

# **Tutorial 07: ARM Subroutine Call and Return**

*Computer Science Department*

*CS2208b: Introduction to Computer Organization and Architecture*

*Winter 2019*

*Instructor: Mahmoud R. El-Sakka*

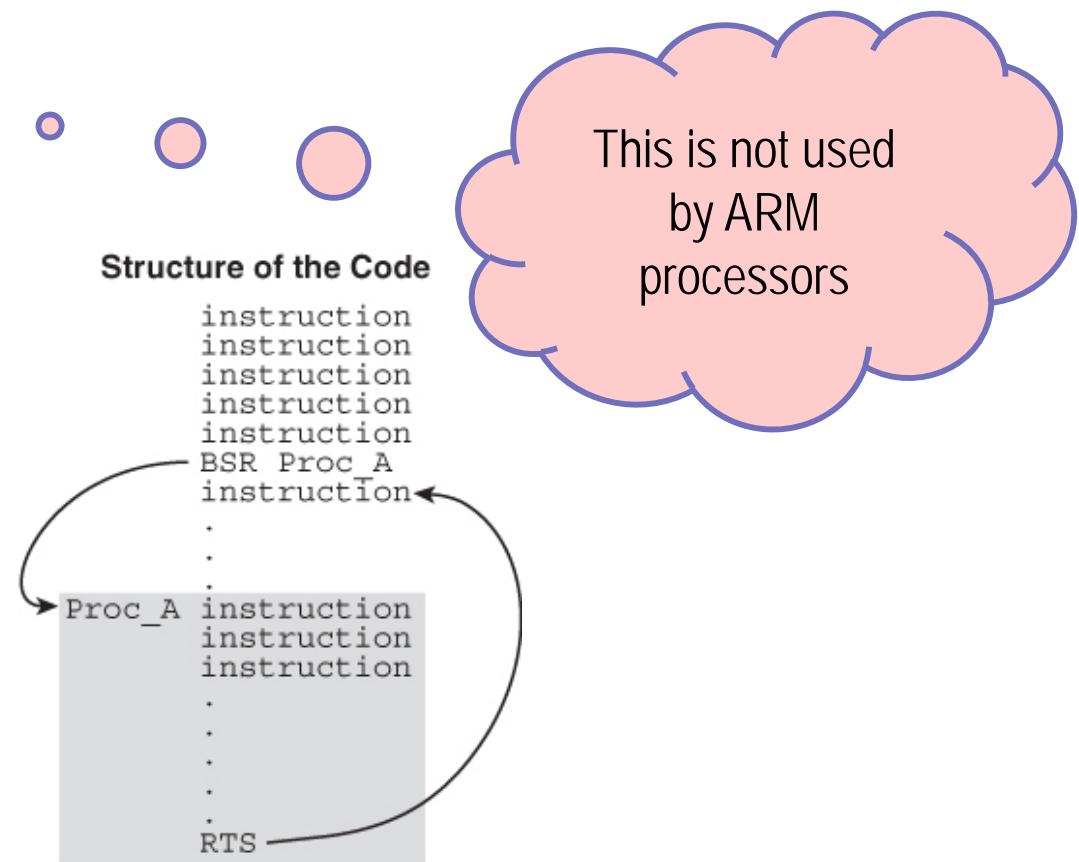
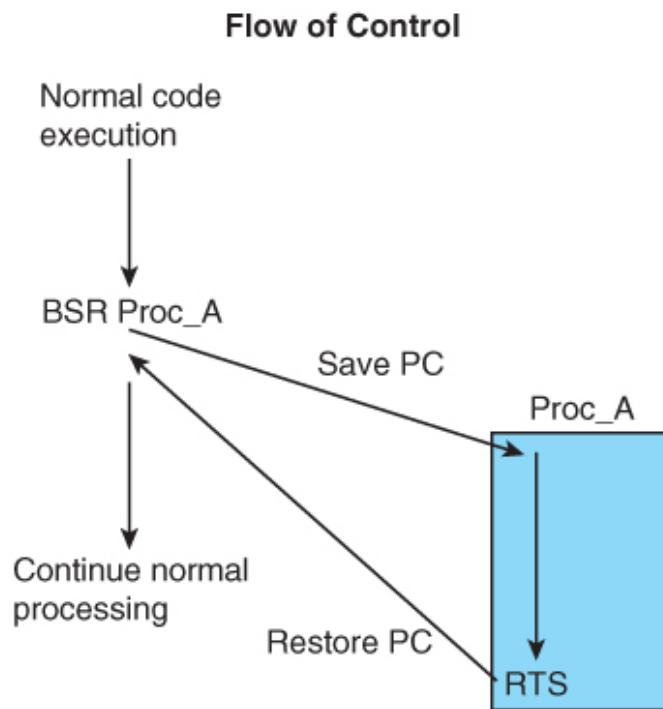
*Office: MC-419*

*Email: [elsakka@csd.uwo.ca](mailto:elsakka@csd.uwo.ca)*

*Phone: 519-661-2111 x86996*

# Subroutine call and Return

FIGURE 3.40 The subroutine call and return



This is not used  
by ARM  
processors

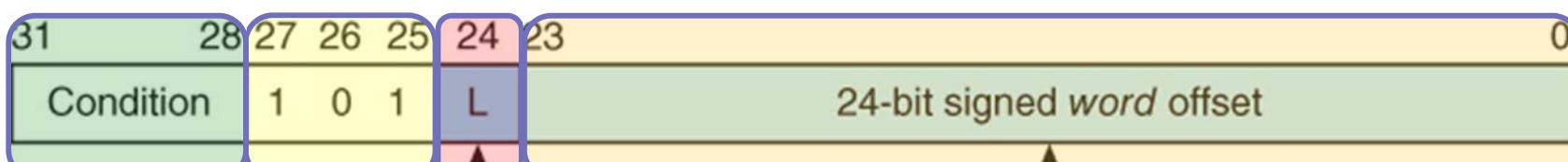
# Subroutine call and Return

- ARM's *branch with link* instruction, **BL**,
  - automatically saves the return address in register **r14**.
- The branch instruction (Figure 3.41) has an 8-bit op-code with a 24-bit *signed* program counter relative offset (*word address offset*).
- The 24-bit offset is
  - shifted left twice to convert the word-offset address to a byte address,  
*sign-extended* to 32 bits,
  - added it to the program counter (*the result is PC ± 32 MBytes*).

Do not forget the pipelining effect

FIGURE 3.41

Encoding ARM's branch and branch-with-link instructions



The L-bit is 0 for a branch instruction and 1 for a branch with link instruction.

The 24-bit word offset is shifted left twice to create a 26-bit byte offset.



# Example 1: Subroutine call and Return

Calling a leaf subroutine using **BL** instruction

```

Main      MOV r1,#0x10          ; just a dummy operation
          BL  Function1        ; jump to Function1
          MOV r3,#0x30          ; just a dummy operation
          BL  Function2        ; jump to Function2
          MOV r5,#0x50          ; just a dummy operation
loop      B    loop
;
;-----  

;----- Leaf subroutine -----  

;-----  

Function1 MOV r2,#0x20          ; just a dummy operation
          MOV r15,r14         ; return from Function1
;
;-----  

;----- Leaf subroutine -----  

;-----  

Function2 MOV r4,#0x40          ; just a dummy operation
          MOV r15,r14         ; return from Function2
;
```

# Example 1: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

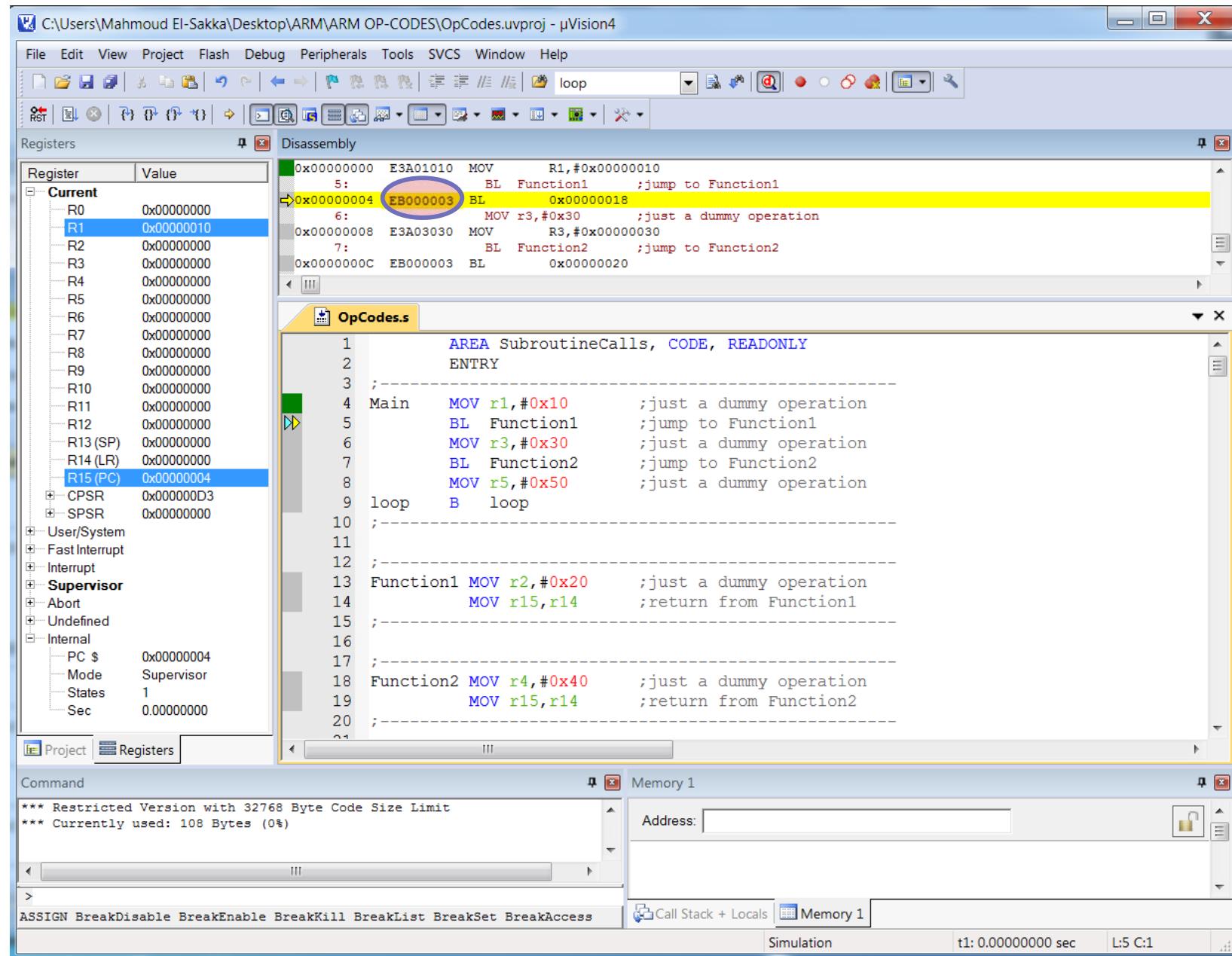
- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- File Menu:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Open, Save, Build, Run, Stop, and various simulation and memory tools.
- Registers Window:** Shows the current values of all ARM registers (R0-R15, CPSR, SPSR) in hex format. The CPSR register is highlighted with a blue selection bar.
- Disassembly Window:** Displays the assembly code with addresses and opcodes. The code includes calls to Function1 and Function2, and a loop that branches back to the start of the loop.
- OpCodes.s Window:** Shows the source code for the assembly file. It defines sections for subroutine calls, entry points, and two functions (Function1 and Function2) that perform dummy operations and return.
- Command Window:** Displays build messages, currently showing a restricted version warning and memory usage information.
- Memory Window:** A memory viewer with an address field and a data dump area.
- Bottom Status Bar:** Shows the current simulation state (t1: 0.0000000 sec), clock frequency (L4 C1), and other system information.

```

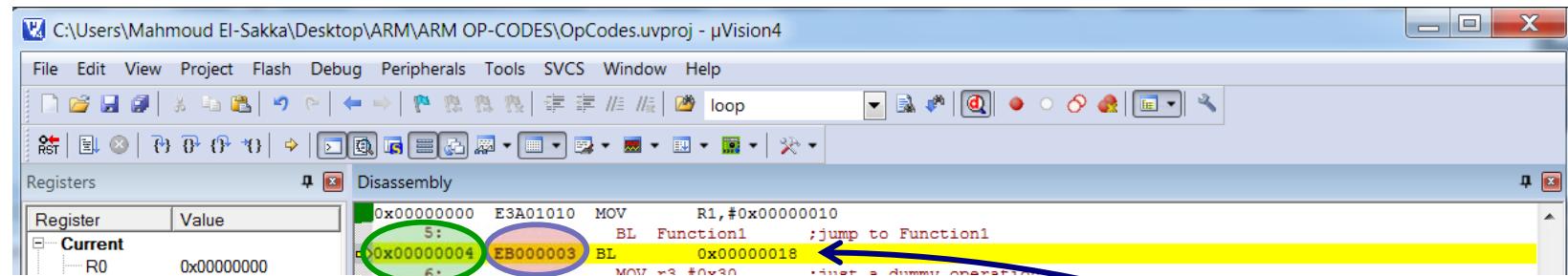
1 AREA SubroutineCalls, CODE, READONLY
2 ENTRY
3 ;
4 Main   MOV r1,#0x10      ;just a dummy operation
5     BL Function1        ;jump to Function1
6     MOV r3,#0x30        ;just a dummy operation
7     BL Function2        ;jump to Function2
8     MOV r5,#0x50        ;just a dummy operation
9 loop    B  loop
10 ;
11 ;
12 ;
13 Function1 MOV r2,#0x20  ;just a dummy operation
14         MOV r15,r14    ;return from Function1
15 ;
16 ;
17 Function2 MOV r4,#0x40  ;just a dummy operation
18         MOV r15,r14    ;return from Function2
19 ;
20 ;
21

