
0x00000000		

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE

Call Stack + Locals Memory 1 Simulation t1: 52.06460000 sec L121 C1 CAP NUM SCRL OVR R/W

Compare.s

Description

CMP: Compare (subtract, sets flags)

CMN: Compare Negative (add, sets flags)

TST: Bitwise AND (sets flags)

Expected Test Result

Check the value of R6:

0x01: success

0xFF: fail

Obtained Test Result

Compilation:

The screenshot shows the Keil µVision IDE interface. The main window displays the assembly code for the file `Compare.s`. The code implements three comparison operations: CMP, CMN, and TST, and checks the result in register R6. The assembly code is as follows:

```
1 .syntax unified
2 .thumb
3
4 .global main
5 main:
6
7     // --- CMP: Compare R0 - R1 ---
8     MOV    R0, #5
9     MOV    R1, #5
10    CMP   R0, R1           // Expect Zero flag (Z=1), because 5 - 5 = 0
11    BNE   fail            // If not equal, fail
12
13    // --- CMN: Compare Negative (R2 + R3) ---
14    MOV    R2, #0x7FFFFFFF
15    MOV    R3, #1
16    CMN   R2, R3           // Expect overflow (V=1), R2 + R3 = 0x80000000 (signed overflow)
17    BVC   fail            // If V (overflow) not set, fail
18
19    // --- TST: Bitwise AND test ---
20    MOV    R4, #0b10101010
21    MOV    R5, #0b00001000
22    TST   R4, R5           // Should not set zero (0b1000 is set in R4)
23    BEQ   fail            // If result is 0, then fail
24
25    // --- All Passed ---
26    MOV    R6, #1
27    B     done
```

The bottom window shows the "Build Output" with the following message:

```
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
assembling Compare.s...
"Compare.s" - 0 Error(s), 0 Warning(s).
```

Build:

C:\Users\hp\OneDrive\Documents\My_Assembly_project.uvproj - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Target_1 startup_ARMCM4.c system_ARMCM4.c Shift_Rotate.s Compare.s

Project: My_Assembly_proj

Target_1

Source Group 1

Compar.s

CMSIS

Device

ARMCM4_ac6.s

startup_ARMCM4.c

system_ARMCM4.c

```

110 );
111
112 #if defined ( __GNUC__ )
113 #pragma GCC diagnostic pop
114 #endif
115
116 /*-----*
117 | Reset Handler called on controller reset
118 | *-----*/
119 __NO_RETURN void Reset_Handler(void)
120 {
121     | SystemInit();           /* CMSIS System Initialization */
122     | __PROGRAM_START();      /* Enter PreMain (C library entry point) */
123 }
124
125
126 #if defined(_ARMCC_VERSION) && (_ARMCC_VERSION >= 6010050)
127 #pragma clang diagnostic push
128 #pragma clang diagnostic ignored "-Wmissing-noreturn"
129 #endif
130
131 /*-----*
132 | Hard Fault Handler
133 | *-----*/
134 void HardFault_Handler(void)
135 {
136     while(1);

```

Build Output

```

Build started: Project: My_Assembly_project
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
Build target 'Target_1'
linking...
.RTE\Device\ARMCM4\ARMCM4_ac6.sct(19): warning: L6314W: No section matches pattern *(.bss.noinit).
Program Size: Code=876 RO-data=976 RW-data=4 ZI-data=3584
Finished: 0 information, 1 warning and 0 error messages.
".\Objects\My_Assembly_project.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed: 00:00:00

```

Simulation L121 C1 CAP NUM SCRL OVR R/W

Execution:

C:\Users\hp\OneDrive\Documents\My_Assembly_project.uvproj - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers Disassembly

Register	Value
R0	0x00000005
R1	0x00000005
R2	0xFFFFFFFF
R3	0x00000001
R4	0x000000AA
R5	0x00000008
R6	0x00000001
R7	0x0000007B
R8	0x00000000
R9	0x00000000
R10	0x0000007C
R11	0x0000007C
R12	0x00000000
R13 (SP)	0x20040000
R14 (LR)	0x000000471
R15 (PC)	0x0000004B6
xPSR	0x11000000

