

These slides are being provided with permission from the copyright for CS2208 use only. The slides must not be reproduced or provided to anyone outside of the class.

All download copies of the slides and/or lecture recordings are for personal use only. Students must destroy these copies within 30 days after receipt of final course evaluations.

Tutorial 10:

ARM Shift Instructions

Computer Science Department

CS2208: Introduction to Computer Organization and Architecture

Fall 2022-2023

Instructor: Mahmoud R. El-Sakka

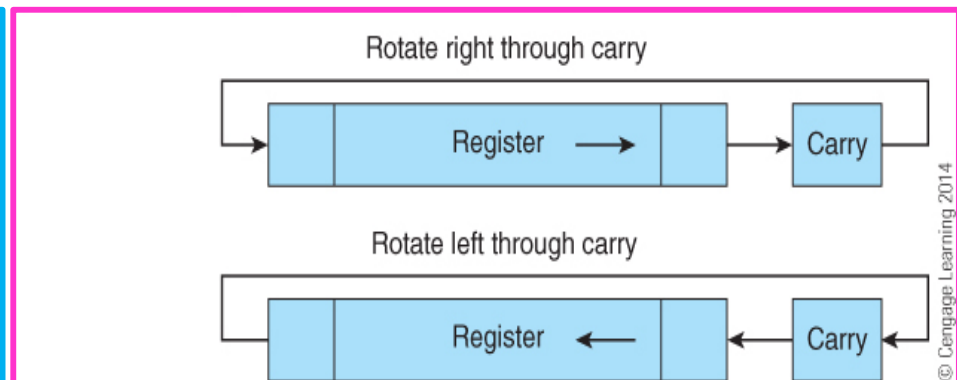
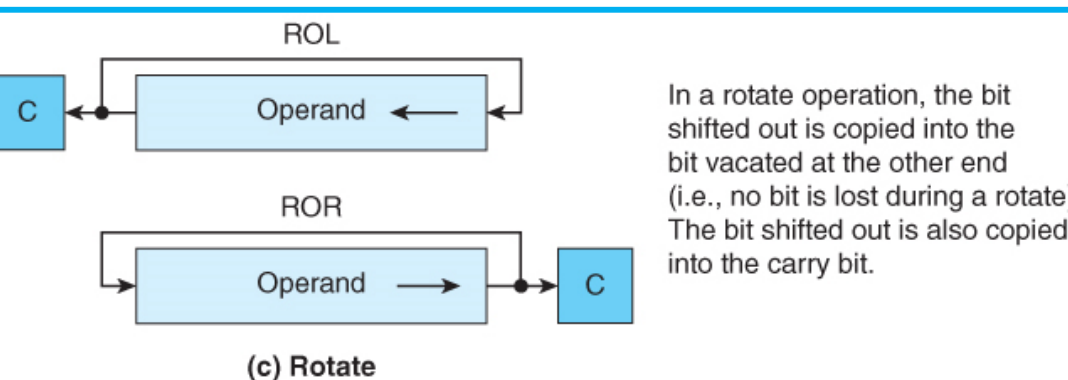
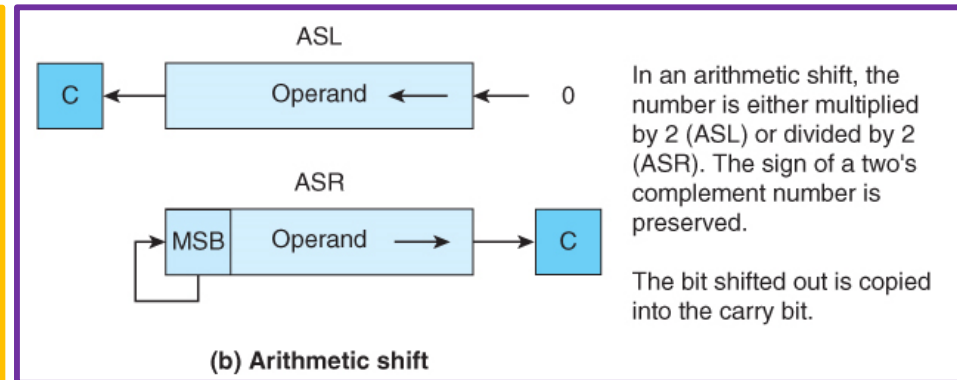
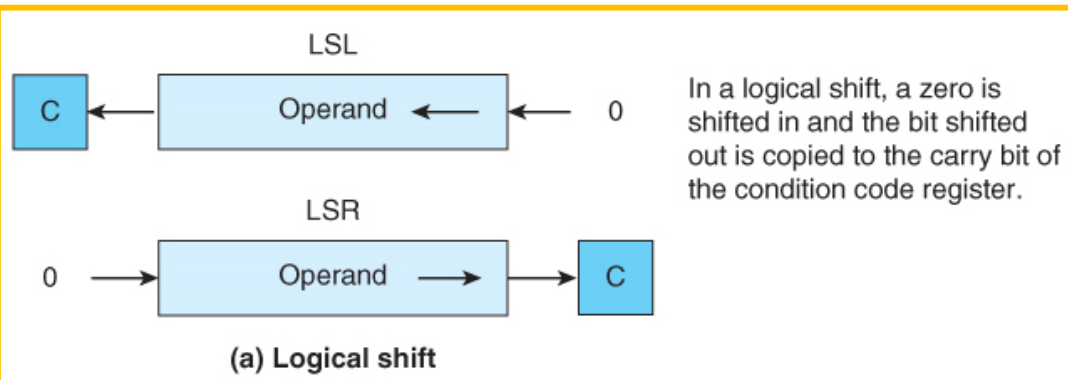
Office: MC-419

Email: elsakka@csd.uwo.ca

Phone: 519-661-2111 x86996

ARM's Data-Processing Instructions (Shift Operations)

- ❑ **Shift** operations move bits one or more places *left* or *right*.
 - **Logical shifts**
 - *insert a 0* in the vacated position.
 - **Arithmetic shifts**
 - *replicate the sign-bit* during a right shift
 - **Circular shifts**
 - *the bit shifted out of one end is shifted in the other end*
i.e., the register is treated as a ring
 - **Circular shifts through carry**
 - *included the carry bit in the shift path*



ARM's Data-Processing Instructions (Shift Operations)

- ❑ **ARM** support both *static* and *dynamic* shifts (except *rotate through carry* instruction which allows *only one single shift* per instruction)
 - In *static shift*, the number of shift places is determined *when the code is written*
 - In *static shift*, the range of the number of shift places is as follow:
 - **LSL**: the range is from **#0** to **#31** (*32 different values*)
 - **LSR**: the range is from **#1** to **#32** (*32 different values*)
 - **ASR**: the range is from **#1** to **#32** (*32 different values*)
 - **ROR**: the range is from **#1** to **#31** (*31 different values*)

The remaining value is used to encode RRX

 - **ROR** + a shift of **#0** → **RRX**
 - In *dynamic shift*, the number of shift places
 - is determined *when the code is executed, i.e., at run time*
 - If the number of dynamic shifts is ≥ 32 , zero will be stored in the destination

Only 5 bits are needed to encode the amount of shifts.