```

# Example 1: Subroutine call and Return



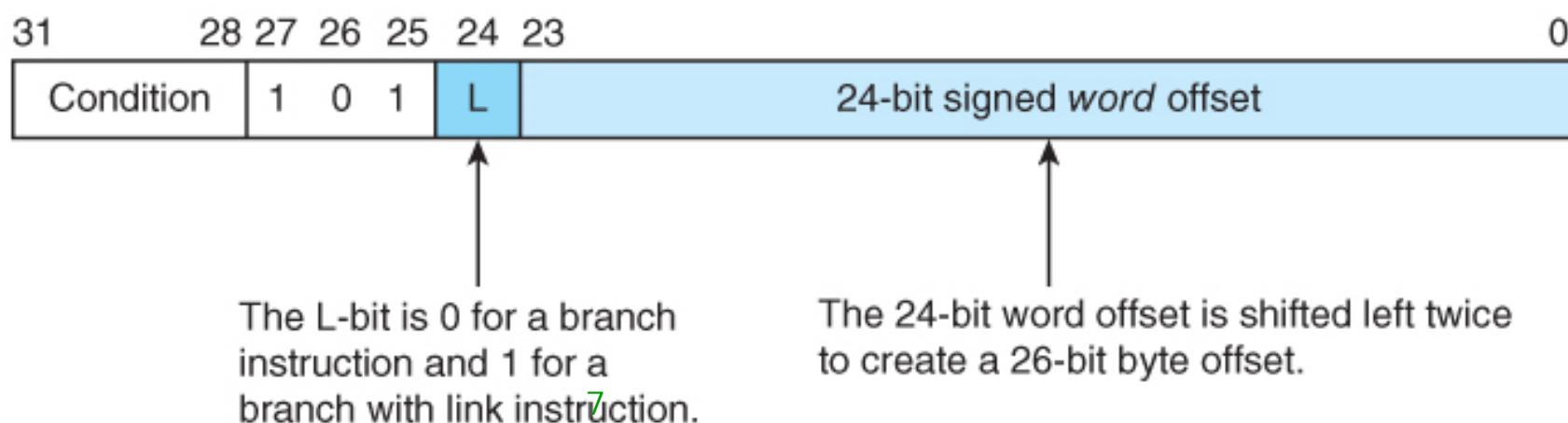
# Example 1: Subroutine call and Return



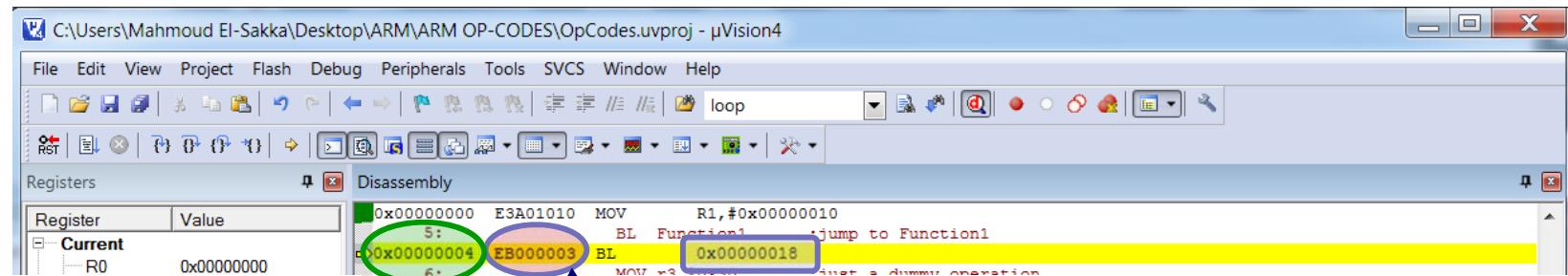
- 24-bit signed *word* offset = 0x000003
- 24-bit signed *byte* offset =  $0x000003 \times 0x4 = 0x00000C$
- Effective address = 24-bit signed *byte* offset + *PC value* + *pipeline effect*  
 $= 0x00000C + 0x00000004 + 0x8 = 0x00000018$

FIGURE 3.41

Encoding ARM's branch and branch-with-link instructions

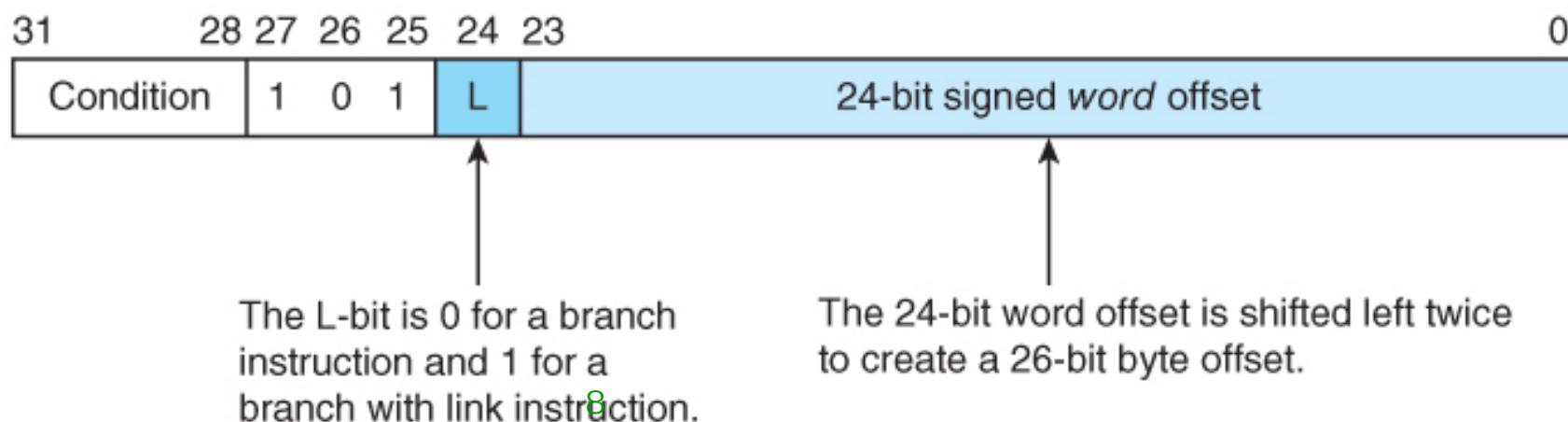


# Example 1: Subroutine call and Return



- *PC value = 0x00000004*
- 24-bit signed *byte* offset = Effective address - *PC value - pipeline effect* =  $0x18 - 0x4 - 0x8 = 0xC$
- 24-bit signed *word* offset = 24-bit signed *byte* offset  $\div$  4  
=  $0x00000C \div 4 = 0x000003$

**FIGURE 3.41** Encoding ARM's branch and branch-with-link instructions



# Example 1: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Open, Save, Build, Run, Stop, and various simulation and memory tools.
- Registers Window:** Shows the current values of all ARM registers (R0-R13, CPSR, SPSR) and internal states (PC \$, Mode, States, Sec).
- Disassembly Window:** Displays the assembly code for the current program. The instruction at address 0x00000018 is highlighted in yellow: `MOV r15,r14 ;return from Function1`.
- Source Editor:** Shows the source code file OpCodes.s containing the assembly code for Main, Function1, and Function2.
- Command Window:** Displays build messages: "Restricted Version with 32768 Byte Code Size Limit" and "Currently used: 108 Bytes (0%)".
- Memory Window:** A memory viewer with tabs for Call Stack + Locals and Memory 1.
- Status Bar:** Shows "Step one line", "Simulation", "t1: 0.0000000 sec", and "L13 C:1".

```

OpCodes.s
1 AREA SubroutineCalls, CODE, READONLY
2 ENTRY
3 ;-----
4 Main   MOV r1,#0x10      ;just a dummy operation
5     BL Function1        ;jump to Function1
6     MOV r3,#0x30      ;just a dummy operation
7     BL Function2        ;jump to Function2
8     MOV r5,#0x50      ;just a dummy operation
9 loop    B  loop
10 ;
11 ;
12 ;-----
13 Function1 MOV r2,#0x20      ;just a dummy operation
14         MOV r15,r14      ;return from Function1
15 ;
16 ;
17 ;-----
18 Function2 MOV r4,#0x40      ;just a dummy operation
19         MOV r15,r14      ;return from Function2
20 ;
21

```

# Example 1: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following windows:

- Registers**: Shows the current values of various registers. The R2 register is highlighted in blue, showing its value as 0x00000020.
- Disassembly**: Displays the assembly code for the subroutine calls and returns. The code includes instructions like MOV r2, #0x20, BL Function1, MOV r15, r14, and B loop.
- OpCodes.s**: Shows the source assembly file containing the subroutine definitions and main program logic.
- Command**: Displays build-related information, including a note about a restricted version with a 32768 byte code size limit.
- Memory 1**: A memory dump window showing the memory contents at address 0x00000000.

```

1 AREA SubroutineCalls, CODE, READONLY
2
3 ENTRY
4 Main   MOV r1,#0x10      ;just a dummy operation
5     BL Function1        ;jump to Function1
6     MOV r3,#0x30      ;just a dummy operation
7     BL Function2        ;jump to Function2
8     MOV r5,#0x50      ;just a dummy operation
9 loop    B    loop
;
12 ;
13 Function1 MOV r2,#0x20      ;just a dummy operation
14         MOV r15,r14      ;return from Function1
15 ;
17 ;
18 Function2 MOV r4,#0x40      ;just a dummy operation
19         MOV r15,r14      ;return from Function2
20 ;

```

# Example 1: Subroutine call and Return

Screenshot of the µVision4 IDE showing the assembly code for a subroutine call and return example.

The project is named "OpCodes.uvproj".

**Registers:**

Register	Value
R0	0x00000000
R1	0x00000010
R2	0x00000020
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)	0x00000008
<b>R15 (PC)</b>	<b>0x00000008</b>
CPSR	0x000000D3
SPSR	0x00000000

**Disassembly:**

```

0x00000004 EB000003 BL 0x00000018
6:          MOV r3,#0x30 ;just a dummy operation
7: >0x00000008 E3A03030 MOV R3,#0x00000030
8:          BL Function2 ;jump to Function2
9:          MOV r5,#0x50 ;just a dummy operation
0x00000010 E3A05050 MOV R5,#0x00000050

```

**Source Code (OpCodes.s):**

```

1      AREA SubroutineCalls, CODE, READONLY
2      ENTRY
3      ;
4      Main   MOV r1,#0x10 ;just a dummy operation
5          BL Function1 ;jump to Function1
6          MOV r3,#0x30 ;just a dummy operation
7          BL Function2 ;jump to Function2
8          MOV r5,#0x50 ;just a dummy operation
9      loop   B loop
10     ;
11     ;
12     ;
13     Function1 MOV r2,#0x20 ;just a dummy operation
14             MOV r15,r14 ;return from Function1
15     ;
16     ;
17     ;
18     Function2 MOV r4,#0x40 ;just a dummy operation
19             MOV r15,r14 ;return from Function2
20     ;
21

```

**Memory:**

Address: [ ]

Call Stack + Locals | Memory 1

Step one line | Simulation | t1: 0.00000000 sec | L:6 C:1

# Example 1: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- File Menu:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Run, Stop, Break, Step, and various simulation and memory tools.
- Registers Window:** Shows the current values of all ARM registers. The R3 register is highlighted with a blue selection bar and has its value (0x00000030) displayed in the Value column.
- Disassembly Window:** Displays the assembly code. A specific instruction at address 0x0000000C is highlighted with a yellow background and circled in orange. This instruction is a BL (Branch and Link) instruction with the address 0x00000020, which corresponds to the start of the Function2 code.
- OpCodes.s File:** Shows the source code for the subroutine calls. It includes sections for Main, Function1, and Function2, each containing dummy operations and subroutine calls.
- Memory Window:** Allows viewing and modifying memory contents. It includes fields for Address and Value, and tabs for Call Stack + Locals and Memory 1.
- Command Window:** Displays build-related messages, including a note about a restricted version size limit.
- Status Bar:** Shows the current simulation time (t1: 0.0000000 sec) and clock cycle count (L:7 C:1).

# Example 1: Subroutine call and Return

```

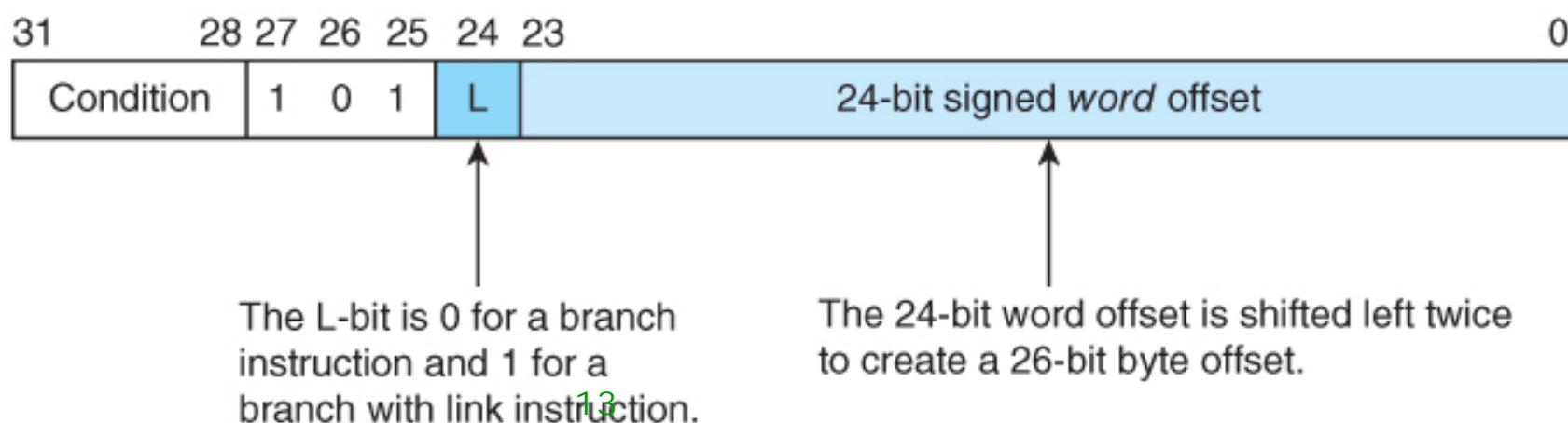
C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers Disassembly
Register Value
Current
R0 0x00000000
R1 0x00000010
R2 0x00000020
R3 0x00000030
0x00000004 EB000003 BL 0x00000018
6:           MOV r3,#0x30 ;just a dummy operation
0x00000008 E3A03030 MOV R3,#0x00000030
7:           BL Function2 ;jump to Function2
0x0000000C EB000003 BL 0x00000020
8:           MOV r5,#0x50 ;just a dummy operation
0x00000010 E3A05050 MOV R5,#0x00000050

```

- 24-bit signed *word* offset = 0x000003
- 24-bit signed *byte* offset =  $0x000003 \times 0x4 = 0x00000C$
- Effective address = 24-bit signed *byte* offset + *PC value* + *pipeline effect*  
 $= 0x00000C + 0x0000000C + 0x8 = 0x00000020$