Banked

System

Internal

Mode Thread

Privilege Privileged

Stack MSP

States 276381582

Sec 23.03179850

FPU

startup_ARMCM4.c system_ARMCM4.c Shift_Rotate.s Compare.s

```

120 {
121     | SystemInit();           /* CMSIS System Initialization */
122     | __PROGRAM_START();      /* Enter PreMain (C library entry point) */
123 }
124
125

```

Command Call Stack + Locals

Name	Location/Value	Type
0x00000000		

Memory 1

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE

Call Stack + Locals Memory 1

Simulation t1: 23.03179850 sec L121 C1 CAP NUM SCRL OVR R/W

Stack.s

Description

PUSH: Store multiple registers to the stack

POP: Restore multiple registers from the stack

STMIA: Store Multiple Increment After (to memory)

LDMIA: Load Multiple Increment After (from memory)

Expected Test Result

Check the value of R0:

0x01: success

0xFF: fail

Obtained Test Result

Compilation:

The screenshot shows the Keil µVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar below has various icons for file operations like Open, Save, and Build. The main workspace shows a project tree for 'My_Assembly_proj' with 'Target_1' selected. Under 'Target_1', there are 'Source Group 1' and 'Stacks'. The 'Stacks' folder contains 'CMSIS' and 'Device' sub-folders, which further contain 'ARMCM4_ac6.s', 'startup_ARMCM4.s', and 'system_ARMCM4.s'. The assembly code for 'Stacks.s' is displayed in the center pane:

```
46 LDR R7, =0x55555555
47 CMP R5, R7
48 BNE fail
49
50 ADD R0, R0, #1           // STMIA/LDMIA passed
51
52 // Final success check (2 tests)
53 MOV R1, #2
54 CMP R0, R1
55 BNE fail
56
57 MOV R0, #1           // All tests passed
58 B done
59
60 fail:
61 MOV R0, #0xFF          // Failure
62
63 done:
64 B done |               // Infinite loop
65
66 // --- Memory buffer for STMIA/LDMIA ---
67 .data
68 .align 4
69 test_buffer:
70     .word 0, 0
71
```

The code uses ARM assembly instructions to perform stack operations and compare register values. It includes comments explaining the purpose of each section: final success check, failure handling, and an infinite loop. The bottom pane shows the 'Build Output' window with the following message:

```
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
assembling Stack.s...
"Stack.s" - 0 Error(s), 0 Warning(s).
```

Build:

The screenshot shows the Keil uVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Build. The Project Explorer on the left shows a project named 'My_Assembly_project' with a target 'Target_1' containing source groups for CMSIS and Device, and specific files like 'ARMCM4_ac6.s', 'startup.ARMCM4', and 'system.ARMCM4'. The main code editor window displays assembly code for 'startup.ARMCM4.c' with tabs for 'startup_ARMCM4.c' and 'system_ARMCM4.c'. The assembly code includes comments for STMIA/LDMIA passes, failure checks, and an infinite loop at the end. The bottom window shows the 'Build Output' with details of the build process, compiler version ('V6.23'), target ('Target_1'), linking information, and a warning about a section mismatch. The total program size is 976 bytes.

```
46 LDR R7, #0x55555555
47 CMP R5, R7
48 BNE fail
49
50 ADD R0, R0, #1          // STMIA/LDMIA passed
51
52 // Final success check (2 tests)
53 MOV R1, #2
54 CMP R0, R1
55 BNE fail
56
57 MOV R0, #1          // All tests passed
58 B done
59
60 fail:
61 MOV R0, #0xFF          // Failure
62
63 done:
64 B done                // Infinite loop
65
66 // --- Memory buffer for STMIA/LDMIA ---
67 .data
68 .align 4
69 test_buffer:
70 .word 0, 0
71
```

Build Output

```
Build started: Project: My_Assembly_project
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
Build target 'Target_1'
linking...
.RTE\Device\ARMCM4\ARMCM4_ac6.sct(19): warning: L6314W: No section matches pattern *(.bss.noinit).
Program Size: Code=940 RO-data=976 RW-data=12 ZI-data=3584
Finished: 0 information, 1 warning and 0 error messages.
".\Objects\My_Assembly_project.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed: 00:00:00
```

Execution:

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

- Registers pane:** Displays the ARM register values. The R13 (SP) register is highlighted in yellow.
- Disassembly pane:** Shows the assembly code for the startup routine. The current instruction at address 0x0000004F2 is `E7FE`.
- Stack pane:** Displays the stack contents starting with `SystemInit();` and `__PROGRAM_START();`
- Call Stack + Locals pane:** Shows the call stack and local variables, currently empty.
- Command pane:** Displays the command line history and current project status.