In case of **LSR** and **ASR**, the value **#32** is encoded as **00000**

ARM's Data-Processing Instructions (Shift Operations)

- ❑ **ARM** implements only the following five shifts
 - **LSL** logical shift left
 - **LSR** logical shift right
 - **ASR** arithmetic shift right
 - **ROR** rotate right
 - **RRX** rotate right through carry (one shift)
- ❑ *Other shift operations have to be synthesized by the programmer.*
 - An *arithmetic shift left* is effectively the same as a *logical shift left*
 - For a 32-bit value, an *n-bit rotate shift left* is identical to a *32 – n rotate shift right*
 - **Rotate left through carry** can be implemented by means of
`ADCS r0, r0, r0 ; add r0 to r0 with carry and set the flags`
 - The instruction means $r0 + r0 + C$, i.e., $2 \times r0 + C$, i.e.,
 - shifting left the content of r0
 - store the value of C in the vacant bit to the left, and
 - storing the shifted out bit in the carry flag

ARM's Data-Processing Instructions (Shift Operations)

- ❑ **ARM** has no explicit shift operations!!.
- ❑ **ARM** combines shifting with other data processing operations, where
 - the second operand in the arithmetic operation (i.e., the LAST parameter in the assembly arithmetic instruction) is allowed to be shifted before it is used.
 - For example,


```
ADD r0, r1, r2, LSL #1      ; [r0] ← [r1] + [r2] × 2
```

 - logically shift left the contents of r2,
 - add the result to the contents of r1, and
 - put the results in r0
- ❑ **ARM** also combines shifting with MOV and MVN operations
 - This way, a shift operation can be performed as a stand-alone operation.
 - For example,


```
MOV r3, r3, LSL #1        ; [r3] ← [r3] × 2
```
 - **ARM** provides pseudo shift instructions, which are translated to **MOV** instructions.


```
LSL r3, r3, #1           ; will be converted to MOV r3, r3, LSL #1
```

or simply

```
LSL r3, #1
```

ARM's Data-Processing Instructions (Shift Operations)

```

AREA prog1, code, READONLY
ENTRY
MOV r3, #2
LDR r1, =0xCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100 1100

LSLS r1, r1, #5
LSLS r1, r1, r3

LSRS r1, r1, #10
LSRS r1, r1, r3

ASRS r1, r1, #2
LSLS r1, r1, #15
ASRS r1, r1, #16

ASRS r1, r1, r3

RORS r1, r1, #4
RORS r1, r1, r3

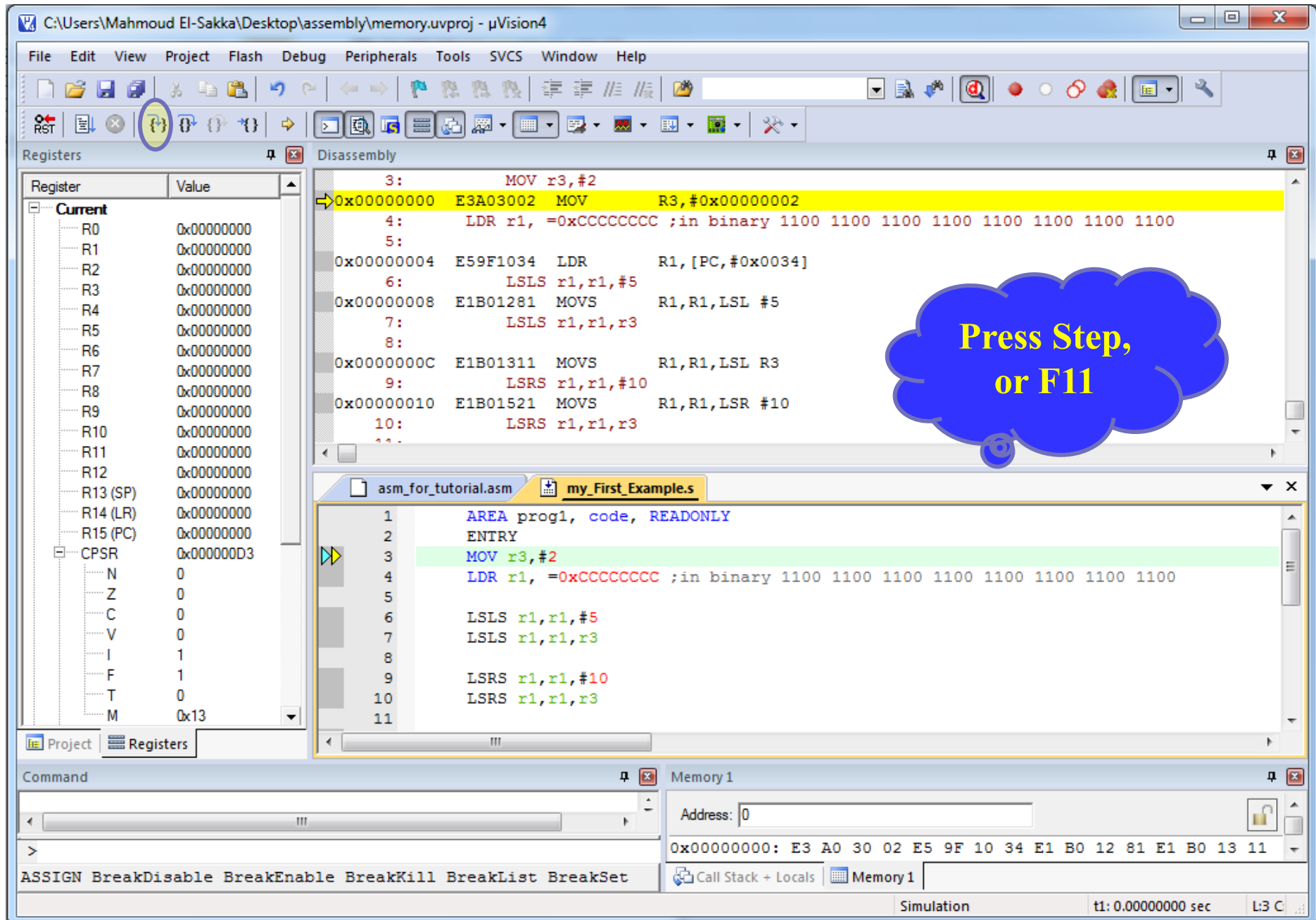
RRXS r1, r1
RRXS r1, r1
RRXS r1, r1
RRXS r1, r1
END

```

MOVS and MVNS

- ✓ Update the N, Z and C flags according to the result
- ✓ Do NOT affect the V flag

ARM's Data-Processing Instructions (Shift Operations)



The screenshot shows the µVision4 IDE interface. The top toolbar has the Step button (a square with a right-pointing arrow) circled in blue. A blue cloud bubble with the text "Press Step, or F11" points to this button. The Registers window on the left shows the current state of the ARM registers. The Disassembly window in the center shows the assembly code being executed. The Source window at the bottom shows the assembly code for "my_First_Example.s".

Registers Window:

Register	Value
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000000
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13

Disassembly Window:

```

3:      MOV r3,#2
0x00000000 E3A03002 MOV      R3,#0x00000002
4:      LDR r1, =0xCCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100 1100
5:
0x00000004 E59F1034 LDR      R1,[PC,#0x0034]
6:      LSLS r1,r1,#5
0x00000008 E1B01281 MOVS     R1,R1,LSL #5
7:      LSLS r1,r1,r3
8:
0x0000000C E1B01311 MOVS     R1,R1,LSL R3
9:      LSRS r1,r1,#10
0x00000010 E1B01521 MOVS     R1,R1,LSR #10
10:     LSRS r1,r1,r3
11:

```

Source Window (my_First_Example.s):

```

1      AREA prog1, code, READONLY
2      ENTRY
3      MOV r3,#2
4      LDR r1, =0xCCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100 1100
5
6      LSLS r1,r1,#5
7      LSLS r1,r1,r3
8
9      LSRS r1,r1,#10
10     LSRS r1,r1,r3
11