FIGURE 3.41

Encoding ARM's branch and branch-with-link instructions



# Example 1: Subroutine call and Return

```

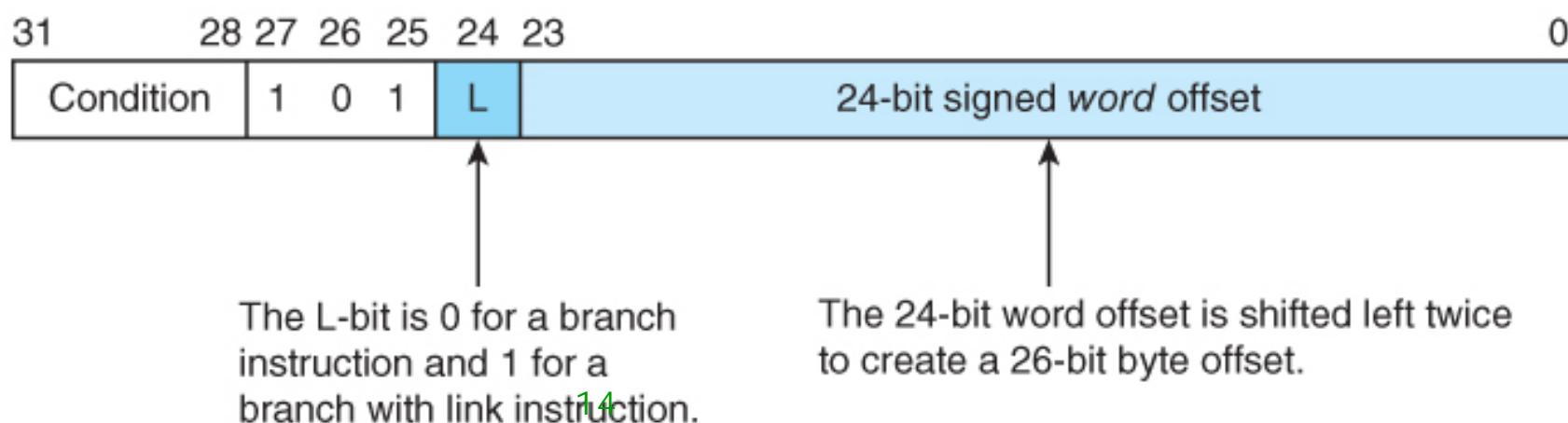
C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers Disassembly
Register Value
Current
R0 0x00000000
R1 0x00000010
R2 0x00000020
R3 0x00000030
0x00000004 EB000003 BL 0x00000018
6: MOV r3,#0x30 ;just a dummy operation
0x00000008 E3A03030 MOV R3,#0x00000030
7: BL Function2 ;jump to Function2
0x0000000C EB000003 BL 0x00000020
8: MOV r5,#0x50 ;just a dummy operation
0x00000010 E3A05050 MOV R5,#0x00000050

```

- *PC value = 0x0000000C* Effective address = **0x00000020**
- 24-bit signed *byte* offset =  
Effective address – *PC value* – *pipeline effect* = **0x20** – **0xC** – **0x8** = **0xC**
- 24-bit signed *word* offset = 24-bit signed *byte* offset ÷ 4  
= **0x00000C** ÷ 4 = **0x000003**

FIGURE 3.41

Encoding ARM's branch and branch-with-link instructions



# Example 1: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbars:** Standard toolbar with icons for file operations, project management, and simulation.
- Registers Window:** Shows the current values of all ARM registers. R14 (LR) is highlighted with a yellow background and contains the value 0x00000010. R15 (PC) is also highlighted and contains the value 0x00000020.
- Disassembly Window:** Displays the assembly code for the program. The current instruction at address 0x00000020 is highlighted in yellow and is a BL (branch and link) instruction to address 0x00000040, which corresponds to the start of Function2. Other instructions include MOV r4, #0x40; MOV r15, r14; and MOV r15, r14.
- Source Editor:** Shows the source code file OpCodes.s. It defines two functions, Main and Function2, each containing dummy operations (MOV r1, #0x10; MOV r3, #0x30; MOV r5, #0x50). The Main function also contains a loop that calls Function2.
- Memory Window:** A small window showing memory starting at address 0x00000000.
- Command Window:** Displays build-related messages, including a note about a restricted version with a 32768 byte code size limit and the current usage of 108 bytes (0%).
- Status Bar:** Shows the current simulation state: t1: 0.0000000 sec L:18 C:1.

# Example 1: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Open, Save, Build, Run, Stop, and various simulation and memory dump tools.
- Registers Window:** Shows the current values of all ARM registers. R4 is highlighted with a blue selection bar, and R14 (LR) is highlighted with a red selection bar.
- Disassembly Window:** Displays the assembly code for the program. The current instruction at address 0x00000024 is highlighted with a yellow selection bar. The code includes:
 

```

        0x0000001C E1A0F00E MOV      PC,R14
        18: Function2 MOV r4,#0x40    ;just a dummy operation
        0x00000020 E3A04040 MOV      R4,#0x00000040
        19:          MOV r15,r14    ;return from Function2
        =>0x00000024 E1A0F00E MOV      PC,R14
        0x00000028 00000000 ANDEQ   R0,R0,R0
        0x0000002C 00000000 ANDEQ   R0,R0,R0
      
```
- Source Editor:** Shows the C source code for OpCodes.s:
 

```

13 Function1 MOV r2,#0x20    ;just a dummy operation
14           MOV r15,r14    ;return from Function1
15 ;
16 ;
17 ;
18 Function2 MOV r4,#0x40    ;just a dummy operation
19           MOV r15,r14    ;return from Function2
20 ;
21
22
23
24
25
26
27
28
29
30
31
32
      
```
- Memory Window:** Shows a memory dump with an empty address field and a lock icon.
- Command Window:** Displays build messages:
 

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 108 Bytes (0%)
      
```
- Bottom Status Bar:** Shows Simulation, t1: 0.00000000 sec, L:19 C:1.

# Example 1: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following windows:

- Registers**: Shows the current register values. R15 (PC) is highlighted at 0x00000010.
- Disassembly**: Shows the assembly code with addresses and opcodes. A specific instruction at address 0x00000010 is highlighted in yellow: `MOV R5,#0x50`.
- OpCodes.s**: The source code file containing the assembly code. It defines three functions: Function1, Function2, and loop.
- Command**: Displays build messages: "Restricted Version with 32768 Byte Code Size Limit" and "Currently used: 108 Bytes (0%)".
- Memory 1**: A memory dump window with an empty address field.
- Call Stack + Locals**: A stack dump window.
- Simulation**: A status bar at the bottom indicating "t1: 0.00000000 sec L:8 C:1".

```

OpCodes.s
7     BL Function2      ;jump to Function2
8     MOV r5,#0x50       ;just a dummy operation
9     loop    B  loop
10    ;
11    ;
12    ;
13    Function1 MOV r2,#0x20   ;just a dummy operation
14          MOV r15,r14    ;return from Function1
15    ;
16    ;
17    ;
18    Function2 MOV r4,#0x40   ;just a dummy operation
19          MOV r15,r14    ;return from Function2
20    ;
21    ;
22    ;
23    ;
24    ;
25    ;
26    ;
27

```

# Example 1: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following windows:

- Registers**: Shows the current register values, with R5 highlighted.
- Disassembly**: Shows the assembly code for the subroutine loop. The instruction at address 0x00000010, `B loop`, is highlighted in yellow.
- OpCodes.s**: Shows the source code for the assembly file. It defines two functions, Main and Function1, each containing a dummy operation and a jump back to the start of the loop. Function1 also contains a return instruction.
- Command**: Displays compiler messages about restricted version and memory usage.
- Memory 1**: A memory dump window showing the first 16 bytes of memory.

```

C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
RST Disassembly Registers OpCodes.s Command Memory 1

Registers
Register Value
Current
R0 0x00000000
R1 0x00000010
R2 0x00000020
R3 0x00000030
R4 0x00000040
R5 0x00000050
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000010
R15 (PC) 0x00000014
CPSR 0x000000D3
SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
PC $ 0x00000014
Mode Supervisor
States 17
Sec 0.00000000
Project Registers

OpCodes.s
3 ;
4 Main    MOV r1,#0x10      ;just a dummy operation
5          BL Function1     ;jump to Function1
6          MOV r3,#0x30      ;just a dummy operation
7          BL Function2     ;jump to Function2
8          MOV r5,#0x50      ;just a dummy operation
9 loop    B  loop           ;just a dummy operation
10 ;
11 ;
12 ;
13 Function1 MOV r2,#0x20      ;just a dummy operation
14          MOV r15,r14      ;return from Function1
15 ;
16 ;
17 ;
18 Function2 MOV r4,#0x40      ;just a dummy operation
19          MOV r15,r14      ;return from Function2
20 ;
21 ;
22 ;

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 108 Bytes (0%)
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
Call Stack + Locals Memory 1
Simulation t1: 0.00000000 sec L:9 C:1

```



## Example 2: Subroutine call and Return

Calling a none leaf subroutine using **BL** instruction

```

Main    MOV r1,#0x10           ; just a dummy operation
        BL  Function1          ; jump to Function1
        MOV r3,#0x30           ; just a dummy operation
        BL  Function2          ; jump to Function2
        MOV r5,#0x50           ; just a dummy operation
loop    B   loop
;
;----- None leaf subroutine -----
;
Function1 MOV r2,#0x20           ; just a dummy operation
        BL  Function2          ; jump to Function2
        MOV r15,r14            ; return from Function1
;
;----- Leaf subroutine -----
;
Function2 MOV r4,#0x40           ; just a dummy operation
        MOV r15,r14            ; return from Function2
;

```

# Example 2: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- File Menu:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Run, Stop, Break, Step, and various simulation and memory tools.
- Registers Window:** Shows the current values of all ARM registers (R0-R15, CPSR, SPSR) in hex format. The CPSR register is highlighted with a yellow background.
- Disassembly Window:** Displays the assembly code for the current program. The highlighted section shows a loop that calls Function1, performs dummy operations on r3 and r5, and then loops back.
- OpCodes.s Window:** Shows the source code for the program. It defines areas for subroutine calls, entry points, and two functions: Main and Function1. Main calls Function1, which in turn calls Function2. Function2 then returns to Function1, and finally Function1 returns to Main.
- Command Window:** Displays compiler messages about restricted version and byte code size limit.
- Memory Window:** Allows viewing and modifying memory at a specific address.
- Bottom Status Bar:** Shows simulation status (t1: 0.0000000 sec), clock frequency (L4 C:1), and other system information.

```

AREA SubroutineCalls, CODE, READONLY
ENTRY
;
Main    MOV r1,#0x10      ;just a dummy operation
        BL Function1      ;jump to Function1
        MOV r3,#0x30      ;just a dummy operation
        MOV r5,#0x50      ;just a dummy operation
loop    B  loop          ;
;
Function1 MOV r2,#0x20      ;just a dummy operation
        BL Function2      ;jump to Function2
        MOV r15,r14       ;return from Function1
;
Function2 MOV r4,#0x40      ;just a dummy operation
        MOV r15,r14       ;return from Function2
;
```

# Example 2: Subroutine call and Return

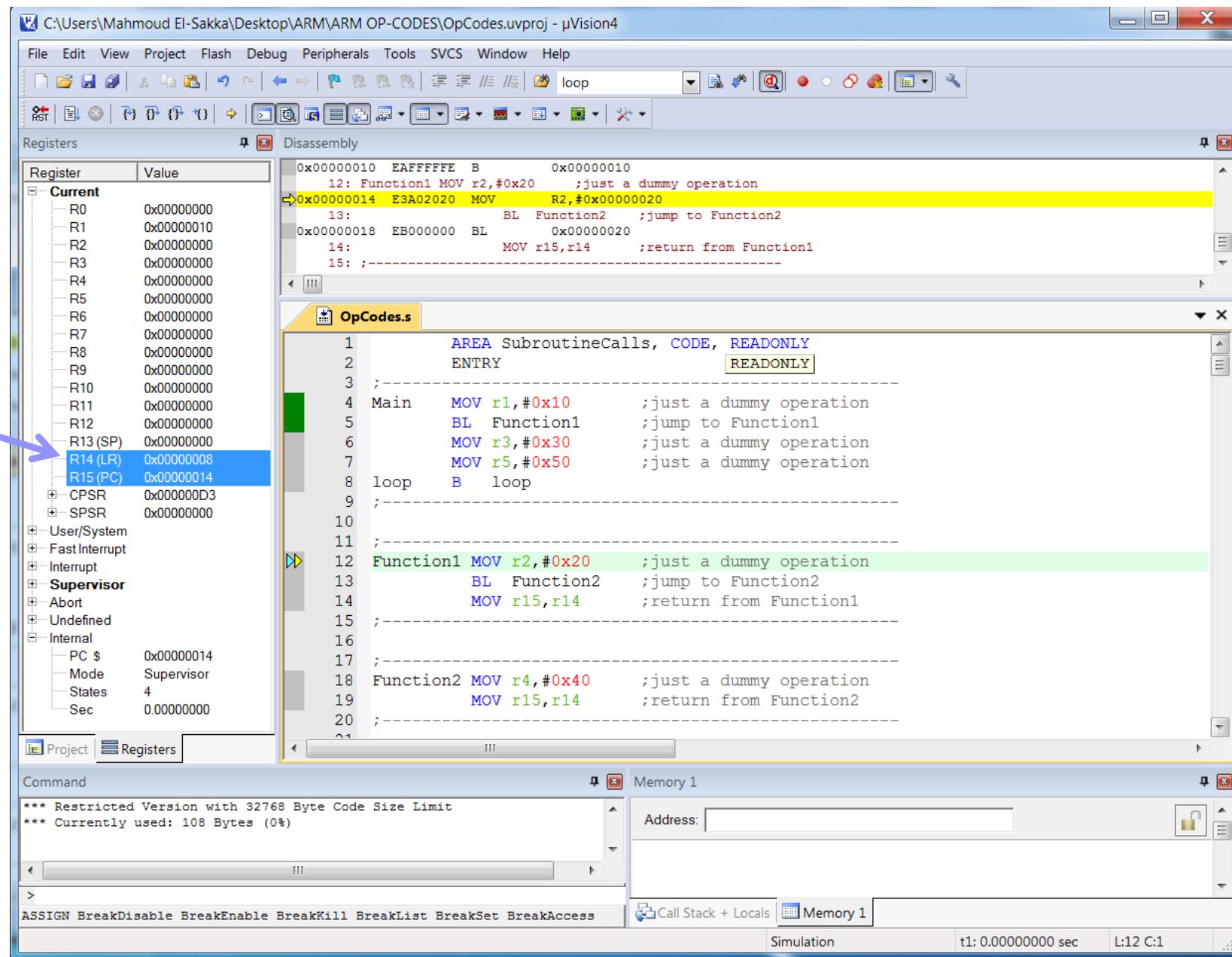
The screenshot shows the µVision4 IDE interface with the following windows:

- Registers**: Shows the current register values, with R15 (PC) highlighted at 0x00000004.
- Disassembly**: Displays the assembly code for the program. The instruction at address 0x00000004 is highlighted in yellow: `EB000002 BL 0x00000014`. This is a branch-and-link (BL) instruction that calls the `Function1` subroutine.
- OpCodes.s**: Shows the source code for the assembly file. It defines two functions: `Main` and `Function1`. The `Main` function contains a loop that calls `Function1` and performs dummy operations. The `Function1` function also contains dummy operations and returns control to `Main`.
- Command**: Shows compiler messages indicating a restricted version with a 32768 byte code size limit and current usage of 108 bytes.
- Memory 1**: A memory dump window showing the memory starting at address 0x00000000.