Floating.s

Description

VMOV: Move values between registers or between core and FPU

VADD.F32: Floating-point addition

VSUB.F32: Floating-point subtraction

VMUL.F32: Floating-point multiplication

VDIV.F32: Floating-point division

VCMP.F32: Floating-point compare

Expected Test Result

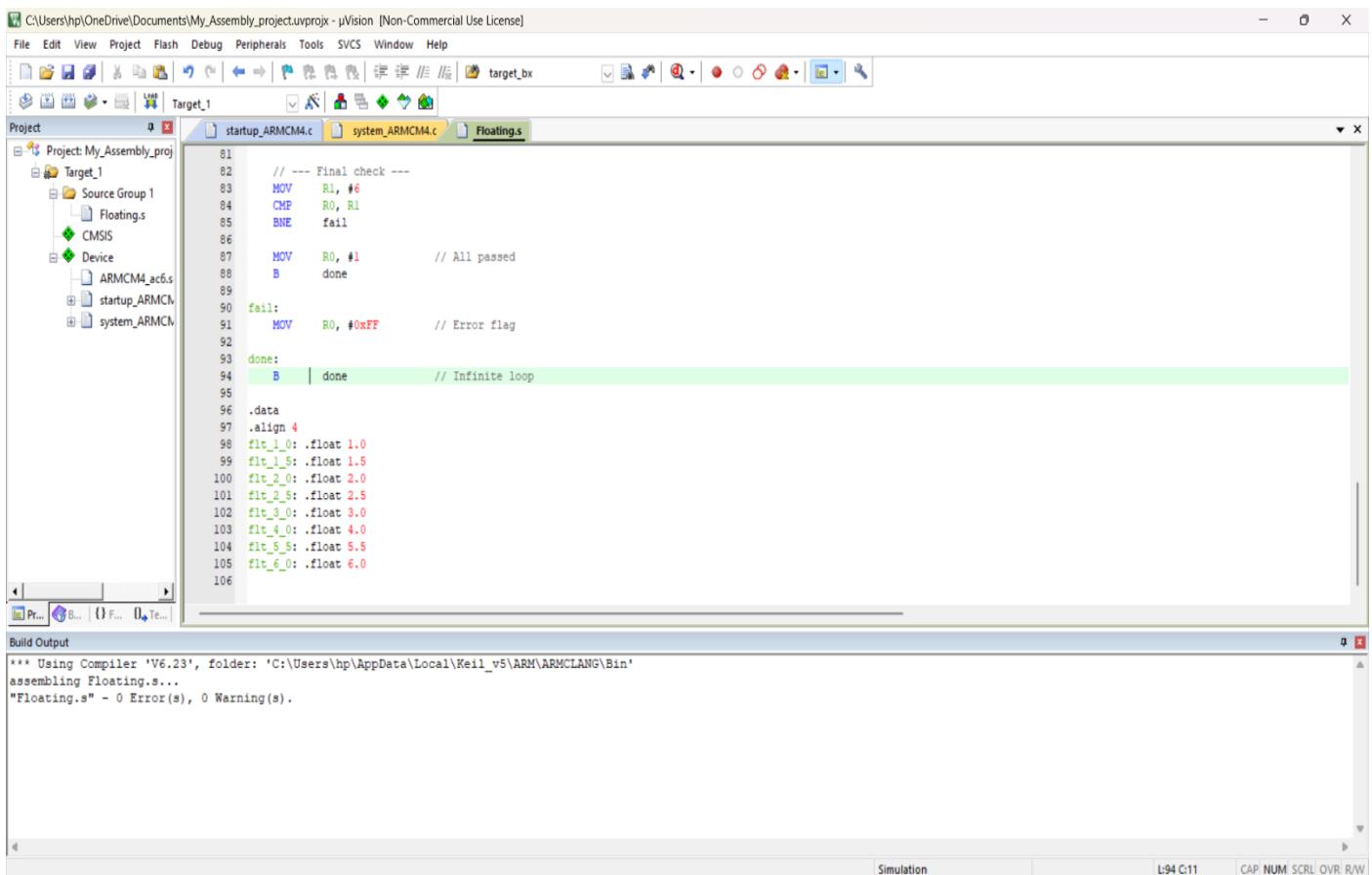
Check the value of R0:

0x01: success

0xFF: fail

Obtained Test Result

Compilation:



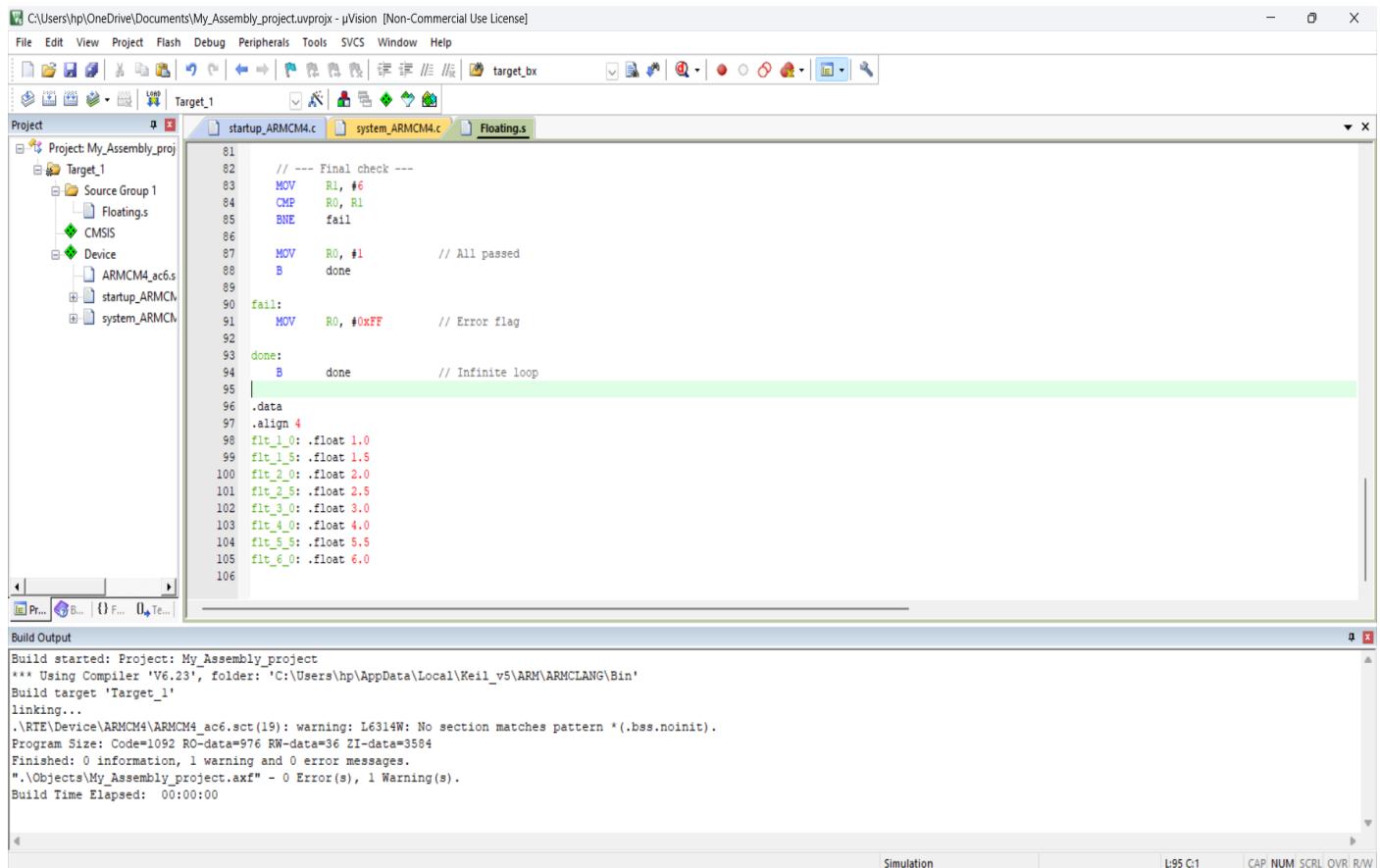
The screenshot shows the Keil µVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar below has various icons for file operations like Open, Save, and Build. The main workspace shows a project tree for 'My_Assembly_proj' with 'Target_1' selected. Under 'Source Group 1', there is a file named 'Floating.s'. The assembly code for 'Floating.s' is displayed in the center pane:

```
81      // --- Final check ---
82      MOV   R1, #6
83      CMP   R0, R1
84      BNE   fail
85
86      MOV   R0, #1           // All passed
87      B    done
88
89
90      fail:
91      MOV   R0, #0xFF        // Error flag
92
93      done:
94      B    done             // Infinite loop
95
96      .data
97      .align 4
98      fit_1_0: .float 1.0
99      fit_1_5: .float 1.5
100     fit_2_0: .float 2.0
101     fit_2_5: .float 2.5
102     fit_3_0: .float 3.0
103     fit_4_0: .float 4.0
104     fit_5_5: .float 5.5
105     fit_6_0: .float 6.0
106
```