```

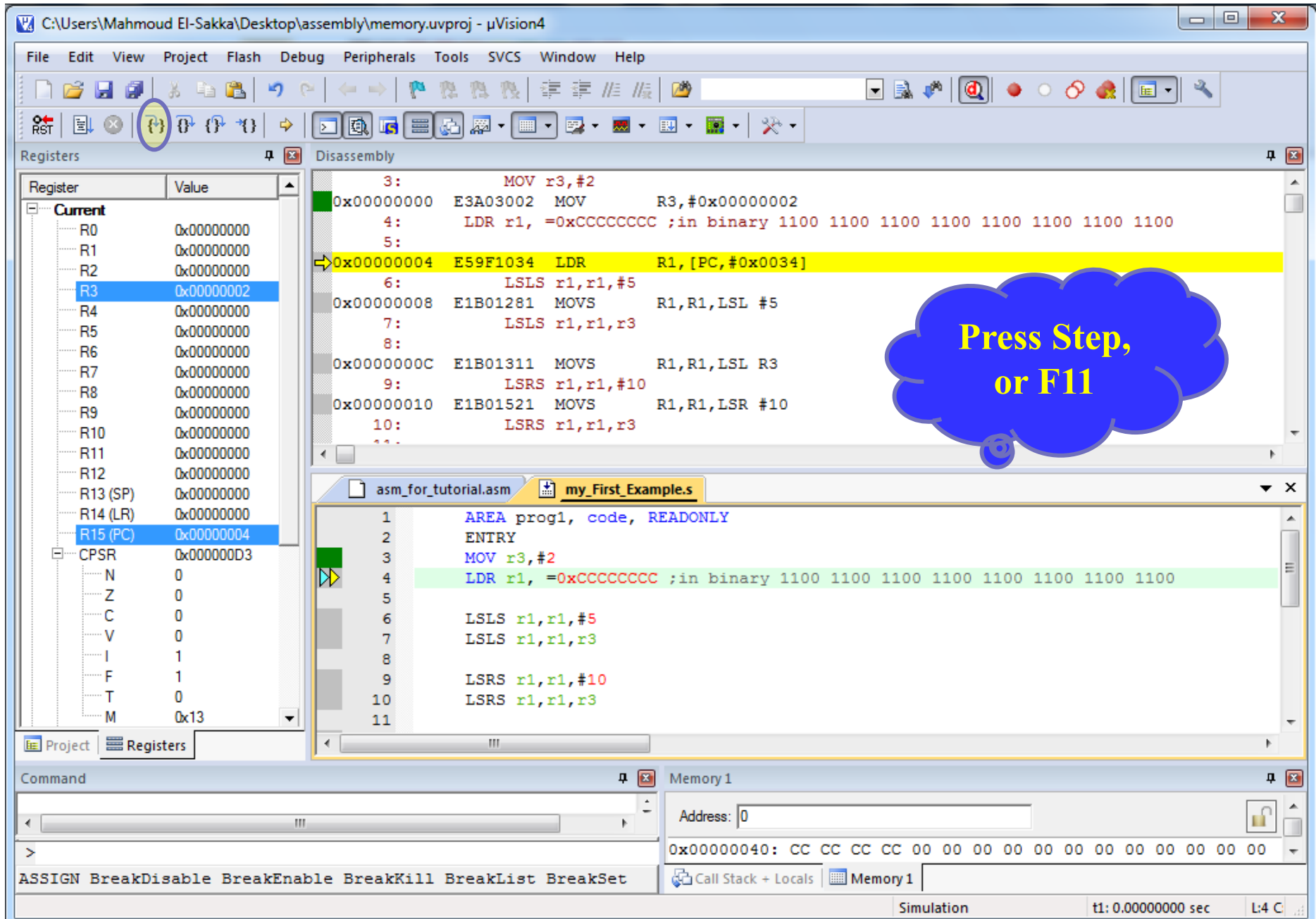
Memory Window:

Address: 0

0x00000000: E3 A0 30 02 E5 9F 10 34 E1 B0 12 81 E1 B0 13 11

Simulation t1: 0.00000000 sec L:3 C

ARM's Data-Processing Instructions (Shift Operations)



The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000004
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

3:      MOV r3,#2
0x00000000 E3A03002 MOV      R3,#0x00000002
4:      LDR r1, =0xCCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100 1100
5:
0x00000004 E59F1034 LDR      R1,[PC,#0x0034]
6:      LSLS r1,r1,#5
0x00000008 E1B01281 MOVS     R1,R1,LSL #5
7:      LSLS r1,r1,r3
8:
0x0000000C E1B01311 MOVS     R1,R1,LSL R3
9:      LSRS r1,r1,#10
0x00000010 E1B01521 MOVS     R1,R1,LSR #10
10:     LSRS r1,r1,r3
11:

```
- Source Code Window (asm_for_tutorial.asm):**

```

1  AREA prog1, code, READONLY
2  ENTRY
3  MOV r3,#2
4  LDR r1, =0xCCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100 1100
5
6  LSLS r1,r1,#5
7  LSLS r1,r1,r3
8
9  LSRS r1,r1,#10
10 LSRS r1,r1,r3
11

```
- Memory Window:**

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

A blue callout bubble with the text "Press Step, or F11" points to the 'Step' button in the toolbar.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0xCCCCCCCC
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000008
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

3:      MOV r3,#2
0x00000000 E3A03002 MOV      R3,#0x00000002
4:      LDR r1, =0xCCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100 1100
5:
0x00000004 E59F1034 LDR      R1,[PC,#0x00000004]
6:      LSLS r1,r1,#5
0x00000008 E1B01281 MOVS     R1,R1,LSL #5
7:      LSLS r1,r1,r3
8:
0x0000000C E1B01311 MOVS     R1,R1,LSL R3
9:      LSRS r1,r1,#10
0x00000010 E1B01521 MOVS     R1,R1,LSR #10
10:     LSRS r1,r1,r3

```
- Source Code Window (asm_for_tutorial.asm):**

```

3      MOV r3,#2
4      LDR r1, =0xCCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100 1100
5
6      LSLS r1,r1,#5
7      LSLS r1,r1,r3
8
9      LSRS r1,r1,#10
10     LSRS r1,r1,r3
11
12     ASRS r1,r1,#2
13     LSLS r1,r1,#15

```
- Diagram:** A diagram showing the LSL (Logical Shift Left) operation. It consists of a box labeled 'C' (Carry flag) with an arrow pointing to it from a box labeled 'Operand'. The 'Operand' box has an arrow pointing to it from a box labeled '0'. The entire diagram is enclosed in a yellow border.
- Binary Data:**
 - Initial value of R1: 1100 1100 1100 1100 1100 1100 1100 1100
 - Value after LSL #5: 1001 1001 1001 1001 1001 1001 1001 0000
- Annotation:** A blue cloud with the text "Press Step, or F11" pointing to the assembly code.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0x99999980
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000000C
CPSR	0xA00000D3
N	1
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

3:      MOV r3,#2
0x00000000 E3A03002 MOV      R3,#0x00000002
4:      LDR r1, =0xCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100
5:
0x00000004 E59F1034 LDR      R1,[PC,#0x0034]
6:      LSLS r1,r1,#5
0x00000008 E1B01281 MOVS     R1,R1,LSL #5
7:      LSLS r1,r1,r3
8:
0x0000000C E1B01311 MOVS     R1,R1,LSL R3
9:      LSRS r1,r1,#10
0x00000010 E1B01521 MOVS     R1,R1,LSR #10
10:     LSRS r1,r1,r3

```
- Source Code Window (asm_for_tutorial.asm):**