```

AREA SubroutineCalls, CODE, READONLY
ENTRY
;
Main    MOV r1,#0x10      ;just a dummy operation
        BL Function1      ;jump to Function1
        MOV r3,#0x30      ;just a dummy operation
        MOV r5,#0x50      ;just a dummy operation
loop    B  loop          ;just a dummy operation
;
Function1 MOV r2,#0x20      ;just a dummy operation
        BL Function2      ;jump to Function2
        MOV r15,r14       ;return from Function1
;
Function2 MOV r4,#0x40      ;just a dummy operation
        MOV r15,r14       ;return from Function2
;
```

# Example 2: Subroutine call and Return



The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- File Menu:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Open, Save, Build, Run, Stop, and various simulation and memory tools.
- Registers Window:** Shows the current values of all ARM registers. A blue arrow points to the R14 (LR) register, which has a value of 0x00000008. Other registers show values like R0=0x00000000, R1=0x00000010, etc.
- Disassembly Window:** Displays the assembly code. The current instruction at address 0x00000014 is highlighted in yellow: E3A02020 MOV R2, #0x00000020. The assembly code includes:
 

```

        0x00000010 EAFFFFFE B    0x00000010
        12: Function1 MOV r2,#0x20 ;just a dummy operation
        13:          E3A02020 MOV R2,#0x00000020
        14:          BL Function2 ;jump to Function2
        0x00000018 EB000000 BL   0x00000020
        15:          MOV r15,r14 ;return from Function1
        16:          ;
      
```
- Source Editor:** Shows the source code file OpCodes.s. It defines two functions, Main and Function1, and a loop. The assembly code in the disassembly window corresponds to the source code in the editor.
- Memory Window:** Shows a memory dump with an address field and a data field.
- Command Window:** Displays build messages:
 

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 108 Bytes (0%)
      
```
- Bottom Status Bar:** Simulation, t1: 0.00000000 sec, L:12 C:1

# Example 2: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbars:** Standard toolbar with icons for file operations, project management, and simulation.
- Registers Window:** Shows the current register values. R2 is highlighted in red, indicating it is the current register being viewed.
- Disassembly Window:** Displays the assembly code. The instruction at address 0x00000018, which is the return address from Function1, is highlighted in yellow.
- Source Editor:** The code file "OpCodes.s" is open. The assembly code is as follows:
 

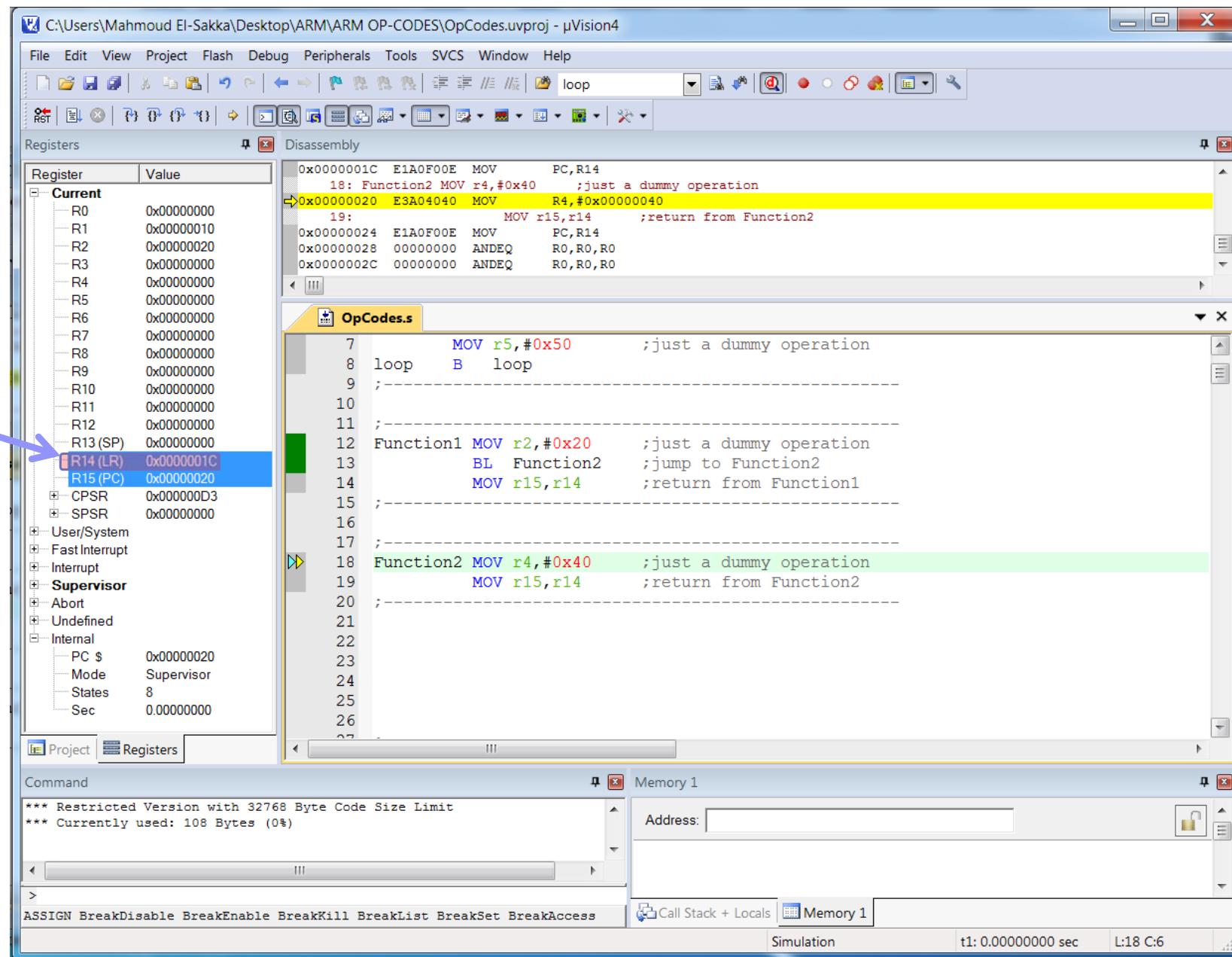
```

      7      MOV r5,#0x50      ;just a dummy operation
      8  loop    B  loop
      9 ;
      10
      11 ;
      12 Function1 MOV r2,#0x20      ;just a dummy operation
      13           BL Function2 ;jump to Function2
      14           MOV r15,r14 ;return from Function1
      15 ;
      16
      17 ;
      18 Function2 MOV r4,#0x40      ;just a dummy operation
      19           MOV r15,r14 ;return from Function2
      20 ;
      21
      22
      23
      24
      25
      26
      27
      
```
- Command Window:** Shows build messages:
 

```

      *** Restricted Version with 32768 Byte Code Size Limit
      *** Currently used: 108 Bytes (0%)
      
```
- Memory Window:** A memory dump window with an empty address field.
- Bottom Status Bar:** Simulation status: t1: 0.00000000 sec L:13 C:46

# Example 2: Subroutine call and Return



The screenshot shows the µVision4 IDE interface with the following details:

- File Path:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Registers Window:**
  - Current Register Values:
    - R0: 0x00000000
    - R1: 0x00000010
    - R2: 0x00000020
    - 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): 0x0000001C (highlighted with a blue arrow)
    - R15 (PC): 0x00000020 (highlighted with a blue arrow)
  - User/System, Fast Interrupt, Interrupt, Supervisor, Abort, Undefined, Internal, PC \$, Mode, States, Sec, CPSR, SPSR are listed but have no values.
- Disassembly Window:**

```

0x00000000 E1A0F00E MOV    PC,R14
18: Function2 MOV r4,#0x40 ;just a dummy operation
→0x00000020 E3A04040 MOV    R4,#0x00000040
19:           MOV r15,r14 ;return from Function2
0x00000024 E1A0F00E MOV    PC,R14
0x00000028 00000000 ANDEQ R0,R0,R0
0x0000002C 00000000 ANDEQ R0,R0,R0
    
```
- Source Code Window:**

```

OpCodes.s
7      MOV r5,#0x50      ;just a dummy operation
8 loop   B  loop
9 ;
10
11 ;
12 Function1 MOV r2,#0x20 ;just a dummy operation
13           BL Function2 ;jump to Function2
14           MOV r15,r14 ;return from Function1
15 ;
16
17 ;
18 Function2 MOV r4,#0x40 ;just a dummy operation
19           MOV r15,r14 ;return from Function2
20 ;
21
22
23
24
25
26
    
```
- Memory Window:** Shows a memory dump with Address: field and Call Stack + Locals tab selected.
- Command Window:**

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 108 Bytes (0%)
    
```
- Bottom Status Bar:** Simulation t1: 0.00000000 sec L:18 C:6

# Example 2: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Open, Save, Build, Run, Stop, and various simulation and memory tools.
- Registers Window:** Shows the current values of all ARM registers. R4 is highlighted with a blue border and has a value of 0x00000040. R15 (PC) is also highlighted with a blue border and has a value of 0x00000024.
- Disassembly Window:** Displays the assembly code. The instruction at address 0x00000024 is highlighted with a yellow background: E1A0F00E MOV PC,R14. This is the return instruction from Function2. Other instructions like BL Function2 and MOV r4,#0x40 are also visible.
- Source Editor:** Shows the source code file OpCodes.s. It contains two function definitions: Function1 and Function2. Function1 contains a jump to Function2 and a return. Function2 contains a dummy operation and a return.
- Command Window:** Displays build messages: "\*\*\* Restricted Version with 32768 Byte Code Size Limit" and "\*\*\* Currently used: 108 Bytes (0%)".
- Memory Window:** A memory dump window titled "Memory 1" with an empty address field.
- Bottom Navigation:** Includes tabs for Call Stack + Locals, Memory 1, Simulation, and status indicators t1: 0.0000000 sec and L:19 C:50.

# Example 2: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Open, Save, Build, Run, Stop, and various simulation and memory tools.
- Registers Window:** Shows the current state of ARM registers. R15 (PC) is highlighted with the value 0x0000001C.
- Disassembly Window:** Displays the assembly code for the program. The current instruction is highlighted in yellow: E1A0F00E MOV PC,R14. The assembly code includes:
 

```

      18: Function2 MOV r4,#0x40 ;just a dummy operation
      19:           MOV r15,r14 ;return from Function2
      19:           MOV r15,r14 ;return from Function2
      20:           MOV r15,r14 ;return from Function2
      21:
      22:
      23:
      24:
      25:
      26:
      27:
      28:
      29:
      30:
      31:
      32:
      
```
- OpCodes Window:** Shows the assembly code for the subroutine Function2. The current instruction is highlighted in green: MOV r15,r14 ;return from Function2.
 

```

      13:     BL Function2 ;jump to Function2
      14:     MOV r15,r14 ;return from Function1
      15: ;
      16:
      17: ;
      18: Function2 MOV r4,#0x40 ;just a dummy operation
      19:           MOV r15,r14 ;return from Function2
      20: ;
      21:
      22:
      23:
      24:
      25:
      26:
      27:
      28:
      29:
      30:
      31:
      32:
      
```
- Command Window:** Displays build messages:
 

```

      *** Restricted Version with 32768 Byte Code Size Limit
      *** Currently used: 108 Bytes (0%)
      
```
- Memory Window:** A memory dump tool with an address field and a data viewer.
- Bottom Status Bar:** Simulation, t1: 0.00000000 sec, L:14 C:50

# Example 2: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Run, Stop, Break, and various simulation and memory manipulation tools.
- Registers Window:** Shows the current register values for R0-R15 and CPSR/SPSR.
- Disassembly Window:** Displays the assembly code with comments. The highlighted line is "MOV r15,r14 ;return from Function2".
- Source Editor:** Shows the source code file OpCodes.s with the same assembly code. Lines 14 and 15 are highlighted.
- Command Window:** Displays build messages: "\*\*\* Restricted Version with 32768 Byte Code Size Limit" and "\*\*\* Currently used: 108 Bytes (0%)".
- Memory Window:** A memory viewer with an empty address field and a lock icon.
- Bottom Status Bar:** Simulation status: t1: 0.0000000 sec L14 C1.

```

0x00000001C E1A0F00E MOV      PC,R14
18: Function2 MOV r4,#0x40    ;just a dummy operation
0x00000020 E3A04040 MOV      R4,#0x00000040
19:          MOV r15,r14    ;return from Function2
0x00000024 E1A0F00E MOV      PC,R14
0x00000028 00000000 ANDEQ   R0,R0,R0
0x0000002C 00000000 ANDEQ   R0,R0,R0

OpCodes.s
13     BL  Function2    ;jump to Function2
14     MOV r15,r14    ;return from Function1
15 ;
16 ;
17 ;
18 Function2 MOV r4,#0x40    ;just a dummy operation
19     MOV r15,r14    ;return from Function2
20 ;
21 ;
22 ;
23 ;
24 ;
25 ;
26 ;
27 ;
28 ;
29 ;
30 ;
31 ;
32 ;

```



## Example 3: Subroutine call and Return

Calling a none leaf subroutine using **B** instruction and a stack without using **r14**



## Example 3: Subroutine call and Return

Calling a none leaf subroutine using **B** instruction and a stack without using **r14**

```
Main    MOV r1,#0x10          ; just a dummy operation
        ADR r13,stack         ; set up the stack pointer
        STR r15,[r13,#-4]!   ; pre-decrement the stack pointer
        B Function1           ; jump to Function1 (B not BL)
        MOV r3,#0x30          ; just a dummy operation
loop    B loop
;-----
;-----
Function1 MOV r2,#0x20          ; just a dummy operation
        STR r15,[r13,#-4]!   ; pre-decrement the stack pointer
        B Function2           ; jump to Function2 (B not BL)
        MOV r5,#0x50          ; just a dummy operation
        LDR r12,[r13],#4      ; get the PC and post-increment the stack pointer
        SUB r15,r12,#4        ; modify the restored PC (because it is 4 too big)
                           ; and return from Function1
;-----
;-----
Function2 MOV r4,#0x40          ; just a dummy operation
        LDR r12,[r13],#4      ; get the PC and post-increment the stack pointer
        SUB r15,r12,#4        ; modify the restored PC (because it is 4 too big)
                           ; and return from Function2
;-----
;-----
stack   SPACE 0x40            ; reserved room for the stack to grow up
        DCD 0x0               ; the base of the stack
;
```

# Example 3: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Registers:** Shows the current values of all ARM registers (R0-R15, CPSR, SPSR) and various system modes (User, System, Fast Interrupt, Supervisor, Abort, Undefined, Internal). The CPSR register is highlighted.
- Disassembly:** Displays the assembly code for the subroutine calls. The first few instructions are:
 