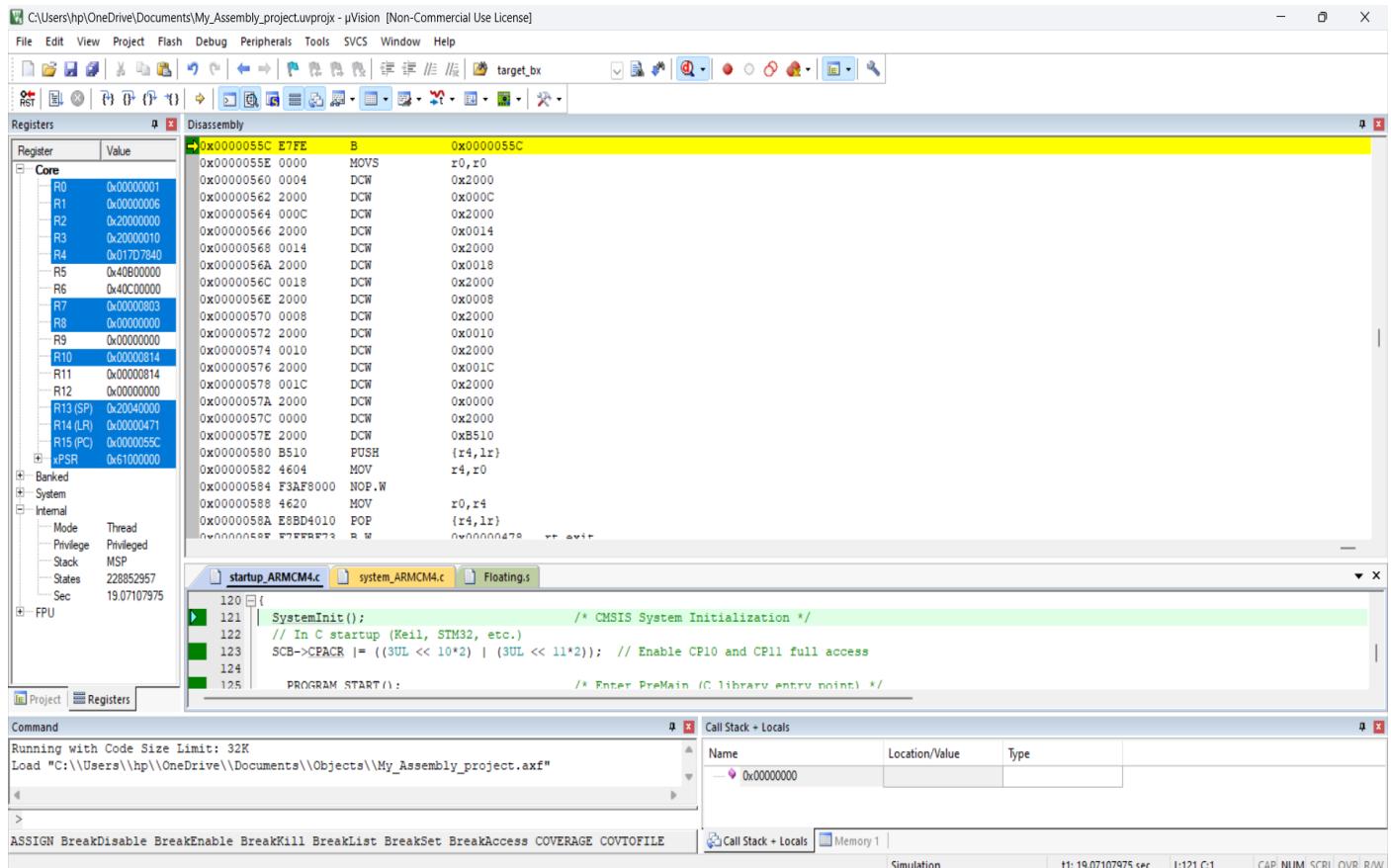
The bottom pane shows the 'Build Output' window with the following text:

```
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
assembling Floating.s...
"Floating.s" - 0 Error(s), 0 Warning(s).
```

Build:



Execution:



Control_Flow.s

Description

CBZ: If the register value is zero, branch to the specified label.