```

4      LDR r1, =0xCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100
5
6      LSLS r1,r1,#5
7      LSLS r1,r1,r3
8
9      LSRS r1,r1,#10
10     LSRS r1,r1,r3
11
12     ASRS r1,r1,#2
13     LSLS r1,r1,#15
14     ASRS r1,r1,#16

```
- Annotations:**
 - A yellow box highlights the LSL instruction: `LSL` with a diagram showing `C ← Operand ← 0`.
 - A red arrow points from the `LSLS r1,r1,#5` instruction to the register R1 value (0x99999980).
 - A green arrow points from the `LSLS r1,r1,r3` instruction to the register R1 value.
 - A blue arrow points from the `LSLS r1,r1,r3` instruction to the `C` (Carry) flag in the CPSR register.
 - Two binary representations of the value 0x99999980 are shown:
 - 1100 1100 1100 1100 1100 1100 1100 1100
 - 1001 1001 1001 1001 1001 1001 1000 0000

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0x99999980
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000000C
CPSR	0xA00000D3
N	1
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

3:      MOV r3,#2
0x00000000 E3A03002 MOV      R3,#0x00000002
4:      LDR r1, =0xCCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100
5:
0x00000004 E59F1034 LDR      R1,[PC,#0x0034]
6:      LSLS r1,r1,#5
0x00000008 E1B01281 MOVS     R1,R1,LSL #5
7:      LSLS r1,r1,r3
8:
0x0000000C E1B01311 MOVS     R1,R1,LSL R3
9:      LSRS r1,r1,#10
0x00000010 E1B01521 MOVS     R1,R1,LSR #10
10:     LSRS r1,r1,r3

```
- Source Code Window (asm_for_tutorial.asm):**

```

4      LDR r1, =0xCCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100
5
6      LSLS r1,r1,#5
7      LSLS r1,r1,r3
8
9      LSRS r1,r1,#10
10     LSRS r1,r1,r3
11
12     ASRS r1,r1,#2
13     LSLS r1,r1,#15
14     ASRS r1,r1,#16

```
- Diagram:** A diagram showing the LSL operation: $C \leftarrow \text{Operand} \leftarrow 0$.
- Binary Representation:**

Initial value of R1: 1001 1001 1001 1001 1001 1001 1000 0000

Value after LSL R3 (5 shifts): 0110 0110 0110 0110 0110 0110 0000 0000
- Instruction List:**
 - 4: LDR r1, =0xCCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100
 - 5:
 - 6: LSLS r1,r1,#5
 - 7: LSLS r1,r1,r3
 - 8:
 - 9: LSRS r1,r1,#10
 - 10: LSRS r1,r1,r3
 - 11:
 - 12: ASRS r1,r1,#2
 - 13: LSLS r1,r1,#15
 - 14: ASRS r1,r1,#16
- Command Window:**

```

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet

```
- Memory Window:**

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00
- Simulation Status:** t1: 0.00000000 sec, L:7

Press Step, or F11

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:** Shows the current state of registers. R1 is highlighted with a value of 0x66666600. The CPSR (Current Program Status Register) is also shown, with the C (Carry) flag set to 0.
- Disassembly Panel:** Shows the assembly code being executed. The instruction at address 0x00000004 is `LSLS r1, r1, #5`, which is highlighted in green. The instruction at address 0x00000008 is `LSLS r1, r1, r3`, which is highlighted in yellow.
- Source Code Panel:** Shows the assembly code being edited. The instruction at line 6 is `LSLS r1, r1, #5`, which is highlighted in green. The instruction at line 9 is `LSRS r1, r1, #10`, which is highlighted in yellow.
- Memory Panel:** Shows the memory address 0x00000040 with the value 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.

Annotations in the image include:

- A yellow box labeled "LSL" showing a diagram of the shift operation: `C ← Operand ← 0`.
- A red arrow pointing from the `LSLS r1, r1, #5` instruction in the disassembly panel to the register R1 in the registers panel.
- A green arrow pointing from the `LSRS r1, r1, #10` instruction in the source code panel to the register R1 in the registers panel.
- A blue arrow pointing from the `LSRS r1, r1, #10` instruction in the source code panel to the C flag in the CPSR register.
- Two yellow boxes showing the binary representation of the register R1 after the shift operations:
 - Top box: `1001 1001 1001 1001 1001 1001 1001 1000 0000`
 - Bottom box: `0110 0110 0110 0110 0110 0110 0110 0000 0000`

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0x66666600
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000010
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

3:      MOV r3,#2
0x00000000 E3A03002 MOV      R3,#0x00000002
4:      LDR r1, =0xCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100 1100
5:
0x00000004 E59F1034 LDR      R1,[PC,#0x0034]
6:      LSLS r1,r1,#5
0x00000008 E1B01281 MOVS     R1,R1,LSL #5
7:      LSLS r1,r1,r3
8:
0x0000000C E1B01311 MOVS     R1,R1,LSL R3
9:      LSRS r1,r1,#10
0x00000010 E1B01521 MOVS     R1,R1,LSR #10
10:     LSRS r1,r1,r3

```
- Source Code Window (asm_for_tutorial.asm):**

```

6      LSLS r1,r1,#5
7      LSLS r1,r1,r3
8
9      LSRS r1,r1,#10
10     LSRS r1,r1,r3
11
12     ASRS r1,r1,#2
13     LSLS r1,r1,#15
14     ASRS r1,r1,#16
15
16     ASRS r1,r1,r3

```
- Diagram:** A box labeled "LSR" shows the operation: 0 → Operand → C. The "C" (Carry) register is highlighted in blue.
- Bit Patterns:**
 - Initial state (highlighted in yellow): 0110 0110 0110 0110 0110 0110 0000 0000
 - Result after LSR #10 (highlighted in green): 0000 0000 0001 1001 1001 1001 1001 1001
- Callout:** A blue cloud-shaped box with a play button icon contains the text: "Press Step, or F11".
- Memory Window:** Shows address 0x00000040 with data: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00.
- Simulation Status:** Simulation, t1: 0.00000000 sec, L:9 C.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the uVision4 IDE interface with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0x00199999
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000014
CPSR	0x200000D3
N	0
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

0x0000000C E1B01311 MOVS R1,R1,LSL R3
9:          LSRS r1,r1,#10
0x00000010 E1B01521 MOVS R1,R1,LSR #10
10:         LSRS r1,r1,r3
11:
0x00000014 E1B01331 MOVS R1,R1,LSR R3
12:         ASRS r1,r1,#2
0x00000018 E1B01141 MOVS R1,R1,ASR #2
13:         LSLS r1,r1,#15
0x0000001C E1B01781 MOVS R1,R1,LSL #15
14:         ASRS r1,r1,#16
15:
0x00000020 E1B01841 MOVS R1,R1,ASR #16
16:         ASRS r1,r1,r3

```
- asm_for_tutorial.asm Window:**

```

6  LSLS r1,r1,#5
7  LSLS r1,r1,r3
8
9  LSRS r1,r1,#10
10 LSRS r1,r1,r3
11
12 ASRS r1,r1,#2
13 LSLS r1,r1,#15
14 ASRS r1,r1,#16
15
16 ASRS r1,r1,r3

```
- Memory Window:**

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00
- Call Stack + Locals Window:**

Memory 1
- Simulation Status:**

Simulation t1: 0.00000000 sec L:10

Annotations in the image include a red arrow pointing from the CPSR register to the 'C' flag in the Disassembly window, a blue arrow pointing from the 'C' flag to the 'C' flag in the Registers window, and a green arrow pointing from the 'C' flag to the 'C' flag in the Disassembly window. A yellow box highlights the value 0000 0000 0001 1001 1001 1001 1001 1001 in the Disassembly window, and a red circle with the number 1 is next to it.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0x00199999
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000014
CPSR	0x200000D3
N	0
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

0x0000000C E1B01311 MOVS    R1,R1,LSL R3
9:          LSRS    r1,r1,#10
0x00000010 E1B01521 MOVS    R1,R1,LSR #10
10:         LSRS    r1,r1,r3
11:
0x00000014 E1B01331 MOVS    R1,R1,LSR R3
12:         ASRS    r1,r1,#2
0x00000018 E1B01141 MOVS    R1,R1,ASR #2
13:         LSLS    r1,r1,#15
0x0000001C E1B01781 MOVS    R1,R1,LSL #15
14:         ASRS    r1,r1,#16
15:
0x00000020 E1B01841 MOVS    R1,R1,ASR #16

```
- Source Code Window (asm_for_tutorial.asm):**