```

      0x00000000 E3A01010 MOV R1,#0x00000010
      5:          ADR r13,stack ;set up the stack pointer
      6:
      0x00000004 E28FD070 ADD R13,PC,#0x00000070
      7:          STR r15,[r13,#-4]! ;pre-decrement the stack pointer
      0x00000008 E52DF004 STR PC,[R13,#-0x0004]!
      8:          B Function1 ;jump to Function1
    
```
- OpCodes.s:** Shows the source code for the subroutine definitions. It includes the AREA definition, ENTRY point, and the assembly code for Main, Function1, and Function2.
- Memory 1:** A hex dump of memory starting at address 0x3c. The dump shows four lines of memory, each consisting of 16 bytes of zeros.
- Command:** Displays compiler messages and command-line options. It includes the size limit message and the ASSIGN command.

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 128 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
  
```

# Example 3: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following windows:

- Registers**: Shows the current register values. R1 is highlighted in blue.
- Disassembly**: Shows the assembly code for the subroutine calls. The instruction at address 0x00000004 is highlighted in yellow: `0x00000004 E28FD070 ADD R13,PC,#0x00000070`. This is the subroutine call instruction to Function1.
- OpCodes.s**: Shows the source code for the assembly file. It includes sections for SubroutineCalls, Main, and Function1, along with dummy operations and stack pointer setup.
- Command**: Displays build messages: "Restricted Version with 32768 Byte Code Size Limit" and "Currently used: 128 Bytes (0%)".
- Memory 1**: Shows memory starting at address 0x0000003C filled with zeros.

```

AREA SubroutineCalls, CODE, READONLY
ENTRY
;
Main    MOV r1,#0x10      ;just a dummy operation
        ADR r13,stack   ;set up the stack pointer
;
        STR r15,[r13,#-4]! ;pre-decrement the stack pointer
        B Function1       ;jump to Function1
;
        MOV r3,#0x30      ;just a dummy operation
loop    B loop
;
Function1 MOV r2,#0x20      ;just a dummy operation
;
        STR r15,[r13,#-4]! ;pre-decrement the stack pointer
        B Function2       ;jump to Function2
;
        MOV r5,#0x50      ;just a dummy operation

```

# Example 3: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Registers:** Shows the current values of various registers, including R13 (SP) at 0x0000007C.
- Disassembly:** Displays the assembly code for the `OpCodes.s` file. The highlighted instruction is `STR r15,[r13,#-4]!`. A callout bubble states: "In case of STR instruction, the pipeline effect is 12 bytes, not 8 bytes."
- OpCodes.s:** The source code contains a main loop and two subroutines, `Function1` and `Function2`.
- Memory:** A memory dump window shows the memory starting at address 0x0000003C, all filled with zeros.
- Command:** A command window showing a map operation and an error message: "\*\*\* error 10: Syntax error".

**In case of STR instruction,  
the pipeline effect is 12  
bytes, not 8 bytes.**

**PC value + pipeline effect**  
 $= 8 + 12 = 20$   
 $= 0x00000014$

# Example 3: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Registers:** Shows the current values of all ARM registers. The PC register is highlighted with a blue selection bar.
- Disassembly:** Displays the assembly code for the program. The instruction at address 0x0000000C is highlighted in yellow, indicating it is the current instruction being executed.
- Source Editor:** Shows the source code file "OpCodes.s". The assembly code in the source editor matches the disassembly view.
- Memory:** A memory dump window showing memory starting at address 0x0000003C. The value at address 0x00000078 is highlighted in red.
- Command Window:** Shows the command map and any errors or logs.

```

AREA SubroutineCalls, CODE, READONLY
ENTRY
;
Main    MOV r1,#0x10      ;just a dummy operation
        ADR r13,stack     ;set up the stack pointer
        STR r15,[r13,-4]! ;pre-decrement the stack pointer
        B Function1       ;jump to Function1
        MOV r3,#0x30      ;just a dummy operation
loop    B loop             ;loop
;
Function1 MOV r2,#0x20      ;just a dummy operation
        STR r15,[r13,-4]! ;pre-decrement the stack pointer
        B Function2       ;jump to Function2
        MOV r5,#0x50      ;just a dummy operation

```

# Example 3: Subroutine call and Return

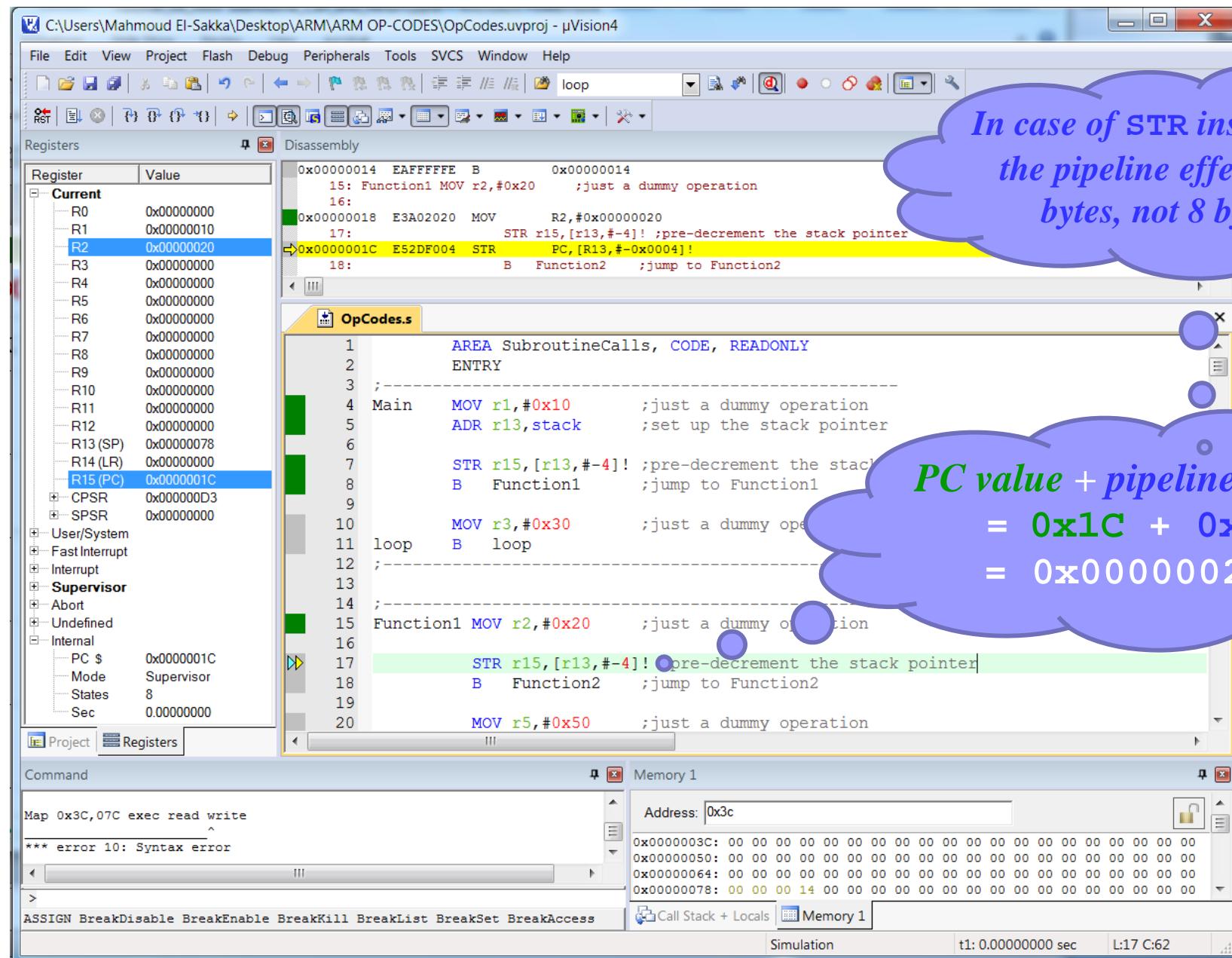
The screenshot shows the µVision4 IDE interface with the following windows:

- Registers**: Shows the current register values, including R0-R14 and CPSR/SPSR. The PC register is highlighted with the value 0x00000018.
- Disassembly**: Displays the assembly code for the program. The instruction at address 0x00000018 is highlighted in yellow: `MOV R2, #0x00000020`. This is followed by a comment: `;just a dummy operation`.
- OpCodes.s**: The source code file containing the assembly instructions. It defines an area named `SubroutineCalls` and contains the following code:
 

```

1 AREA SubroutineCalls, CODE, READONLY
2 ENTRY
3 ;
4 Main MOV r1, #0x10 ;just a dummy operation
5 ADR r13, stack ;set up the stack pointer
6
7 STR r15, [r13, #-4]! ;pre-decrement the stack pointer
8 B Function1 ;jump to Function1
9
10 MOV r3, #0x30 ;just a dummy operation
11 loop B loop
12 ;
13 ;
14 ;
15 Function1 MOV r2, #0x20 ;just a dummy operation
16
17 STR r15, [r13, #-4]! ;pre-decrement the stack pointer
18 B Function2 ;jump to Function2
19
20 MOV r5, #0x50 ;just a dummy operation
      
```
- Memory 1**: A memory dump window showing memory starting at address 0x0000003C. The value at address 0x00000078 is highlighted in yellow and shows the value 14, which corresponds to the PC value of 0x00000018 seen in the Registers window.
- Command**: A terminal-like window showing the command `Map 0x3C,07C exec read write` and an error message: `*** error 10: Syntax error`.

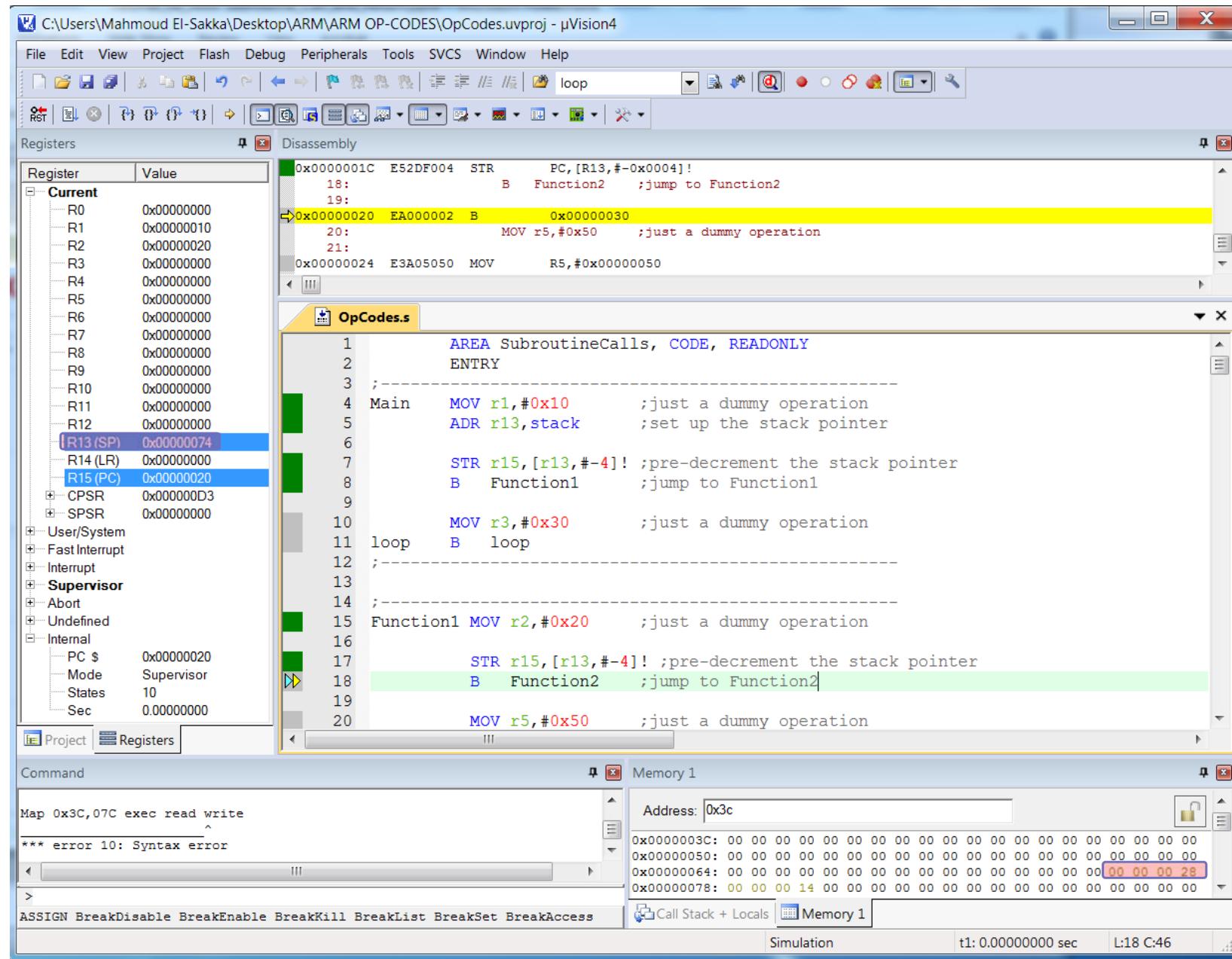
# Example 3: Subroutine call and Return



In case of STR instruction,  
the pipeline effect is 12  
bytes, not 8 bytes.

$$\begin{aligned}
 & \text{PC value} + \text{pipeline effect} \\
 &= 0x1C + 0xC \\
 &= 0x00000028
 \end{aligned}$$

# Example 3: Subroutine call and Return



# Example 3: Subroutine call and Return

C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

RST DCD Screenshot Stop Run Break Step Backward Step Forward Stop Break

Registers Disassembly

Register	Value
R0	0x00000000
R1	0x00000010
R2	0x00000020
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000074
R14 (LR)	0x00000000
R15 (PC)	0x00000030
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x00000030
Mode	Supervisor
States	13
Sec	0.00000000

OpCodes.s

```

22: LDR r12,[r13],#4 ;get the PC and post-increment the stack pointer
23: SUB r15,r12,#4 ;modify the restored PC (because it is 4 too big)
24: ;and return from Function1
25: ;
26: ;
27: ;
28: Function2 MOV r4,#0x40 ;just a dummy operation
29: ;
30: LDR r12,[r13],#4 ;get the PC and post-increment the stack pointer
31: SUB r15,r12,#4 ;modify the restored PC (because it is 4 too big)
32: ;and return from Function2
33: ;
34: ;
35: ;
36: AREA SubroutineCalls, DATA, READWRITE
37: ;
38: SPACE 0x40 ;reserved room for the stack to grow up
39: stack DCD 0x0 ;the base of the stack
40: ;
41: ;

```

Command

```

Map 0x3C,07C exec read write
*** error 10: Syntax error
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess

```

Memory 1

Address:	0x3c
0x0000003C:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28
0x00000078:	00 00 00 14 00 00 00 00 00 00 00 00 00 00 00 00

Call Stack + Locals Memory 1

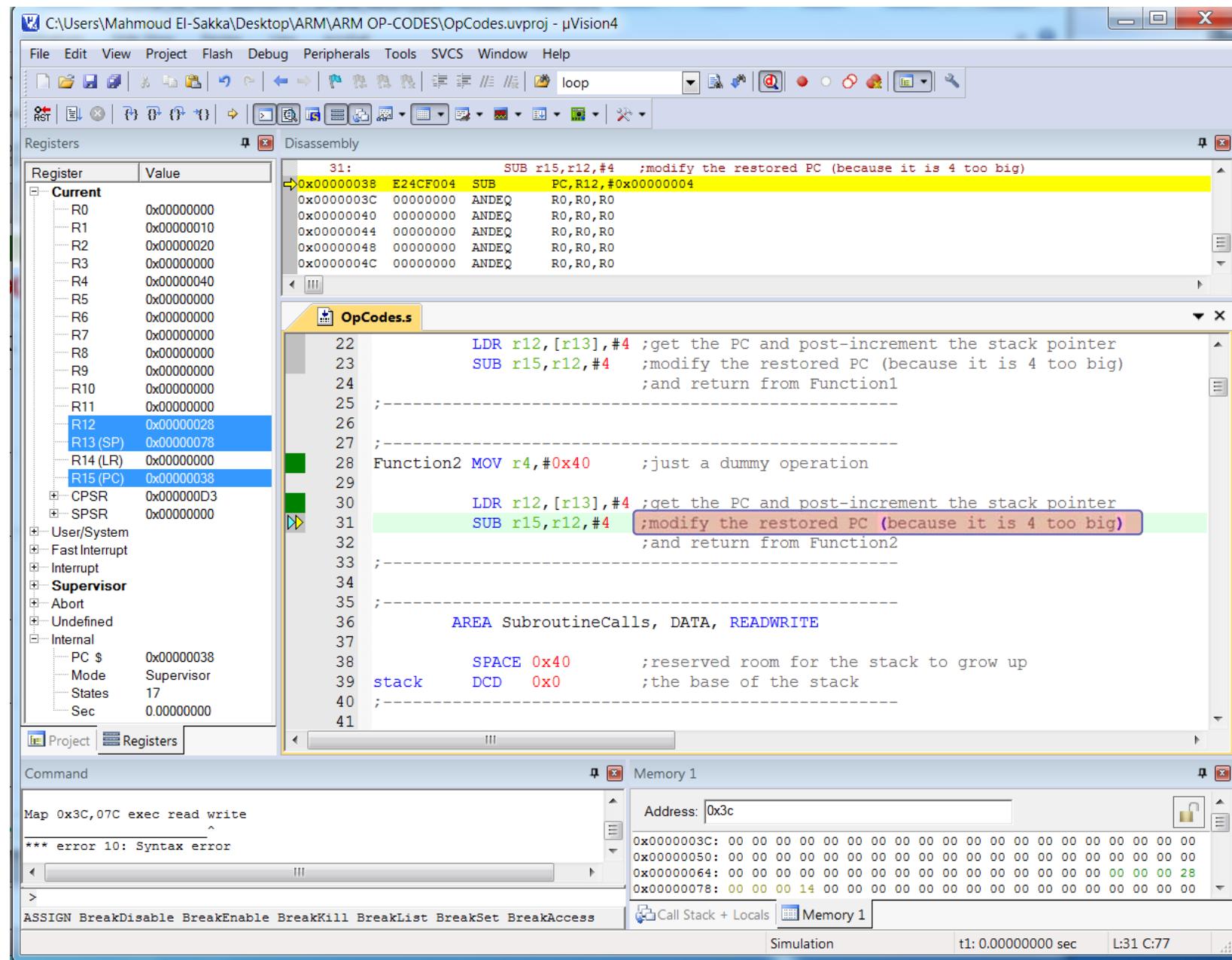
Simulation t1: 0.0000000 sec L:28 C:51

# Example 3: Subroutine call and Return

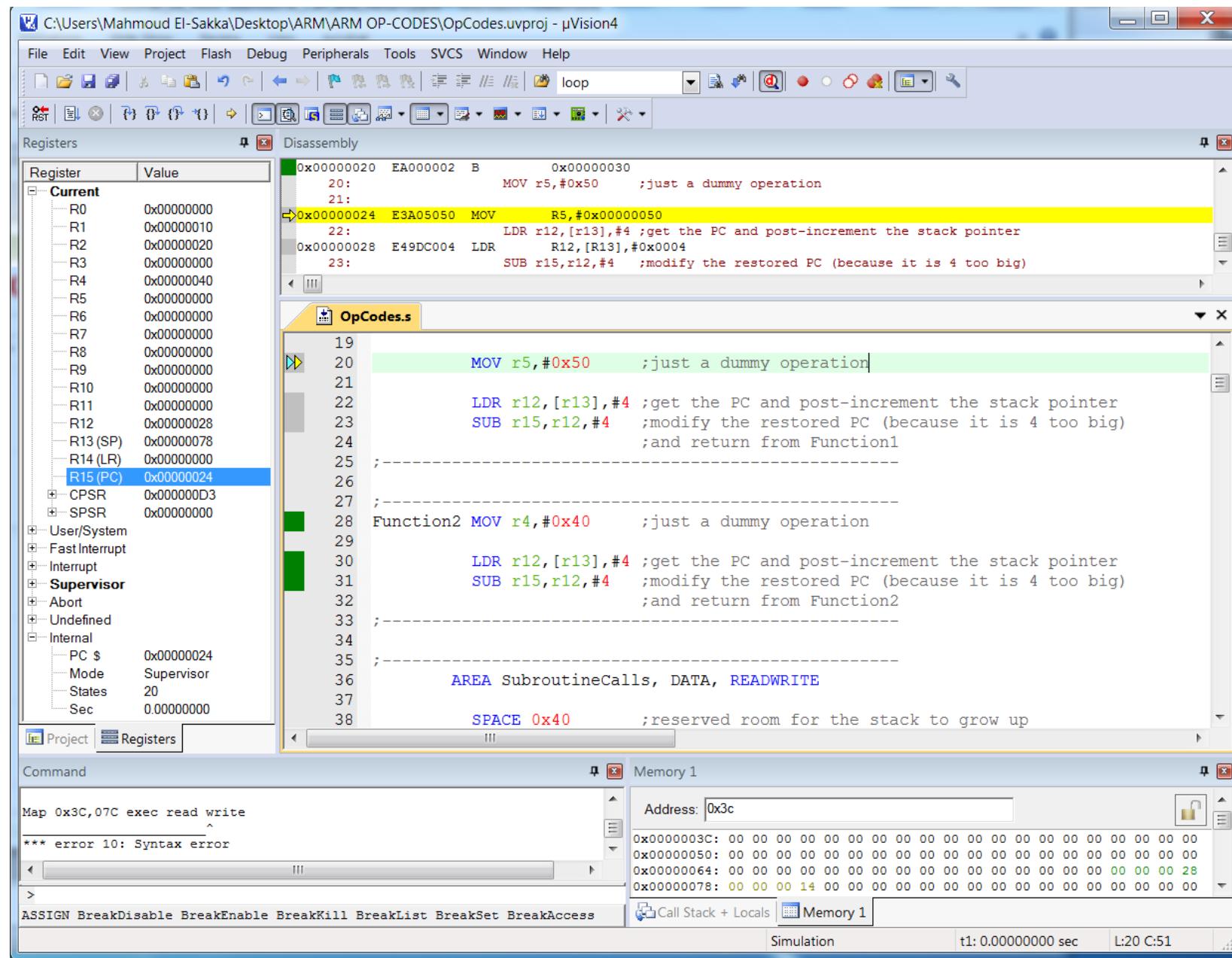
The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbars:** Standard toolbar with icons for file operations, project management, and simulation.
- Registers Window:** Shows the current register values. R4 is highlighted with a blue border and has the value 0x00000040. R13 (SP) is highlighted with a pink border and has the value 0x00000074. R15 (PC) is highlighted with a blue border and has the value 0x00000034.
- Disassembly Window:** Displays the assembly code for the subroutine. The highlighted instruction is LDR r12,[r13],#4;get the PC and post-increment the stack pointer at address 0x00000034.
- Source Editor:** Shows the source code for OpCodes.s. It contains two functions: Function1 and Function2. Function1 is a dummy operation. Function2 performs a subroutine call and return. The assembly code in the source editor matches the disassembly window.
- Memory Window:** Shows memory starting at address 0x0000003C. The byte at 0x00000064 is highlighted with a pink border and has the value 28 (hex).
- Command Window:** Displays a command-line interface with errors related to mapping memory.
- Bottom Status Bar:** Simulation, t1: 0.00000000 sec, L30 C:76

## Example 3: Subroutine call and Return



## Example 3: Subroutine call and Return



# Example 3: Subroutine call and Return

C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

RST loop

Registers

Register	Value
R0	0x00000000
R1	0x00000010
R2	0x00000020
R3	0x00000000
R4	0x00000040
<b>R5</b>	<b>0x00000050</b>
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000028
<b>R13 (SP)</b>	<b>0x00000078</b>
R14 (LR)	0x00000000
<b>R15 (PC)</b>	<b>0x00000028</b>
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
<b>Supervisor</b>	
Abort	
Undefined	
Internal	
PC \$	0x00000028
Mode	Supervisor
States	21
Sec	0.00000000

Disassembly

```

0x00000020 EA000002 B    0x00000030
  20:          MOV r5,#0x50      ;just a dummy operation
  21:
0x00000024 E3A05050 MOV   R5,#0x00000050
  22:          LDR r12,[r13],#4 ;get the PC and post-increment the stack pointer
→ 0x00000028 E49DC004 LDR   R12,[R13],#0x0004
  23:          SUB r15,r12,#4   ;modify the restored PC (because it is 4 too big)
;
```

OpCodes.s

```

19
20          MOV r5,#0x50      ;just a dummy operation
21
22          LDR r12,[r13],#4 ;get the PC and post-increment the stack pointer
23          SUB r15,r12,#4   ;modify the restored PC (because it is 4 too big)
24          ;and return from Function1
25 ;
26
27 ;
28 Function2 MOV r4,#0x40      ;just a dummy operation
29
30          LDR r12,[r13],#4 ;get the PC and post-increment the stack pointer
31          SUB r15,r12,#4   ;modify the restored PC (because it is 4 too big)
32          ;and return from Function2
33 ;
34
35 ;
36 AREA SubroutineCalls, DATA, READWRITE
37
38 SPACE 0x40      ;reserved room for the stack to grow up
;
```

Memory 1

Address: 0x3c

```

0x0000003C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28
0x00000078: 00 00 00 14 00 00 00 00 00 00 00 00 00 00 00 00
;
```

Call Stack + Locals Memory 1

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess

Simulation t1: 0.00000000 sec L:22 C:76

# Example 3: Subroutine call and Return

C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - μVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

RST Disassembly Registers

Register	Value
R0	0x00000000
R1	0x00000010
R2	0x00000020
R3	0x00000000
R4	0x00000040
R5	0x00000050
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000014
R13 (SP)	0x0000007C
R14 (LR)	0x00000000
R15 (PC)	0x0000002C
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x0000002C
Mode	Supervisor
States	24
Sec	0.00000000

OpCodes.s

```

19
20     MOV r5,#0x50      ;just a dummy operation
21
22     LDR r12,[r13],#4 ;get the PC and post-increment the stack pointer
23     SUB r15,r12,#4   ;modify the restored PC (because it is 4 too big)
24     ;and return from Function1
25     ;-----
26
27     ;-----
28 Function2 MOV r4,#0x40      ;just a dummy operation
29
30     LDR r12,[r13],#4 ;get the PC and post-increment the stack pointer
31     SUB r15,r12,#4   ;modify the restored PC (because it is 4 too big)
32     ;and return from Function2
33     ;-----
34
35     ;-----
36     AREA SubroutineCalls, DATA, READWRITE
37
38     SPACE 0x40          ;reserved room for the stack to grow up

```

Map 0x3C,07C exec read write  
\*\*\* error 10: Syntax error

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess

Memory 1

Address:	0x3c
0x0000003C:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28
0x00000078:	00 00 00 14 00 00 00 00 00 00 00 00 00 00 00 00

Call Stack + Locals Memory 1 Simulation t1: 0.00000000 sec L:23 C:77

# Example 3: Subroutine call and Return

C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

RST Disassembly Registers OpCodes.s Memory 1

**Registers**

Register	Value
R0	0x00000000
R1	0x00000010
R2	0x00000020
R3	0x00000000
R4	0x00000040
R5	0x00000050
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000014
R13 (SP)	0x0000007C
R14 (LR)	0x00000000
<b>R15 (PC)</b>	<b>0x00000010</b>
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
<b>Supervisor</b>	
Abort	
Undefined	
Internal	
PC \$	0x00000010
Mode	Supervisor
States	27
Sec	0.00000000

**Disassembly**

```

0x00000000C EA000001 B 0x00000018
10:          MOV r3,#0x30 ;just a dummy operation
→ 0x00000010 E3A03030 MOV R3,#0x00000030
11: loop    B loop
12: ;
13: ;
14: ;
15: Function1 MOV r2,#0x20 ;just a dummy operation
16: ;
17: STR r15,[r13,-4]! ;pre-decrement the stack pointer
18: B Function2 ;jump to Function2
19: ;
20: MOV r5,#0x50 ;just a dummy operation
21: ;
22: LDR r12,[r13],#4 ;get the PC and post-increment the stack pointer
23: SUB r15,r12,#4 ;modify the restored PC (because it is 4 too big)
   ;and return from Function1
24: ;
25: ;
26: ;
27: ;
28: Function2 MOV r4,#0x40 ;just a dummy operation

```

**OpCodes.s**

**Memory 1**

Address: 0x3c

0x0000003C:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28
0x00000078:	00 00 00 14 00 00 00 00 00 00 00 00 00 00 00 00

Call Stack + Locals Memory 1

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess

Simulation t1: 0.00000000 sec L:10 C:51

# Example 3: Subroutine call and Return

C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

RST Disassembly Registers OpCodes.s Memory 1 Command

**Registers**

Register	Value
R0	0x00000000
R1	0x00000010
R2	0x00000020
<b>R3</b>	<b>0x00000030</b>
R4	0x00000040
R5	0x00000050
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000014
R13 (SP)	0x0000007C
R14 (LR)	0x00000000
<b>R15 (PC)</b>	<b>0x00000014</b>
CPSR	0x000000D3
SPSR	0x00000000