CBNZ: If the register value is non-zero, branch to the specified label.

IT: Conditional execution of the following instructions based on a condition.

Expected Test Result

Check the value of R0:

0x01: success

0xFF: fail

Obtained Test Result

Compilation:

The screenshot shows the Keil µVision IDE interface. The assembly code for `Control_Flow.s` is displayed in the main editor window. The code includes comments explaining the purpose of each instruction: initializing R0 to 0, performing CBZ and CBNZ checks, and demonstrating IT conditional execution. The build output window at the bottom shows the compiler version, the file being assembled, and that there were no errors or warnings.

```
2 .thumb
3 .thumb_func
4 .global main
5
6 main:
7   MOV   R0, #0      // Initialize R0 to 0, used for CBZ and CBNZ checks
8
9   // --- CBZ: Compare and Branch if Zero ---
10  MOV   R1, #0     // Set R1 to 0
11  CBZ   R1, label_cbz // Branch if R1 is zero
12  B    fail        // Should not branch here if CBZ works
13 label_cbz:
14  ADD   R0, R0, #1  // Increment R0 if CBZ worked correctly
15
16 // --- CBNZ: Compare and Branch if Non-Zero ---
17  MOV   R1, #5     // Set R1 to non-zero value
18  CBNZ  R1, label_cbnz // Branch if R1 is non-zero
19  B    fail        // Should not branch here if CBNZ works
20 label_cbnz:
21  ADD   R0, R0, #1  // Increment R0 if CBNZ worked correctly
22
23 // --- IT: Conditional Execution ---
24  MOVS  R2, #1     // Set R2 to 1
25  CMP   R2, #1     // Compare R2 to 1
26  IT    EQ         // If Equal, execute following instruction
27  ADDEQ R0, R0, #1 // Increment R0 if R2 == 1
28
```

Build Output

```
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
assembling Control_Flow.s...
"Control_Flow.s" - 0 Error(s), 0 Warning(s).
```

Build:

The screenshot shows the Keil uVision IDE interface. The project tree on the left shows a single target named 'Target_1' with source files 'Control_Flows.s' and 'startup_ARMCM4.c'. The assembly code for 'startup_ARMCM4.c' is displayed in the main window, demonstrating various ARM assembly instructions like MOV, CBZ, CBNZ, IT, and BKPT. The build output window at the bottom shows the compilation process, linking, and final statistics.

```
2 .thumb
3 .thumb_func
4 .global main
5
6 main:
7     MOV    R0, #0          // Initialize R0 to 0, used for CBZ and CBNZ checks
8
9     // --- CBZ: Compare and Branch if Zero ---
10    MOV   R1, #0           // Set R1 to 0
11    CBZ  R1, label_cbz // Branch if R1 is zero
12    B    fail             // Should not branch here if CBZ works
13 label_cbz:
14    ADD   R0, R0, #1       // Increment R0 if CBZ worked correctly
15
16    // --- CBNZ: Compare and Branch if Non-Zero ---
17    MOV   R1, #5           // Set R1 to non-zero value
18    CBNZ R1, label_cbnz // Branch if R1 is non-zero
19    B    fail             // Should not branch here if CBNZ works
20 label_cbnz:
21    ADD   R0, R0, #1       // Increment R0 if CBNZ worked correctly
22
23    // --- IT: Conditional Execution ---
24    MOVS R2, #1           // Set R2 to 1
25    CMP   R2, #1           // Compare R2 to 1
26    IT    EQ              // If Equal, execute following instruction
27    ADDEQ R0, R0, #1      // Increment R0 if R2 == 1
28
```