```

6      LSLS    r1,r1,#5
7      LSLS    r1,r1,r3
8
9      LSRS    r1,r1,#10
10     LSRS    r1,r1,r3
11
12     ASRS    r1,r1,#2
13     LSLS    r1,r1,#15
14     ASRS    r1,r1,#16
15
16     ASRS    r1,r1,r3

```
- Diagram:**

```

graph LR
    0 --> LSR[LSR]
    LSR --> Operand[Operand]
    Operand --> C[C]

```
- Binary Sequences:**

```

0000 0000 0001 1001 1001 1001 1001 1001
0000 0000 0000 0110 0110 0110 0110 0110

```
- Call to Action:** Press Step, or F11

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the Keil uVision4 IDE with the following components:

- Registers Window:**
 - R0: 0x00000000
 - R1: 0x00066666
 - R2: 0x00000000
 - R3: 0x00000002
 - R4: 0x00000000
 - R5: 0x00000000
 - R6: 0x00000000
 - R7: 0x00000000
 - R8: 0x00000000
 - R9: 0x00000000
 - R10: 0x00000000
 - R11: 0x00000000
 - R12: 0x00000000
 - R13 (SP): 0x00000000
 - R14 (LR): 0x00000000
 - R15 (PC): 0x00000018
 - CPSR: 0x000000D3
 - N: 0
 - Z: 0
 - C: 0
 - V: 0
 - I: 1
 - F: 1
 - T: 0
 - M: 0x13
- Disassembly Window:**
 - 0x0000000C E1B01311 MOVs R1,R1,LSL R3
 - 9: LSRs r1,r1,#10
 - 0x00000010 E1B01521 MOVs R1,R1,LSR #10
 - 10: LSRs r1,r1,r3
 - 11:
 - 0x00000014 E1B01331 MOVs R1,R1,LSR R3
 - 12: ASRS r1,r1,#2
 - 0x00000018 E1B01141 MOVs R1,R1,ASR #2
 - 13: LSLs r1,r1,#15
 - 0x0000001C E1B01781 MOVs R1,R1,LSL #15
 - 14: ASRS r1,r1,#16
 - 15:
 - 0x00000020 E1B01841 MOVs R1,R1,ASR #16
 - 16: LSRs r1,r1,r3
- Assembly Window (asm_for_tutorial.asm):**
 - 9 LSRs r1,r1,#10
 - 10 LSRs r1,r1,r3
 - 11:
 - 12 ASRS r1,r1,#2
 - 13 LSLs r1,r1,#15
 - 14 ASRS r1,r1,#16
 - 15:
 - 16 ASRS r1,r1,r3
 - 17:
 - 18 RORS r1,r1,#4
 - 19 RORS r1,r1,r3
- Memory Window:**
 - Address: 0
 - 0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:** Shows the current state of registers. R15 (PC) is at 0x00000018. The CPSR register shows the Carry flag (C) is 0.
- Disassembly Panel:** Shows the assembly code being executed. The instruction at address 0x00000018 is `ASRS r1,r1,#2`, which is highlighted in yellow.
- Assembly Source Panel:** Shows the source code for `my_First_Example.s`. The instruction `ASRS r1,r1,#2` is highlighted in green.
- Diagram:** A diagram of the ASR instruction is shown in the top right. It illustrates the shift operation: the MSB (Most Significant Bit) of the operand is shifted into the Carry flag (C).
- Bit Patterns:** Two bit patterns are shown, representing the operand and the result of the shift operation. The first pattern is `0000 0000 0000 0110 0110 0110 0110 0110`, and the second pattern is `0000 0000 0000 0001 1001 1001 1001 1001`. Arrows indicate the shift of the MSB into the Carry flag.
- Call to Action:** A blue cloud bubble in the bottom right corner contains the text "Press Step, or F11", indicating the next step in the simulation.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0x00019999
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000001C
CPSR	0x200000D3
N	0
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

0x0000000C E1B01311 MOVS R1,R1,LSL R3
9:          LSRS r1,r1,#10
0x00000010 E1B01521 MOVS R1,R1,LSR #10
10:         LSRS r1,r1,r3
11:
0x00000014 E1B01331 MOVS R1,R1,LSR R3
12:         ASRS r1,r1,#2
0x00000018 E1B01141 MOVS R1,R1,ASR #2
13:         LSLS r1,r1,#15
0x0000001C E1B01781 MOVS R1,R1,LSL #15
14:         ASRS r1,r1,#16
15:
0x00000020 E1B01841 MOVS R1,R1,ASR #16
16:         ASRS r1,r1,r3

```
- Source Code Window (asm_for_tutorial.asm):**

```

10      LSRS r1,r1,r3
11
12      ASRS r1,r1,#2
13      LSLS r1,r1,#15
14      ASRS r1,r1,#16
15
16      ASRS r1,r1,r3
17
18      RORS r1,r1,#4
19      RORS r1,r1,r3
20

```
- Memory Window:**

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

Annotations in the image include:

- A red arrow pointing from the `LSRS r1,r1,r3` instruction to the value `0x00019999` in register R1.
- A blue arrow pointing from the `LSLS r1,r1,#15` instruction to the bit field `0000 0000 0000 0001 1001 1001 1001 1001` in the memory window.
- A green arrow pointing from the `ASRS r1,r1,#16` instruction to the same bit field in the memory window.
- A yellow circle with the number '1' next to the bit field.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:**

Register	Value
R0	0x00000000
R1	0x00019999
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000001C
CPSR	0x200000D3
N	0
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Panel:**

```

0x0000000C E1B01311 MOVS      R1,R1,LSL R3
9:          LSRS   r1,r1,#10
0x00000010 E1B01521 MOVS      R1,R1,LSR #10
10:         LSRS   r1,r1,r3
11:
0x00000014 E1B01331 MOVS      R1,R1,LSR R3
12:         ASRS   r1,r1,#2
0x00000018 E1B01141 MOVS      R1,R1,ASR #2
13:         LSLS   r1,r1,#15
0x0000001C E1B01781 MOVS      R1,R1,LSL #15
14:         ASRS   r1,r1,#16
15:
0x00000020 E1B01841 MOVS      R1,R1,ASR #16

```
- Source Code Panel (asm_for_tutorial.asm):**

```

10      LSRS   r1,r1,r3
11
12      ASRS   r1,r1,#2
13      LSLS   r1,r1,#15
14      ASRS   r1,r1,#16
15
16      ASRS   r1,r1,r3
17
18      RORS   r1,r1,#4
19      RORS   r1,r1,r3
20

```
- Diagram:** A diagram showing the LSL operation: $C \leftarrow \text{Operand} \leftarrow 0$. The 'C' (Carry) flag is highlighted in blue.
- Bit Pattern:** A 32-bit binary representation of the value in R1 after the LSL instruction:

0000	0000	0000	0001	1001	1001	1001	1001
1100	1100	1100	1100	1000	0000	0000	0000
- Callout:** A blue cloud-shaped box with the text "Press Step, or F11".

ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the uVision4 IDE interface with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0xCCCC8000
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000020
CPSR	0x800000D3
N	1
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

13:      LSLS r1,r1,#15
0x0000001C E1B01781 MOVS      R1,R1,LSL #15
14:      ASRS r1,r1,#16
15:      ASRS r1,r1,r3
0x00000020 E1B01841 MOVS      R1,R1,ASR #16
16:      ASRS r1,r1,r3
17:
0x00000024 E1B01351 MOVS      R1,R1,ASR R3
18:      RORS r1,r1,#4
0x00000028 E1B01261 MOVS      R1,R1,ROR #4
19:      RORS r1,r1,r3
20:
0x0000002C E1B01371 MOVS      R1,R1,ROR R3

```
- Source Code Window (asm_for_tutorial.asm):**

```

11
12      ASRS r1,r1,#2
13      LSLS r1,r1,#15
14      ASRS r1,r1,#16
15
16      ASRS r1,r1,r3
17
18      RORS r1,r1,#4
19      RORS r1,r1,r3
20
21      RRSX r1,r1

```
- Memory Window:**

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

A binary representation of the R1 register value (0xCCCC8000) is shown as: 1100 1100 1100 1100 1000 0000 0000 0000. A red circle highlights the bit at position 16 (the 17th bit from the left), which is 0. A red arrow points from this bit to the 'C' (Carry) flag in the CPSR register. A green arrow points from the 'N' (Negative) flag in the CPSR register to the instruction at line 14 (ASRS r1,r1,#16).

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0xCCCC8000
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000020
CPSR	0x800000D3
N	1
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

13:      LSLS r1,r1,#15
0x0000001C E1B01781 MOVS    R1,R1,LSL #15
14:      ASRS r1,r1,#16
15:      MOVS r1,r1,ASR #16
16:      ASRS r1,r1,r3
17:      MOVS r1,r1,ASR R3
18:      RORS r1,r1,#4
19:      RORS r1,r1,r3
20:      MOVS r1,r1,ROR R3

```
- Source Code Window (asm_for_tutorial.asm):**

```

11
12      ASRS r1,r1,#2
13      LSLS r1,r1,#15
14      ASRS r1,r1,#16
15
16      ASRS r1,r1,r3
17
18      RORS r1,r1,#4
19      RORS r1,r1,r3
20
21      RRXS r1,r1

```
- ASR Diagram:**

```

graph LR
    MSB[MSB] --> C[C]
    Operand[Operand] --> C
    style MSB fill:none,stroke:none
    style C fill:none,stroke:none

```
- Bit Patterns:**

Initial value (R1): 1100 1100 1100 1100 1000 0000 0000 0000

Result after ASRS r1, r1, #16: 1111 1111 1111 1111 1100 1100 1100 1100
- Callout:** Press Step, or F11

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0xFFFFCCCC
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000024
CPSR	0xA00000D3
N	1
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

13:      LSLS r1,r1,#15
0x0000001C E1B01781 MOVS      R1,R1,LSL #15
14:      ASRS r1,r1,#16
15:
0x00000020 E1B01841 MOVS      R1,R1,ASR #16
16:      ASRS r1,r1,r3
17:
0x00000024 E1B01351 MOVS      R1,R1,ASR R3
18:      RORS r1,r1,#4
0x00000028 E1B01261 MOVS      R1,R1,ROR #4
19:      RORS r1,r1,r3
20:
0x0000002C E1B01371 MOVS      R1,R1,ROR R3

```
- Source Code Window (asm_for_tutorial.asm):**

```

13      LSLS r1,r1,#15
14      ASRS r1,r1,#16
15
16      ASRS r1,r1,r3
17
18      RORS r1,r1,#4
19      RORS r1,r1,r3
20
21      RRXS r1,r1
22      RRXS r1,r1
23      RRXS r1,r1

```
- Callout Box:** 1111 1111 1111 1111 1100 1100 1100 1100
- Registers Window:** R1 is highlighted with a red arrow pointing to the callout box. R15 (PC) and CPSR are also highlighted.
- Source Code Window:** Line 16 (ASRS r1,r1,r3) is highlighted with a green arrow pointing to the callout box.
- Registers Window:** The C flag is highlighted with a green arrow pointing to the callout box.
- Memory Window:** Address 0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

1

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0xFFFFCCCC
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000024
CPSR	0xA00000D3
N	1
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

13:      LSLS r1,r1,#15
0x0000001C E1B01781 MOVS    R1,R1,LSL #15
14:      ASRS r1,r1,#16
15:
0x00000020 E1B01841 MOVS    R1,R1,ASR #16
16:      ASRS r1,r1,r3
17:
0x00000024 E1B01351 MOVS    R1,R1,ASR R3
18:      RORS r1,r1,#4
0x00000028 E1B01261 MOVS    R1,R1,ROR #4
19:      RORS r1,r1,r3
20:
0x0000002C E1B01371 MOVS    R1,R1,ROR R3

```
- ASR Diagram:**

```

graph LR
    MSB[MSB] --> C[C]
    Operand[Operand] --> C
    ASR[ASR] --- MSB
    ASR --- Operand
    
```
- Bit Patterns:**

```

1111 1111 1111 1111 1100 1100 1100 1100
1111 1111 1111 1111 1111 0011 0011 0011

```
- Assembly Code Window:**

```

13 LSLS r1,r1,#15
14 ASRS r1,r1,#16
15
16 ASRS r1,r1,r3
17
18 RORS r1,r1,#4
19 RORS r1,r1,r3
20
21 RRXS r1,r1
22 RRXS r1,r1
23 RRXS r1,r1

```
- Command Window:**

```

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet

```
- Memory Window:**

```

Address: 0
0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

```

A blue cloud bubble with the text "Press Step, or F11" is overlaid on the assembly code window.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the uVision4 IDE interface. The **Registers** window on the left shows the current state of ARM registers. Register **R1** is highlighted with a value of **0xFFFF333**. The **Disassembly** window shows the following instructions:

```

13:      LSLS r1,r1,#15
0x0000001C E1B01781 MOVS      R1,R1,LSL #15
14:      ASRS r1,r1,#16
15:
0x00000020 E1B01841 MOVS      R1,R1,ASR #16
16:      ASRS r1,r1,r3
17:
0x00000024 E1B01351 MOVS      R1,R1,ASR R3
18:      RORS r1,r1,#4
0x00000028 E1B01261 MOVS      R1,R1,ROR #4
19:      RORS r1,r1,r3
20:
0x0000002C E1B01371 MOVS      R1,R1,ROR R3

```

The instruction **R1,R1,ROR #4** is highlighted in yellow, and its binary representation is shown as **1111 1111 1111 1111 1111 0011 0011 0011**. The **asm_for_tutorial.asm** window shows the source code:

```

15
16      ASRS r1,r1,r3
17
18      RORS r1,r1,#4
19      RORS r1,r1,r3
20
21      RRXS r1,r1
22      RRXS r1,r1
23      RRXS r1,r1
24      RRXS r1,r1
25      END

```

The **Command** window at the bottom shows the command **ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet**. The **Memory** window shows the address **0x00000040** with the value **CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00**.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:** Shows the current state of registers. R15 (PC) is at 0x00000028. CPSR is 0x800000D3. The Carry flag (C) is 0.
- Disassembly Window:** Shows assembly instructions. Instruction 18 is highlighted: `RORS r1, r1, #4`. Instruction 19 is `RORS r1, r1, r3`.
- Source Code Window:** Shows the assembly code for `my_First_Example.s`. Instruction 18 is highlighted: `RORS r1, r1, #4`.
- Diagram:** A diagram of the ROR instruction. It shows an 'Operand' box with an arrow pointing to a 'C' (Carry) box. The diagram is labeled 'ROR'.
- Bit Patterns:** Two bit patterns are shown, representing the state of the register and the carry flag after the ROR instruction. The first pattern is `1111 1111 1111 1111 1111 0011 0011 0011`. The second pattern is `0011 1111 1111 1111 1111 1111 0011 0011`.
- Call to Action:** A blue cloud-shaped bubble with the text "Press Step, or F11" is overlaid on the source code window.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the uVision4 IDE with the following components:

- Registers Window:** Shows the current state of registers. R1 contains 0x3FFFFFF3. The Program Counter (PC) is at 0x0000002C.
- Disassembly Window:** Shows the assembly code being executed. The current instruction is `MOVN R1, R1, ROR, R3` at address 0x0000002C. The instruction is highlighted in yellow.
- Source File Window:** Shows the assembly source code `asm_for_tutorial.asm`. The current instruction is `RORS r1, r1, #4` at line 19.
- Binary Representation:** A binary representation of the instruction is shown as `0011 1111 1111 1111 1111 1111 0011 0011`. A red arrow points from the `RORS r1, r1, #4` instruction to this binary representation.
- Command Window:** Shows the command `ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet`.
- Memory Window:** Shows the memory address 0x00000040 with the value `CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00`.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0x3FFFFFF3
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000002C
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

0x00000028 E1B01261 MOVN    R1,R1,ROR #4
19:         RORS    r1,r1,r3
20:
0x0000002C E1B01371 MOVN    R1,R1,ROR R3
21:         RRXS    r1,r1
0x00000030 E1B01061 MOVN    R1,R1,RRX
22:         RRXS    r1,r1
0x00000034 E1B01061 MOVN    R1,R1,RRX
23:         RRXS    r1,r1
0x00000038 E1B01061 MOVN    R1,R1,RRX
24:         RRXS    r1,r1
0x0000003C E1B01061 MOVN    R1,R1,RRX
0x00000040 CCCCCCCC STCGTL   p12,CR12,[R12],{204}
0x00000044 CCCCCCCC UNDEF
  
```
- Source Code Window (asm_for_tutorial.asm):**

```

15
16     ASRS    r1,r1,r3
17
18     RORS    r1,r1,#4
19     RORS    r1,r1,r3
20
21     RRXS    r1,r1
22     RRXS    r1,r1
23     RRXS    r1,r1
24     RRXS    r1,r1
25     END
  
```
- Diagram:** A box labeled "ROR" contains a diagram showing an "Operand" being rotated right into a register "C".
- Bit Patterns:**
 - Initial state: 0011 1111 1111 1111 1111 1111 0011 0011
 - After RORS r1, r1, r3: 1100 1111 1111 1111 1111 1111 1100 1100
- Annotation:** A blue cloud with the text "Press Step, or F11" is overlaid on the source code window.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers:** A list of registers (R0-R15, CPSR) with their current values. R1 is highlighted with a value of 0xCFFFFFFC.
- Disassembly:** A list of assembly instructions. The instruction `RRXS r1, r1` is highlighted in yellow. A red arrow points from this instruction to the R1 register. A green arrow points from this instruction to the 'C' flag in the CPSR register.
- asm_for_tutorial.asm:** A file containing assembly code. The instruction `RRXS r1, r1` is highlighted in green. A red arrow points from this instruction to the R1 register. A green arrow points from this instruction to the 'C' flag in the CPSR register.
- Memory 1:** A window showing memory addresses and values. The address 0x00000040 is highlighted with a value of 00 00 00 00 00 00 00 00.

The highlighted instruction `RRXS r1, r1` is shown in the Disassembly window with the following details:

- Address: 0x00000030
- Instruction: `RRXS r1, r1`
- Value: 1100 1111 1111 1111 1111 1111 1100 1100

The CPSR register is shown with the following flags:

- N: 1
- Z: 0
- C: 1
- V: 0
- I: 1
- F: 1
- T: 0
- M: 0x13

ARM's Data-Processing Instructions (Shift Operations)

Rotate right through carry

```

    graph LR
      Register[Register] --> Carry[Carry]
      Carry --> Register
  
```

Registers

Register	Value
R0	0x00000000
R1	0xCFFFFFFC
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000030
CPSR	0xA00000D3
N	1
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13

Disassembly

```

0x00000028 E1B01261 MOVS R1,R1,ROR #4
19: RORS r1,r1,r3
20:
0x0000002C E1B01371 MOVS R1,R1,ROR R3
21: RRXS r1,r1
0x00000030 E1B01061 MOVS R1,R1,RRX
22: RRXS r1,r1
0x00000034 E1B01061 MOVS R1,R1,RRX
23: RRXS r1,r1
0x00000038 E1B01061 MOVS R1,R1,RRX
24: RRXS r1,r1
0x0000003C E1B01061 MOVS R1,R1,RRX
0x00000040 CCCCCCCC STCGTL p12,CR12,[R12],{204}
0x00000044 CCCCCCCC ANDEC R0,R0,R0
  
```

asm_for_tutorial.asm

```

15
16 ASRS r1,r1,r3
17
18 RORS r1,r1,#4
19 RORS r1,r1,r3
20
21 RRXS r1,r1
22 RRXS r1,r1
23 RRXS r1,r1
24 RRXS r1,r1
25 END
  
```

my_First_Example.s

```

15
16 ASRS r1,r1,r3
17
18 RORS r1,r1,#4
19 RORS r1,r1,r3
20
21 RRXS r1,r1
22 RRXS r1,r1
23 RRXS r1,r1
24 RRXS r1,r1
25 END
  
```

Memory 1

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

Simulation t1: 0.00000000 sec L:21

Press Step, or F11

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers:** A list of registers (R0-R15, CPSR) with their current values. R1 is highlighted with a red arrow pointing to its value 0xE7FFFE6.
- Disassembly:** A list of assembly instructions. The instruction `RRXS r1, r1` at address 0x00000034 is highlighted in yellow. A yellow box contains the binary sequence `1110 0111 1111 1111 1111 1111 1111 1110 0110`. A green arrow points from this box to the `C` flag in the assembly code.
- asm_for_tutorial.asm:** A list of assembly instructions. The instruction `RRXS r1, r1` at address 22 is highlighted in green. A blue arrow points from the `C` flag in the assembly code to the `C` flag in the CPSR register.
- CPSR:** A list of flags (N, Z, C, V, I, F, T, M) with their current values. The `C` flag is highlighted with a green arrow pointing to it.
- Memory 1:** A window showing memory addresses and values. The address 0x00000040 is shown with a value of `CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00`.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:** Shows the current state of registers. R15 (PC) is at 0x00000034. CPSR is 0x800000D3. The 'C' (Carry) flag is 0.
- Disassembly Window:** Shows assembly code. Line 22 is highlighted: `RRXS r1, r1`. The instruction is `0x00000034 E1B01061 MOVNS R1, R1, RRX`.
- Diagram:** A red box labeled "Rotate right through carry" shows a diagram of a register and a carry flag. The register is a blue box labeled "Register" with an arrow pointing to a smaller blue box labeled "Carry".
- Bit Patterns:** Two rows of bit patterns are shown, representing the register and carry state before and after the shift.

1110 0111 1111 1111 1111 1111 1111 1110	0110
0111 0011 1111 1111 1111 1111 1111 1111	0011
- Code Window:** Shows assembly code for `my_First_Example.s`. Line 22 is highlighted: `RRXS r1, r1`.
- Command Window:** Shows the command `ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet`.
- Memory Window:** Shows memory address 0x00000040 with value `CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00`.
- Simulation Status:** Shows `Simulation` with `t1: 0.00000000 sec` and `L:22`.

A blue cloud with the text "Press Step, or F11" is overlaid on the code window, indicating the next step in the simulation.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the uVision4 IDE interface with the following components:

- Registers Window:** Shows the current state of registers. R1 is highlighted with the value 0x73FFFFFF. R15 (PC) is at 0x00000038.
- Disassembly Window:** Shows the disassembled instructions. The instruction at address 0x00000038 is `MOV R1, R1, RRX`, which is highlighted in yellow. The instruction at address 0x00000039 is `RRXS r1, r1`.
- Source Code Window:** Shows the assembly code for `my_First_Example.s`. The instructions are:


```

15
16      ASRS r1, r1, r3
17
18      RORS r1, r1, #4
19      RORS r1, r1, r3
20
21      RRXS r1, r1
22      RRXS r1, r1
23      RRXS r1, r1
24      RRXS r1, r1
25      END
      