**Disassembly**

```

11: loop    B    loop
12: ;
13:
14: ;
0x00000014  EAFFFFFE  B      0x00000014
15: Function1 MOV r2,#0x20 ;just a dummy operation
16:

OpCodes.s
9
10      MOV r3,#0x30 ;just a dummy operation
11  loop   B    loop
12  ;
13:
14;
15: Function1 MOV r2,#0x20 ;just a dummy operation
16:
17      STR r15,[r13,-4]! ;pre-decrement the stack pointer
18      B    Function2 ;jump to Function2
19:
20      MOV r5,#0x50 ;just a dummy operation
21:
22      LDR r12,[r13],#4 ;get the PC and post-increment the stack pointer
23      SUB r15,r12,#4 ;modify the restored PC (because it is 4 too big)
24      ;and return from Function1
25  ;-----from-----
26:
27  ;
28: Function2 MOV r4,#0x40 ;just a dummy operation

```

**Memory 1**

Address: 0x3c

0x0000003C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28
0x00000078: 00 00 00 14 00 00 00 00 00 00 00 00 00 00 00 00

Call Stack + Locals Memory 1

Map 0x3C,07C exec read write  
\*\*\* error 10: Syntax error

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess

Simulation t1: 0.00000000 sec L11 C:17



## Example 4: Subroutine call and Return

Calling a none leaf subroutine using **B** instruction and a stack without using **r14**

```
Main    MOV r1,#0x10          ; just a dummy operation
       ADR r13,stack         ; set up the stack pointer
       STR r15,[r13,#-4]!    ; pre-decrement the stack pointer
       B  Function1           ; jump to Function1 (B not BL)
       MOV r3,#0x30          ; just a dummy operation
loop    B   loop
;-----
;-----  
Function1 MOV r2,#0x20          ; just a dummy operation
       STR r15,[r13,#-4]!    ; pre-decrement the stack pointer
       NOP                  ; NOP
       B  Function2           ; jump to Function2 (B not BL)
       MOV r5,#0x50          ; just a dummy operation
       LDR r12,[r13],#4      ; get the PC and post-increment the stack pointer
       SUB r15,r12,#4        ; modify the restored PC (because it is 4 too big)
       ; and return from Function1
;-----  
Function2 MOV r4,#0x40          ; just a dummy operation
       LDR r12,[r13],#4      ; get the PC and post-increment the stack pointer
       SUB r15,r12,#4        ; modify the restored PC (because it is 4 too big)
       ; and return from Function2
;-----  
stack   SPACE 0x40          ; reserved room for the stack to grow up
       DCD 0x0               ; the base of the stack
```

What is the effect of adding these two NOP instructions as shown?

What does it need to be done here?

What does it need to be done here?



## Example 4: Subroutine call and Return

Calling a none leaf subroutine using **B** instruction and a stack without using **r14**

```
Main    MOV r1,#0x10           ; just a dummy operation
       ADR r13,stack          ; set up the stack pointer
       STR r15,[r13,#-4]!     ; pre-decrement the stack pointer
       NOP
       B Function1            ; jump to Function1 (B not BL)
       MOV r3,#0x30            ; just a dummy operation
loop    B loop
;-----
;-----
```

```
Function1 MOV r2,#0x20           ; just a dummy operation
          STR r15,[r13,#-4]!   ; pre-decrement the stack pointer
          NOP
          B Function2          ; jump to Function2 (B not BL)
          MOV r5,#0x50            ; just a dummy operation
          LDR r15,[r13],#4        ; get the PC and post-increment the stack pointer
                                      ; and return from Function1
;-----
;-----
```

```
Function2 MOV r4,#0x40           ; just a dummy operation
          LDR r15,[r13],#4        ; get the PC and post-increment the stack pointer
                                      ; and return from Function2
;-----
;-----
```

```
stack   SPACE 0x40             ; reserved room for the
          DCD 0x0                ; the base of the stack
;-----
```

Try this example by  
yourself!!

You need to re-map the memory to  
make the stack space read/write  
enabled (Debug/Memory Map)



## Example 5: Subroutine call and Return

Calling a none leaf subroutine using **BL** instruction, **r14** and a stack



# Example 5: Subroutine call and Return

Calling a none leaf subroutine using **BL** instruction, **r14** and a stack

```
Main    MOV r1,#0x10      ; just a dummy operation
        ADR r13,stack    ; set up the stack pointer
        BL  Function1     ; jump to Function1
        MOV r3,#0x30      ; just a dummy operation
loop    B   loop
;
;
Function1 MOV r2,#0x20      ; just a dummy operation
          STR r14,[r13,#-4]! ; pre-decrement the stack pointer
          BL  Function2     ; jump to Function2
          MOV r5,#0x50      ; just a dummy operation
          LDR r14,[r13],#4    ; get the PC and post-increment the stack pointer
          MOV r15,r14      ; return from Function1
;
;
Function2 MOV r4,#0x40      ; just a dummy operation
          MOV r15,r14      ; return from Function2
;
;
stack   SPACE 0x40      ; reserved room for the stack to grow up
          DCD 0x0       ; the base of the stack
;
```

These two instructions  
can be combined in one  
instruction:

LDR r15,[r13],#4

# Example 5: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- File Menu:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Open, Save, Build, Run, Stop, and various simulation and memory manipulation tools.
- Registers Window:** Shows the current values of all ARM registers (R0-R15, CPSR, SPSR) and mode information (Mode: Supervisor).
- Disassembly Window:** Displays the assembly code for the current program. The highlighted section shows the main loop and a subroutine call to Function1.
 

```

      0x00000000 E3A01010 MOV     R1,#0x00000001
      5:          ADR r13,stack ;set up the stack pointer
      6:
      0x00000004 E28FD068 ADD     R13,PC,#0x00000068
      7:          BL  Function1 ;jump to Function1
      8:
      0x00000008 EB000001 BL     0x00000014
    
```
- Source Editor:** Shows the source code file OpCodes.s containing the assembly code for Main and Function1.
 

```

1   AREA SubroutineCalls, CODE, READONLY
2
3   ENTRY
4
4   Main   MOV r1,#0x10 ;just a dummy operation
5       ADR r13,stack ;set up the stack pointer
6
7       BL  Function1 ;jump to Function1
8
9       MOV r3,#0x30 ;just a dummy operation
10  loop   B  loop
11
12
13
14  Function1 MOV r2,#0x20 ;just a dummy operation
15
16      STR r14,[r13,-4]! ;pre-decrement the stack pointer
17      BL  Function2 ;jump to Function2
18
19      MOV r5,#0x50 ;just a dummy operation
20
    
```
- Command Window:** Displays build messages and command history. It shows a restricted version warning and current usage statistics.
 

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 120 Bytes (0%)
    
```
- Memory Window:** Shows memory dump starting at address 0x3c.
 

Address	Value
0x0000003C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000078	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Status Bar:** Simulation, t1: 0.0000000 sec, L: 4 C: 53

# Example 5: Subroutine call and Return

C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

RST Disassembly Registers OpCodes.s Command Memory 1

**Registers**

Register	Value
Current	
R0	0x00000000
<b>R1</b>	<b>0x00000010</b>
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
<b>R15 (PC)</b>	<b>0x00000004</b>
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
<b>Supervisor</b>	
Abort	
Undefined	
Internal	
PC \$	0x00000004
Mode	Supervisor
States	1
Sec	0.00000000

**Disassembly**

```

0x00000000 E3A01010 MOV R1,#0x00000010
5:          ADR r13,stack ;set up the stack pointer
6:
→0x00000004 E28FD068 ADD R13,PC,#0x00000068
7:          BL Function1 ;jump to Function1
8:
0x00000008 EB000001 BL 0x00000014

```

**OpCodes.s**

```

1      AREA SubroutineCalls, CODE, READONLY
2
3      ENTRY
4      Main    MOV r1,#0x10      ;just a dummy operation
5      ADR r13,stack ;set up the stack pointer
6
7      BL Function1 ;jump to Function1
8
9      MOV r3,#0x30      ;just a dummy operation
10     loop   B loop
11
12
13
14     Function1 MOV r2,#0x20      ;just a dummy operation
15
16     STR r14,[r13,-4]! ;pre-decrement the stack pointer
17     BL Function2 ;jump to Function2
18
19     MOV r5,#0x50      ;just a dummy operation
20

```

**Command**

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 120 Bytes (0%)

```

**Memory 1**

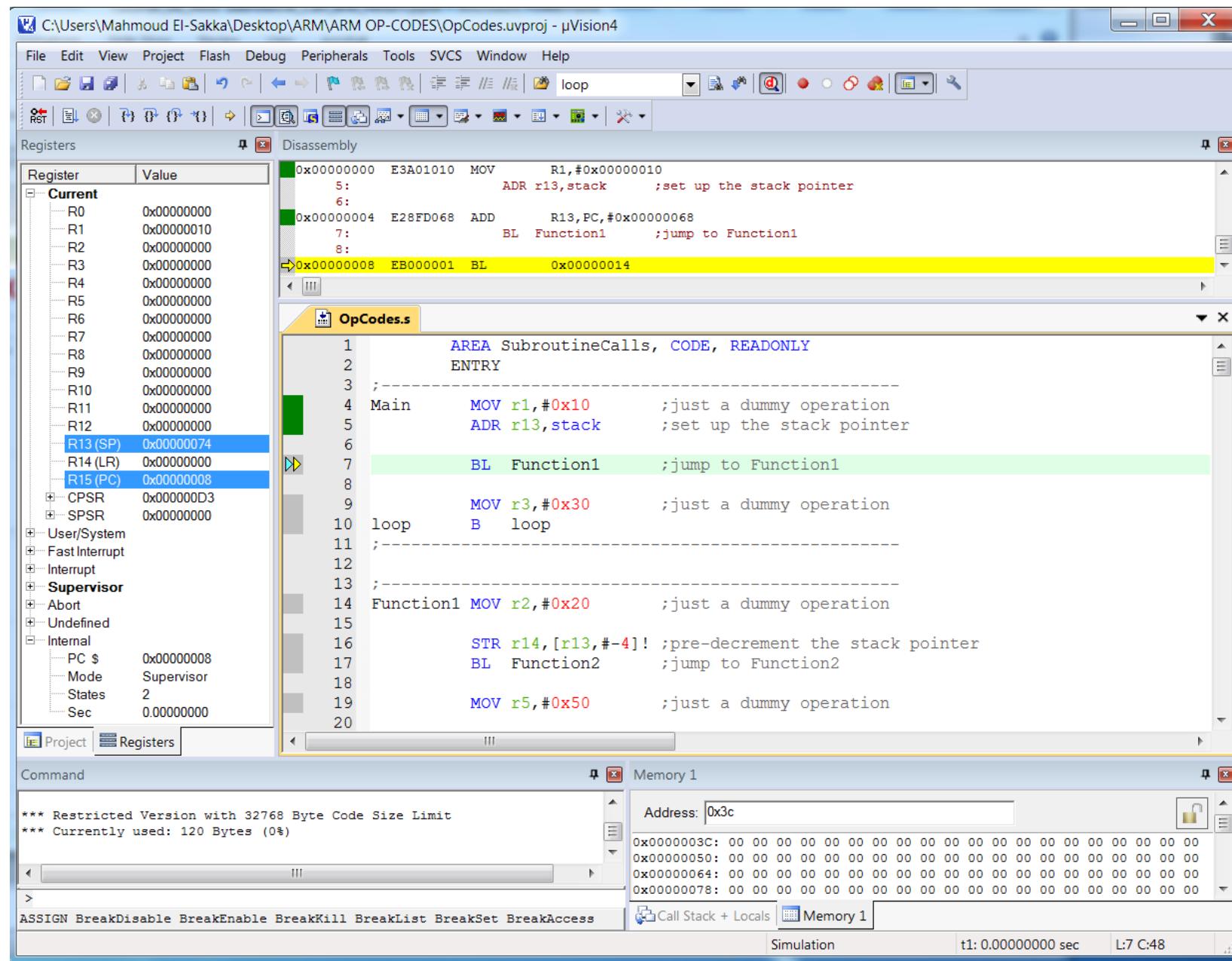
Address: 0x3c

0x0000003C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000078: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Call Stack + Locals Memory 1

Simulation t1: 0.0000000 sec L:5 C:55

# Example 5: Subroutine call and Return



# Example 5: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Open, Save, Build, Run, Stop, and various simulation and memory manipulation tools.
- Registers Window:** Shows the current values of all ARM registers. The R14 (LR) register is highlighted in blue, containing the value 0x000000C.
- Disassembly Window:** Displays the assembly code for the program. The current instruction at address 0x00000014 is highlighted in yellow: `MOV R2,#0x00000020`. The assembly code includes:
 

```

        0x00000010  EAFFFFFE B      0x00000010
          14: Function1 MOV r2,#0x20 ;just a dummy operation
          15:
        =>0x00000014  E3A02020 MOV     R2,#0x00000020
          16:           STR r14,[r13,-4]! ;pre-decrement the stack pointer
        0x00000018  E52DE004 STR     R14,[R13,-0x0004]!
          17:           BL Function2 ;jump to Function2
      
```
- Source Editor:** Shows the source code file OpCodes.s. The code defines two functions: Main and Function1. Function1 contains a call to Function2. The assembly code for Function1 is highlighted in green:
 

```

1          AREA SubroutineCalls, CODE, READONLY
2
3          ;-----
4          Main    MOV r1,#0x10      ;just a dummy operation
5          ADR r13,stack      ;set up the stack pointer
6
7          BL Function1      ;jump to Function1
8
9          MOV r3,#0x30      ;just a dummy operation
10         loop   B loop
11
12
13
14         Function1 MOV r2,#0x20      ;just a dummy operation
15
16         STR r14,[r13,-4]! ;pre-decrement the stack pointer
17         BL Function2      ;jump to Function2
18
19         MOV r5,#0x50      ;just a dummy operation
20
      
```
- Command Window:** Displays build messages:
 

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 120 Bytes (0%)
      
```
- Memory Window:** Shows memory starting at address 0x3c. The first few bytes are 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.
 

Address	Value
0x0000003C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000078	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

# Example 5: Subroutine call and Return

Screenshot of the µVision4 IDE showing the assembly code for a subroutine call and return example.

**Registers:**

Register	Value
R0	0x00000000
R1	0x00000010
<b>R2</b>	<b>0x00000020</b>
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
<b>R13 (SP)</b>	<b>0x00000074</b>
<b>R14 (LR)</b>	<b>0x000000C</b>
<b>R15 (PC)</b>	<b>0x00000018</b>
CPSR	0x000000D3
SPSR	0x00000000

**Disassembly:**

```

0x00000010 EAFFFFFE B 0x00000010
 14: Function1 MOV r2,#0x20 ;just a dummy operation
 15:
0x00000014 E3A02020 MOV R2,#0x00000020
 16:           STR r14,[r13,-4]! ;pre-decrement the stack pointer
0x00000018 E52DE004 STR R14,[R13,-0x0004]!
 17:           BL Function2 ;jump to Function2

```

**OpCodes.s:**