Build Output:

```
Build started: Project: My_Assembly_project
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
Build target 'Target_1'
linking...
.\RTEDevice\ARMCM4\ARMCM4_ac6.sct(19): warning: L6314W: No section matches pattern *(.bss.noinit).
Program Size: Code=884 RO-data=976 RW-data=4 ZI-data=3584
Finished: 0 information, 1 warning and 0 error messages.
"\Objects\My_Assembly_project.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed: 00:00:00
```

Execution:

The screenshot shows the Keil uVision IDE during execution. The 'Registers' window on the left displays the state of various ARM registers (R0-R15, SP, LR, PC, xPSR) with their current values. The 'Disassembly' window in the center shows the assembly code being executed, with the instruction pointer (IP) highlighting the current instruction. The assembly code includes CMSIS System Initialization and the entry point __PROGRAM_START(). The 'Control_Flows' tab in the bottom right shows the control flow graph. The 'Command' window at the bottom shows the command history, and the 'Call Stack + Locals' window shows the current call stack and local variable values.

Registers

Register	Value
R0	0x00000001
R1	0x00000005
R2	0x00000001
R3	0x00000003
R4	0x17D7840
R5	0x00000001
R6	0x0000001C
R7	0x00000073
R8	0x00000000
R9	0x00000000
R10	0x00000074
R11	0x00000074
R12	0x00000000
R13 (SP)	0x20040000
R14 (LR)	0x00000071
R15 (PC)	0x000000BE
xPSR	0x61000000

Disassembly

```
0x0000004BE E7FE B 0x0000004BE
0x0000004C0 B510 PUSH {r4,lr}
0x0000004C2 4604 MOV r4,r0
0x0000004C4 F3AF8000 NOP.W
0x0000004C8 4620 MOV r0,r4
0x0000004CA E8BD4010 POP {r4,lr}
0x0000004CE F7FFBD3 B,W 0x000000478 __rt_exit
0x0000004D2 4770 BX lr
0x0000004D4 4901 LDR r1,[pc,#4] : @0x000004DC
0x0000004D6 2018 MOVS r0,#0x18
0x0000004D8 BEAB BKPT 0xAB
0x0000004DA E7FE B 0x0000004DA
0x0000004DC 0026 DCW 0x00002
0x0000004DE 0002 DCW 0x4602
0x0000004E0 4602 MOV r2,r0
0x0000004E2 1CC9 ADDS r1,r1,#3
0x0000004E4 F0210103 BIC r1,r1,#0x03
0x0000004E8 E9D20300 LDRD r0,r3,[r2,#0]
0x0000004EC 1A1B SUBS r3,r3,r0
0x0000004EE 428B CMP r3,r1
0x0000004F0 D201 BCS 0x0000004F6
0x0000004F2 2000 MOVS r0,#0x00
0x0000004F4 4770 BX lr
0x0000004F6 4401 NNN r1,r1,r0
```

Control_Flows

```
120 {
121     | SystemInit();
122     | /* CMSIS System Initialization */
123     | __PROGRAM_START(); /* Enter PreMain (C library entry point) */
124 }
125
```

Call Stack + Locals

Name	Location/Value	Type
0x00000000		

Branching.s

Description

B: Unconditional branch

BL: Branch with link (calls a subroutine)

Expected Test Result

Check the value of R5 or R0:

0x01: only B passed

0x02: only BL passed

0x03: both B and BL executed correctly

Obtained Test Result

Compilation:

The screenshot shows the Keil µVision IDE interface. The main window displays the assembly code for the file `Branching.s`. The code contains several test cases for branching instructions. Lines 23 and 28 are highlighted in green, indicating they were executed. The build output window at the bottom shows the compiler's message: "*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin' assembling Branching.s... "Branching.s" - 0 Error(s), 0 Warning(s)." The status bar at the bottom right shows "L:28 C:27".

```
8      // --- Test B ---
9      B      test_b
10     after_b:
11
12     // --- Test BL ---
13     BL      test_bl
14     after_bl:
15
16     // Copy final result to R0 for easy inspection
17     MOV    R0, R5
18     B      done
19
20
21 // === Target for B ===
22 test_b:
23     ORR    R5, R5, #0x01 // Set bit 0 (B passed)
24     B      after_b
25
26 // === Target for BL ===
27 test_bl:
28     ORR    R5, R5, #0x02 | // Set bit 1 (BL passed)
29     BX     LR
30
31 done:
32     B      done        // Infinite loop
33
```

Build Output:
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
assembling Branching.s...
"Branching.s" - 0 Error(s), 0 Warning(s).

Build:

C:\Users\hp\OneDrive\Documents\My_Assembly_project.uvproj - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Project: My_Assembly_project Target_1 startup_ARMCM4.c system_ARMCM4.c Branching.s

```
8 // --- Test B ---
9     B test_b
10    after_b:
11
12 // --- Test BL ---
13     BL test_bl
14    after_bl:
15
16 // Copy final result to R0 for easy inspection
17     MOV R0, R5
18     B done
19
20 // === Target for B ===
21 test_b:
22     ORR R5, R5, #0x01 // Set bit 0 (B passed)
23     B after_b
24
25 // === Target for BL ===
26 test_bl:
27     ORR R5, R5, #0x02 // Set bit 1 (BL passed)
28     BX LR
29
30
31 done:
32     B done           // Infinite loop
33
```

Build Output

```
Build started: Project: My_Assembly_project
*** Using Compiler 'V6.23', folder: 'C:\Users\hp\AppData\Local\Keil_v5\ARM\ARMCLANG\Bin'
Build target 'Target_1'
linking...
.LTTEDevice\ARMCM4\ARMCM4_ac6.sct(19): warning: L6314W: No section matches pattern *(.bss.noinit).
Program Size: Code=856 RO-data=976 RW-data=4 ZI-data=3584
Finished: 0 information, 1 warning and 0 error messages.
".\Objects\My_Assembly_project.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed: 00:00:00
```

Simulation L33 C1 CAP NUM SCRL OVR R/W

Execution:

C:\Users\hp\OneDrive\Documents\My_Assembly_project.uvproj - µVision [Non-Commercial Use License]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers Disassembly

Register	Value
R0	0x00000003
R1	0x20000008
R2	0x20000008
R3	0x00000000
R4	0x17D7840
R5	0x00000003
R6	0x00000041C
R7	0x000000717
R8	0x00000000
R9	0x00000000
R10	0x000000728
R11	0x000000728
R12	0x00000000
R13 (SP)	0x20040000
R14 (LR)	0x000000493
R15 (PC)	0x0000004A2
xPSR	0x10000000

Disassembly

```
0x0000004A2 E7FE B 0x0000004A2
0x0000004A4 B510 PUSH (r4,lr)
0x0000004A6 4604 MOV r4,r0
0x0000004A8 F3AF8000 NOP.W
0x0000004AC 4620 MOV r0,r4
0x0000004AE E8BD4010 POP (r4,lr)
0x0000004B2 F7FFBFF1 B.W 0x000000478 __rt_exit
0x0000004B6 4770 BX lr
0x0000004B8 4901 LDR r1,[pc,#4] : @0x0000004C0
0x0000004B9 2018 MOVS r0,#0x18
0x0000004BC BEAB BKPT 0xAB
0x0000004BE E7FE B 0x0000004BE
0x0000004C0 0026 DCW 0x00002
0x0000004C2 0002 DCW 0x4602
0x0000004C4 4602 MOV r2,r0
0x0000004C6 1CC9 ADDS r1,r1,#3
0x0000004C8 F0210103 BIC r1,r1,#0x03
0x0000004CC E9D20300 LDRD r0,r3,[r2,$0]
0x0000004D0 1A1B SUBS r3,r3,r0
0x0000004D2 428B CMP r3,r1
0x0000004D4 D201 BCS 0x0000004DA
0x0000004D6 2000 MOVS r0,#0x00
0x0000004D8 4770 BX lr
0x0000004D9 4401 rnm +1 +1 +0
```

Branching.s

```
120 [
121     | SystemInit();
122     |__PROGRAM_START(); /* Enter PreMain (C library entry point) */
123 }
124
125
```

Memory 1

Address: []

Call Stack + Locals Watch 1 Memory 1

Simulation t1: 14.78406275 sec L121 C1 CAP NUM SCRL OVR R/W

Conclusion

By conducting this testing, it is ensured that the Proof-of-Concept (PoC) code functions as intended for the majority of instruction sets across various types within the Cortex-M processor series. This approach can be further extended to cover the remaining instruction sets as well.

Looking ahead, the same underlying logic can be adapted for other processor architectures, enabling validation of enhanced assembly code. This would contribute to improved safety and increased reliability in microcontroller operations.