```
- Binary Representation:** A yellow box highlights the binary value of R1: `0111 0011 1111 1111 1111 1111 1111 0011`. A red arrow points from the R1 register value to this binary representation.
- Annotations:** A green arrow points from the `RRXS r1, r1` instruction in the source code to the `RRXS r1, r1` instruction in the disassembly window. A blue arrow points from the `RRXS r1, r1` instruction in the disassembly window to the `RRXS r1, r1` instruction in the source code. A green arrow points from the `RRXS r1, r1` instruction in the source code to the `RRXS r1, r1` instruction in the disassembly window.

ARM's Data-Processing Instructions (Shift Operations)

Rotate right through carry

```

    graph LR
      Register[Register] --> Carry[Carry]
      Carry --> Register
  
```

Registers

Register	Value
R0	0x00000000
R1	0x73FFFFFF
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000038
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13

Disassembly

```

0x00000028 E1B01261 MOVS R1,R1,ROR #4
19:         RORS r1,r1,r3
20:
0x0000002C E1B01371 MOVS R1,R1,ROR R3
21:         RRXS r1,r1
22:         RRXS r1,r1
23:         RRXS r1,r1
24:         RRXS r1,r1
0x00000038 E1B01061 MOVS R1,R1,RRX
25:         RRXS r1,r1
0x0000003C E1B01061 MOVS R1,R1,RRX
0x00000040 CCCCCCCC STCGTL p12,CR12,[R12],#204
0x00000044 CCCCCCCC STCGTL p12,CR12,[R12],#204
  
```

asm_for_tutorial.asm

```

15
16     ASRS r1,r1,r3
17
18     RORS r1,r1,#4
19     RORS r1,r1,r3
20
21     RRXS r1,r1
22     RRXS r1,r1
23     RRXS r1,r1
24     RRXS r1,r1
25     END
  
```

my_First_Example.s

```

15
16     ASRS r1,r1,r3
17
18     RORS r1,r1,#4
19     RORS r1,r1,r3
20
21     RRXS r1,r1
22     RRXS r1,r1
23     RRXS r1,r1
24     RRXS r1,r1
25     END
  
```

Memory 1

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

Simulation t1: 0.00000000 sec L:23

Press Step, or F11

ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the uVision4 IDE interface with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0x39FFFFFF
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000003C
CPSR	0x200000D3
N	0
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

0x00000028 E1B01261 MOVS      R1,R1,ROR #4
19:          RORS      r1,r1,r3
20:
0x0000002C E1B01371 MOVS      R1,R1,ROR R3
21:          RRXS      r1,r1
0x00000030 E1B01061 MOVS      R1,R1,RRX
22:          RRXS      r1,r1
0x00000034 E1B01061 MOVS      R1,R1,RRX
23:          RRXS      r1,r1
0x00000038 E1B01061 MOVS      R1,R1,RRX
24:          RRXS      r1,r1
0x0000003C E1B01061 MOVS      R1,R1,RRX
0x00000040 CCCCCCCC STCGTIL  p12,CR12,[R12] {204}
0x00000044 CCCCCCCC UNDEC
  
```
- Source Code Window (asm_for_tutorial.asm):**

```

15
16      ASRS      r1,r1,r3
17
18      RORS      r1,r1,#4
19      RORS      r1,r1,r3
20
21      RRXS      r1,r1
22      RRXS      r1,r1
23      RRXS      r1,r1
24      RRXS      r1,r1
25      END
  
```
- Memory Window:**

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

Annotations in the image include:

- A red arrow pointing from R1 (0x39FFFFFF) to the MOV instruction at address 0x0000002C.
- A green arrow pointing from the CPSR C flag (1) to the RORS instruction at address 0x00000018.
- A yellow box highlighting the binary value 0011 1001 1111 1111 1111 1111 1111 1001, which is the value of R1.
- A blue box highlighting the instruction ASRS r1, r1, r3 at address 0x00000016.
- A green circle with the number 1 next to the CPSR C flag.

ARM's Data-Processing Instructions (Shift Operations)

Rotate right through carry

```

    graph LR
      Register[Register] --> Carry[Carry]
      Carry --> Register
  
```

Registers

Register	Value
R0	0x00000000
R1	0x39FFFFFF
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000003C
CPSR	0x200000D3
N	0
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13

Disassembly

```

0x00000028 E1B01261 MOVS R1,R1,ROR #4
19:      RORS r1,r1,r3
20:
0x0000002C E1B01371 MOVS R1,R1,ROR R3
21:      RRXS r1,r1
22:      RRXS r1,r1
0x00000030 E1B01061 MOVS R1,R1,RRX
23:      RRXS r1,r1
0x00000034 E1B01061 MOVS R1,R1,RRX
24:      RRXS r1,r1
0x00000038 E1B01061 MOVS R1,R1,RRX
0x0000003C E1B01061 MOVS R1,R1,RRX
0x00000040 CCCCCCCC STCGTL p12,CR12,[R12],{204}
0x00000044 CCCCCCCC UNDEC R0,R0,R0
  
```

asm_for_tutorial.asm

```

15
16      ASRS r1,r1,r3
17
18      RORS r1,r1,#4
19      RORS r1,r1,r3
20
21      RRXS r1,r1
22      RRXS r1,r1
23      RRXS r1,r1
24      RRXS r1,r1
25      END
  
```

my_First_Example.s

```

15
16      ASRS r1,r1,r3
17
18      RORS r1,r1,#4
19      RORS r1,r1,r3
20
21      RRXS r1,r1
22      RRXS r1,r1
23      RRXS r1,r1
24      RRXS r1,r1
25      END
  
```

Memory 1

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

Simulation t1: 0.00000000 sec L:24

Press Step, or F11

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0x9CFFFFFF
R2	0x00000000
R3	0x00000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000044
CPSR	0xA00000D3
N	1
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
- Disassembly Window:**

```

19:      RORS r1,r1,r3
20:
0x0000002C E1B01371 MOVS      R1,R1,ROR R3
21:      RRXS r1,r1
0x00000030 E1B01061 MOVS      R1,R1,RRX
22:      RRXS r1,r1
0x00000034 E1B01061 MOVS      R1,R1,RRX
23:      RRXS r1,r1
0x00000038 E1B01061 MOVS      R1,R1,RRX
24:      RRXS r1,r1
0x0000003C E1B01061 MOVS      R1,R1,RRX
0x00000040 CCCCCTCC STCCTL    p12,CR12,[R12],{204}
0x00000044 00000000 ANDEQ    R0,R0,R0

```
- Memory Window (asm_for_tutorial.asm):**

```

15
16      ASRS r1,r1,r3
17
18      RORS r1,r1,#4
19      RORS r1,r1,r3
20
21      RRXS r1,r1
22      RRXS r1,r1
23      RRXS r1,r1
24      RRXS r1,r1
25      END

```
- Memory 1 Window:**

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00
- Command Window:**

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
- Simulation Status:**

Step one line | Simulation | t1: 0.00000000 sec | L:24

A red arrow points from register R1 (0x9CFFFFFF) to the assembly code. A green arrow points from the assembly code to the memory window. A yellow box highlights the binary value 1001 1100 1111 1111 1111 1111 1111 1100.

1

ARM's Data-Processing Instructions (Shift Operations)

```
AREA prog1, code, READONLY
```

```
ENTRY
```

```
MOV r3, #2
```

```
LDR r1, =0xCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100 1100
```

```
LSL r1, r1, #5
```

```
LSL r1, r1, r3
```

```
LSR r1, r1, #10
```

```
LSR r1, r1, r3
```

```
ASR r1, r1, #2
```

```
LSL r1, r1, #15
```

```
ASR r1, r1, #16
```

```
ASR r1, r1, r3
```

```
ROR r1, r1, #4
```

```
ROR r1, r1, r3
```

```
RRX r1, r1
```

```
RRX r1, r1
```

```
RRX r1, r1
```

```
RRX r1, r1
```

```
END
```

Repeat the example again without the “S”