```

1      AREA SubroutineCalls, CODE, READONLY
2
3      ENTRY
4      Main    MOV r1,#0x10      ;just a dummy operation
5          ADR r13,stack      ;set up the stack pointer
6          stack
7          BL Function1      ;jump to Function1
8
9          MOV r3,#0x30      ;just a dummy operation
10     loop    B loop
11
12
13
14     Function1 MOV r2,#0x20      ;just a dummy operation
15
16     STR r14,[r13,-4]! ;pre-decrement the stack pointer
17     BL Function2      ;jump to Function2
18
19     MOV r5,#0x50      ;just a dummy operation
20

```

**Memory 1:**

Address: 0x3c

```

0x0000003C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000078: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

# Example 5: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following windows:

- Registers**: Shows the current values of various ARM registers. The R13 (SP) register is highlighted with a blue selection bar.
- Disassembly**: Displays the assembly code for the program. The current instruction at address 0x0000001C is highlighted in yellow: `BL 0x0000002C`. This is followed by a comment: `;jump to Function2`.
- OpCodes.s**: Shows the source code for the assembly file. It defines two functions: Main and Function1. The Main function initializes R13 to the stack and calls Function1. Function1 performs some dummy operations and then calls Function2. Function2 also contains some dummy operations.
- Memory 1**: A memory dump window showing memory starting at address 0x3c. The value at 0x00000050 is highlighted in red: `00 00 00 0C`.
- Command**: A text input field containing compiler messages and command-line options.

```

AREA SubroutineCalls, CODE, READONLY
ENTRY
;
4 Main      MOV r1,#0x10      ;just a dummy operation
5           ADR r13,stack    ;set up the stack pointer
6
7           BL Function1    ;jump to Function1
8
9           MOV r3,#0x30      ;just a dummy operation
10          loop      B loop
11
13
14 Function1 MOV r2,#0x20      ;just a dummy operation
15
16           STR r14,[r13,-4]! ;pre-decrement the stack pointer
17           BL Function2    ;jump to Function2
18
19           MOV r5,#0x50      ;just a dummy operation
20

```

# Example 5: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Open, Save, Build, Run, Stop, and Simulation.
- Registers Window:** Shows the current values of all ARM registers. The R14 (LR) register is highlighted with a blue selection bar.
- Disassembly Window:** Displays the assembly code for the subroutine. The instruction at address 0x0000002C is highlighted in yellow: `MOV r4,#0x40`. A note above it says "just a dummy operation". The instruction at address 0x0000002E is highlighted in red: `MOV r15,r14`. A note above it says "return from Function2". Other instructions shown include `MOV PC,R14`, `MOV r4,#0x40`, `ANDEQ R0,R0,R0`, and `ANDEQ R0,R0,R0`.
- OpCodes.s Window:** Shows the source code for the subroutine. It includes assembly directives like LDR, MOV, and AREA, along with comments explaining the purpose of each line. The code defines a stack area starting at address 0x0000002C.
- Memory Window:** Shows memory starting at address 0x3C. The first few bytes are zeroed out, followed by a sequence of bytes starting at 0x00000064, which includes the value 0C (hex).
- Command Window:** Displays build-related messages: "\*\*\* Restricted Version with 32768 Byte Code Size Limit" and "\*\*\* Currently used: 120 Bytes (0%)".
- Status Bar:** Shows simulation information: t1: 0.00000000 sec and L:26 C:53.

# Example 5: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for Open, Save, Build, Run, and various simulation and memory manipulation tools.
- Registers Window:** Shows the current values of all ARM registers. R4 is highlighted with a blue selection bar and has a value of 0x00000040. R15 (PC) is also highlighted with a blue selection bar and has a value of 0x00000030.
- Disassembly Window:** Displays the assembly code for the program. The current instruction at address 0x00000030 is highlighted in yellow and is a MOV r14, PC. The comments indicate it is the return from Function2. Other instructions like LDR r14, [r13], #4 and MOV r15, r14 are also highlighted in green.
- Source Editor:** Shows the source code file OpCodes.s. It contains assembly language with comments explaining the operations. It includes sections for Function1 and Function2, and defines a stack area.
- Command Window:** Displays build-related messages: "\*\*\* Restricted Version with 32768 Byte Code Size Limit" and "\*\*\* Currently used: 120 Bytes (0%)".
- Memory Window:** Shows memory starting at address 0x0000003C. The first few bytes are 00 00 00 00 00 00 00 00 ...
- Bottom Status Bar:** Simulation, t1: 0.00000000 sec, L:27 C:52

# Example 5: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following components:

- Title Bar:** C:\Users\Mahmoud El-Sakka\Desktop\ARM\ARM OP-CODES\OpCodes.uvproj - µVision4
- File Menu:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for RST, Run, Stop, Break, Step, and various simulation and memory manipulation tools.
- Registers Window:** Shows the current values of all ARM registers. The PC register is highlighted with a blue selection bar.
- Disassembly Window:** Displays the assembly code for the subroutine. The current instruction at address 0x00000020 is highlighted in yellow: `MOV R5,#0x00000050`. A tooltip "operation" appears over the R5 register in the assembly line.
- OpCodes.s Window:** Shows the source code for the subroutine. It includes comments explaining the purpose of each instruction.
- Command Window:** Displays build-related messages: "\*\*\* Restricted Version with 32768 Byte Code Size Limit" and "\*\*\* Currently used: 120 Bytes (0%)".
- Memory Window:** Shows the memory dump starting at address 0x0000003C. The stack pointer (R13) is at 0x00000050, pointing to the return address 0x00000064.
- Status Bar:** Simulation, t1: 0.00000000 sec, L19 C:53

```

0x0000001C EB000002 BL 0x0000000C
19:           MOV r5,#0x50      ;just a dummy operation
20:
21:           LDR r14,[r13],#4 ;get the PC and post-increment the stack pointer
22:           LDR R14,[R13],#0x0004
23:           MOV r15,r14       ;return from Function1

;-----
;-----
26:           MOV r4,#0x40      ;just a dummy operation
27:           MOV r15,r14       ;return from Function2
28:
29:
31:           AREA SubroutineCalls, DATA, READWRITE
32:
33:           SPACE 0x40        ;reserved room for the stack to grow up
34:           stack DCD 0x0       ;the base of the stack
35:
36:
37

```

# Example 5: Subroutine call and Return

The screenshot shows the µVision4 IDE interface with the following windows:

- Registers**: Shows the current register values. R5 is highlighted in blue.
- Disassembly**: Shows the assembly code for the subroutine. The instruction at address 0x00000024, LDR R14, [R13], #4, is highlighted in yellow.
- OpCodes.s**: The source code file containing the assembly code. The same instruction is highlighted in green.
- Command**: Displays build messages: "\*\*\* Restricted Version with 32768 Byte Code Size Limit" and "\*\*\* Currently used: 120 Bytes (0%)".
- Memory 1**: A memory dump window showing the stack area. Address 0x00000050 contains the value 0x000000C0, which is highlighted in orange.

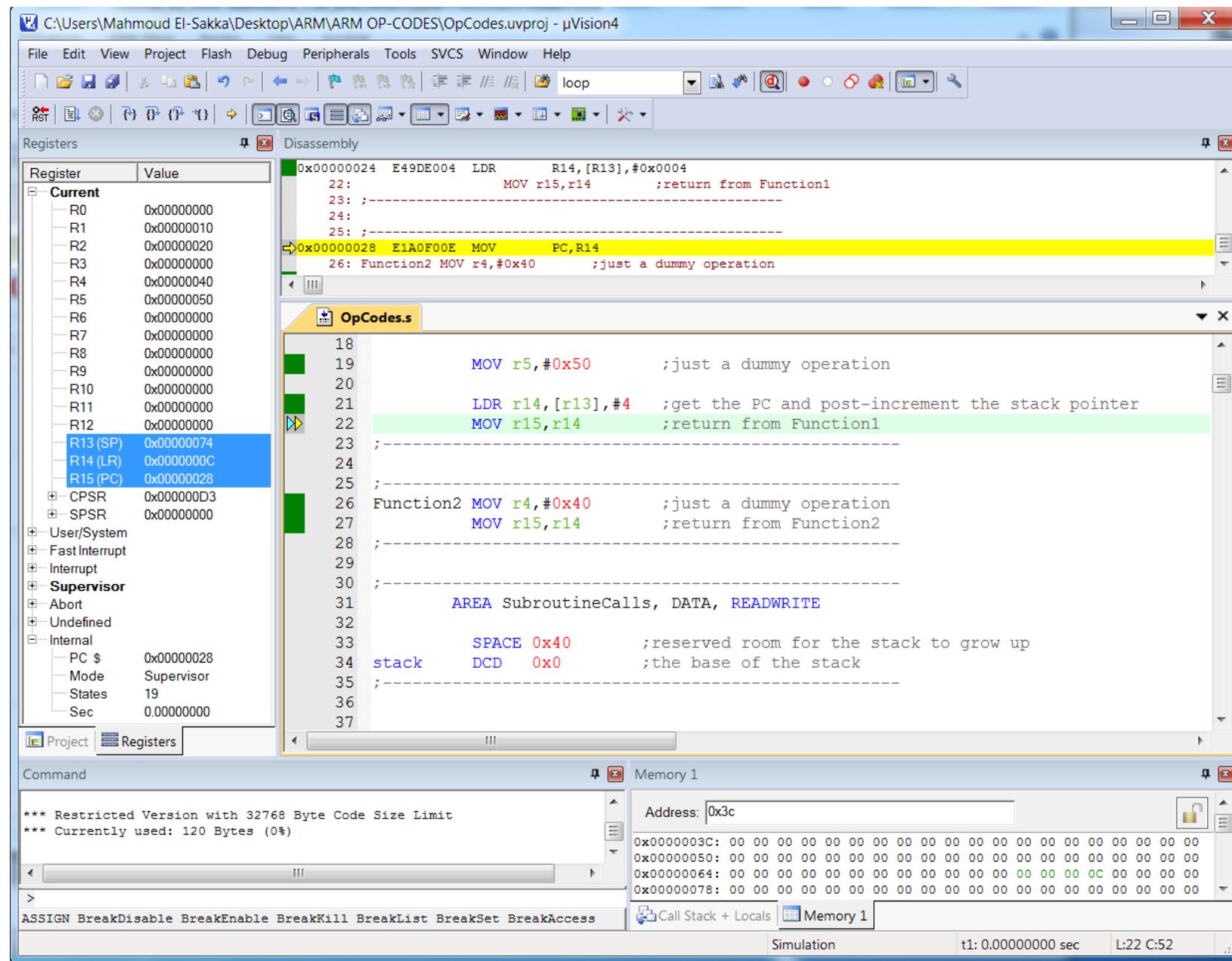
```

0x0000001C EB000002 BL      0x0000002C
19:          MOV r5,#0x50      ;just a dummy operation
20:
0x00000020 E3A05050 MOV     R5,#0x00000050
21:          LDR r14,[r13],#4  ;get the PC and post-increment the stack pointer
22:          LDR R14,[R13],#0x0004
23:          MOV r15,r14       ;return from Function1

18
19:          MOV r5,#0x50      ;just a dummy operation
20:
21:          LDR r14,[r13],#4  ;get the PC and post-increment the stack pointer
22:          MOV r15,r14       ;return from Function1
23:          -----
24:
25:          -----
26: Function2 MOV r4,#0x40      ;just a dummy operation
27:          MOV r15,r14       ;return from Function2
28:
29:          -----
30:          -----
31:          AREA SubroutineCalls, DATA, READWRITE
32:
33:          SPACE 0x40        ;reserved room for the stack to grow up
34: stack    DCD 0x0         ;the base of the stack
35:
36:
37

```

## Example 5: Subroutine call and Return



# Example 5: Subroutine call and Return

Screenshot of the µVision4 IDE showing the assembly code for an ARM subroutine call and return.

The assembly code in the Disassembly window is:

```

    0x00000008 EB000001 BL      0x00000014
    9:          MOV r3,#0x30 ;just a dummy operation
    >0x0000000C E3A03030 MOV R3,#0x00000030
    10:         loop   B loop
    11:         ;
    12:         ;
    13:         ;

```

The source code in the OpCodes.s window is:

```

    8
    9     MOV r3,#0x30 ;just a dummy operation
    10    loop   B loop
    11    ;
    12    ;
    13    ;
    14    Function1 MOV r2,#0x20 ;just a dummy operation
    15
    16    STR r14,[r13,-4]! ;pre-decrement the stack pointer
    17    BL Function2 ;jump to Function2
    18
    19    MOV r5,#0x50 ;just a dummy operation
    20
    21    LDR r14,[r13],#4 ;get the PC and post-increment the stack pointer
    22    MOV r15,r14 ;return from Function1
    23    ;
    24    ;
    25    ;
    26    Function2 MOV r4,#0x40 ;just a dummy operation
    27    MOV r15,r14 ;return from Function2

```

The Registers window shows the current register values:

Register	Value
R0	0x00000000
R1	0x00000010
R2	0x00000020
R3	0x00000000
R4	0x00000040
R5	0x00000050
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000074
R14 (LR)	0x0000000C
R15 (PC)	0x0000000C
CPSR	0x000000D3
SPSR	0x00000000

The Command window displays:

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 120 Bytes (0%)

```

The Memory window shows memory starting at address 0x3c:

Address	Value
0x0000003C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000078	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

# Example 5: Subroutine call and Return

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

RST | Disassembly | Registers | Stack | Memory | Command | Project |

**Registers**

Register	Value
R0	0x00000000
R1	0x00000010
R2	0x00000020
<b>R3</b>	<b>0x00000030</b>
R4	0x00000040
R5	0x00000050
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000074
R14 (LR)	0x0000000C
<b>R15 (PC)</b>	<b>0x00000010</b>
CPSR	0x000000D3
SPSR	0x00000000

**Disassembly**

```

10: loop      B  loop
11: ;-----
12:
13: ;-----
14: Function1 MOV r2,#0x20      ;just a dummy operation
15:

OpCodes.s
8
9      MOV r3,#0x30      ;just a dummy operation
10 loop      B  loop
11 ;-----
12
13 ;-----
14 Function1 MOV r2,#0x20      ;just a dummy operation
15
16     STR r14,[r13,-4]! ;pre-decrement the stack pointer
17     BL Function2      ;jump to Function2
18
19     MOV r5,#0x50      ;just a dummy operation
20
21     LDR r14,[r13],#4   ;get the PC and post-increment the stack pointer
22     MOV r15,r14        ;return from Function1
23 ;
24
25 ;-----
26 Function2 MOV r4,#0x40      ;just a dummy operation
27     MOV r15,r14        ;return from Function2

```

**Command**

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 120 Bytes (0%)

```

**Memory 1**

Address:	0x3c
0x0000003C:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000050:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000078:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess