

These slides are being provided with permission from the copyright for in-class (CS2208B) 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

Winter 2020-2021

Instructor: Mahmoud R. El-Sakka

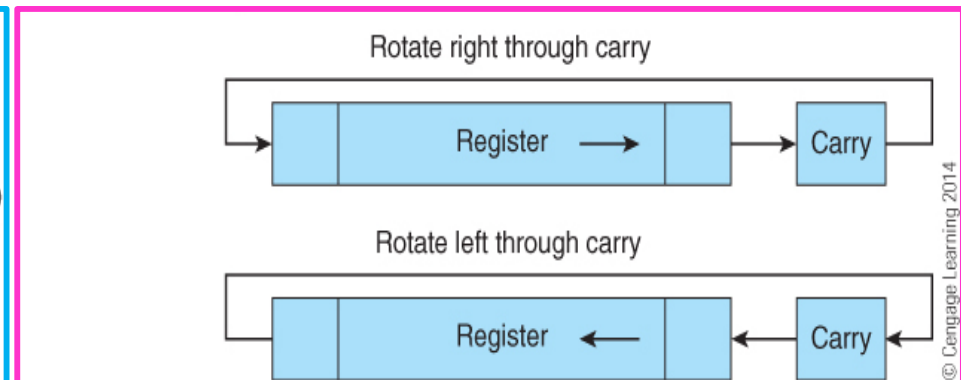
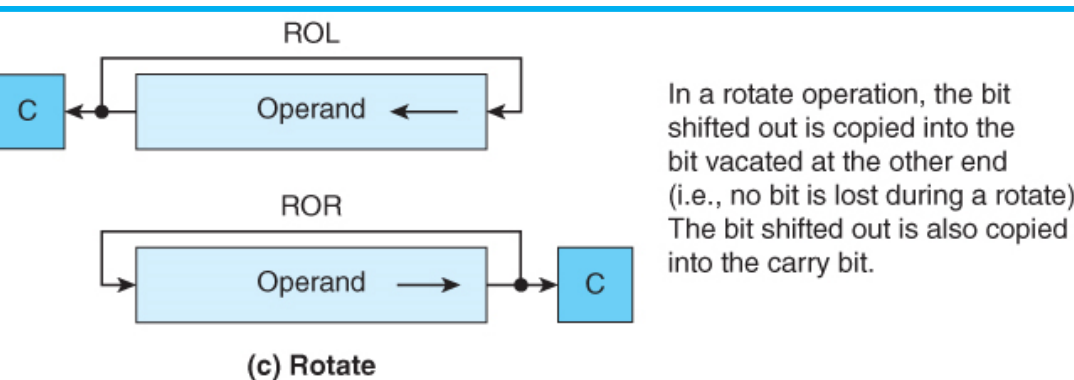
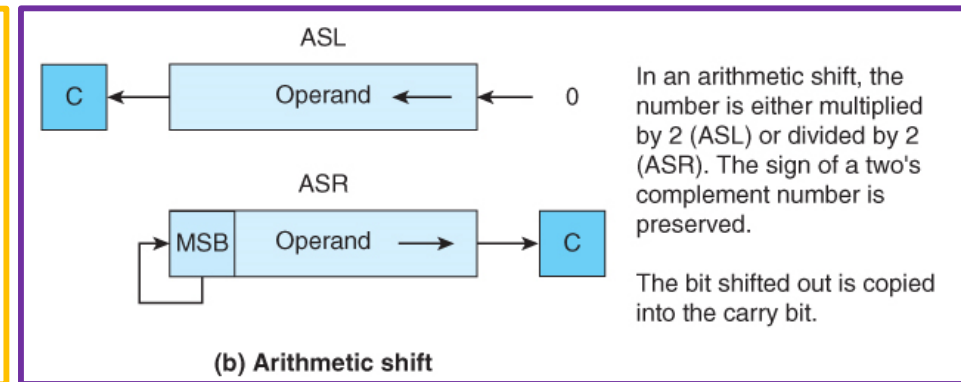
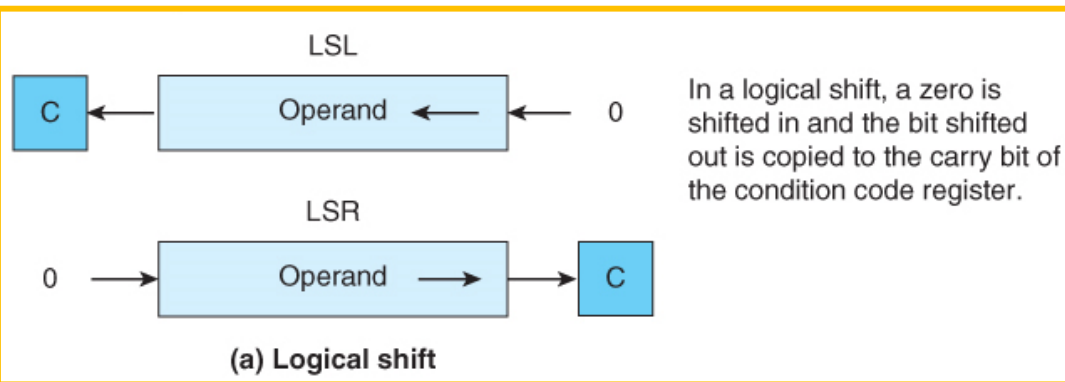
Office: MC-419

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`

$; [r0] \leftarrow [r1] + [r2] \times 2$

 - 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 moving operations
 - This way, a shift operation can be performed as a stand alone operation.
 - For example,

`MOV r3, r3, LSL #1`

$; [r3] \leftarrow [r3] \times 2$
 - **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   $r1 = r1 \cdot 2^5$ 
LSLS r1, r1, r3

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

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

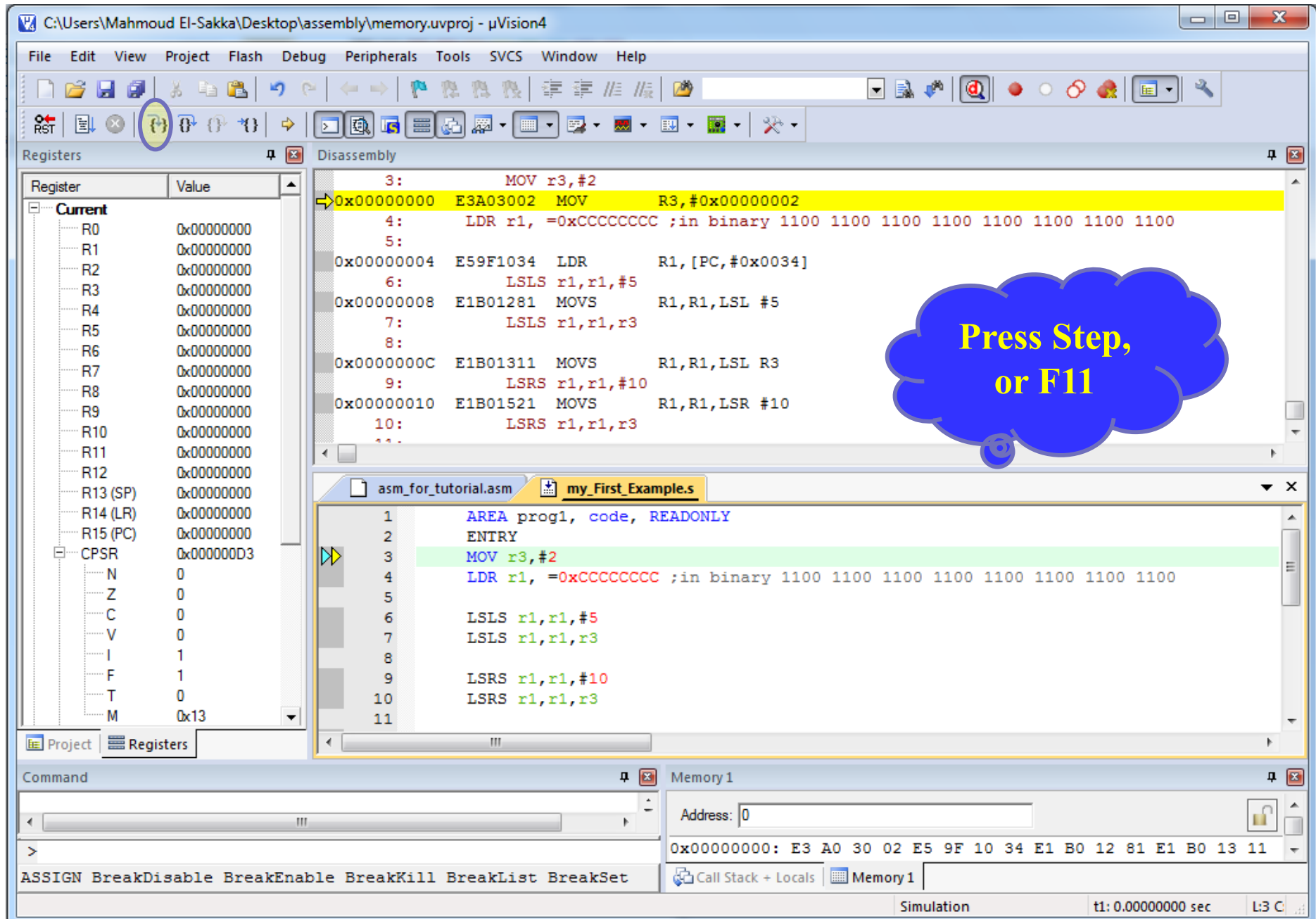
ASRS r1, r1, r3

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

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

```

ARM's Data-Processing Instructions (Shift Operations)



The screenshot shows the uVision4 IDE with the following components:

- Registers Window:** Displays the current state of ARM registers. R0-R15 are at 0x00000000. CPSR is 0x000000D3. N=0, Z=0, C=0, V=0, I=1, F=1, T=0, M=0x13.
- Disassembly Window:** Shows the assembly code being executed. The current instruction is `MOV r3, #2` at address 0x00000000. Subsequent instructions include `LDR r1, =0xCCCCCCCC`, `LDR R1, [PC, #0x0034]`, `LSLS r1, r1, #5`, `MOVS R1, R1, LSL #5`, `LSLS r1, r1, r3`, `MOVS R1, R1, LSL R3`, `LSRS r1, r1, #10`, `MOVS R1, R1, LSR #10`, and `LSRS r1, r1, r3`.
- Source Code Window:** Shows the assembly code for `my_First_Example.s`. The code includes `AREA prog1, code, READONLY`, `ENTRY`, and the same sequence of instructions as the disassembly window.
- Toolbar:** The 'Step' button (represented by a single-step icon) is circled in blue, with a callout bubble indicating to 'Press Step, or F11'.
- Command Window:** Contains the text `ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet`.
- Memory Window:** Shows the memory address 0x00000000 with the value `E3 A0 30 02 E5 9F 10 34 E1 B0 12 81 E1 B0 13 11`.
- Status Bar:** Indicates 'Simulation' mode with a timer at `t1: 0.00000000 sec` and a CPU frequency of `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

```
- 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

```
- Toolbar:** The 'Step' button (represented by a right-pointing arrow) is circled in blue.
- Callout:** A blue cloud-shaped bubble contains the text "Press Step, or F11".
- Memory Window:** Shows address 0x00000040 with hex data CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00.
- Command Window:** Contains the text "ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet".
- Status Bar:** Shows "Simulation" and "t1: 0.00000000 sec".

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. R1 is highlighted with a value of 0xCCCCCCCC. R15 (PC) is 0x00000008.
- Disassembly Window:** Shows the assembly code being executed. The current instruction is `LSLS r1, r1, #5` at address 0x00000008.
- Source Code Window:** Shows the assembly code being edited. The current instruction is `LSLS r1, r1, #5`.
- Diagram:** A diagram illustrating the LSL (Logical Shift Left) operation. It shows a carry flag (C) and an operand being shifted left by 5 bits. The operand is 0xCCCCCCCC, and the result is 0xCCCCCCCC.
- Blue Cloud Bubble:** Contains the text "Press Step, or F11".

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:**

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 Panel:**

```

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 File Panel (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

```
- Memory Panel:**

Address: 0

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

Annotations:

- A yellow box highlights the **LSL** instruction: `LSL` with a diagram showing the **Operand** being shifted left into the **C** (Carry) flag.
- A red arrow points from the **C** flag in the CPSR register to the **LSL** instruction.
- A green arrow points from the **C** flag in the CPSR register to the **LSL** instruction.
- A blue circle with the number **1** highlights the **LSL** instruction.
- Two yellow boxes show the binary representation of the operand `0xCCCCCCCC` (1100 1100 1100 1100 1100 1100 1100 1100) and the result of the shift operation (1001 1001 1001 1001 1001 1001 1001 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 Values:**
 - Initial value of R1: 1001 1001 1001 1001 1001 1001 1000 0000
 - Value after LSL R3: 0110 0110 0110 0110 0110 0110 0000 0000
- Instruction:** A blue cloud contains the text "Press Step, or F11".
- Memory Window:**

Address: 0

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

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 yellow. A red arrow points from this instruction to the R1 register in the Registers panel.
- Source Code Panel:** Shows the assembly code in `asm_for_tutorial.asm`. The instruction `LSLS r1, r1, #5` is highlighted in green. A blue arrow points from this instruction to the Disassembly panel.
- Memory Panel:** Shows the memory address 0x00000040 with the value `CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00`.
- Annotations:**
 - A yellow box at the top right shows the LSL instruction with a diagram: `C ← Operand ← 0`.
 - Two yellow boxes show the binary representation of the register R1 before and after the shift operation. The first box shows the value `1001 1001 1001 1001 1001 1001 1001 1000 0000`. The second box shows the value `0110 0110 0110 0110 0110 0110 0110 0000 0000`. A red arrow points from the first box to the second box, indicating the shift operation.
 - A green circle with the number 0 is placed next to the C flag in the CPSR register, indicating its state after the shift operation.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:**

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 Panel:**

```

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 Panel (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. Below it, two binary strings are shown:

0110 0110 0110 0110 0110 0110 0000 0000

 and

0000 0000 0001 1001 1001 1001 1001 1001
- Annotation:** A blue cloud with the text "Press Step, or F11" points to the step-through button in the IDE.

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
16:         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

```
- 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 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:** A diagram showing the LSR (Logical Shift Right) operation. It takes a value '0' and shifts it right through an 'Operand' box to a 'C' (Carry) flag box.
- Hexadecimal Values:**
 - Initial value: 0000 0000 0001 1001 1001 1001 1001 1001
 - Result after shift: 0000 0000 0000 0110 0110 0110 0110 0110
- 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 for an ARM assembly project. The main window shows the disassembly of assembly code, with the following instructions visible:

```

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

```

The Registers window on the left shows the current state of the registers:

Register	Value
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

The CPSR value 0x000000D3 is highlighted in blue. A green arrow points from the CPSR value to the binary representation of the CPSR value shown in the Memory window: 0000 0000 0000 0110 0110 0110 0110 0110. A red arrow points from the R1 register value to the instruction 0x00000018: ASRS r1,r1,#2. A blue arrow points from the instruction 0x0000001C: ASRS r1,r1,#16 to the CPSR value. A green arrow points from the instruction 0x00000018: ASRS r1,r1,#2 to the CPSR value. A green arrow points from the instruction 0x0000001C: ASRS r1,r1,#16 to the CPSR value. A green arrow points from the instruction 0x00000020: ASRS r1,r1,#16 to the CPSR value. A green arrow points from the instruction 0x00000018: ASRS r1,r1,#2 to the CPSR value. A green arrow points from the instruction 0x0000001C: ASRS r1,r1,#16 to the CPSR value. A green arrow points from the instruction 0x00000020: ASRS r1,r1,#16 to the CPSR value.

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. The Command window shows the text: ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet.

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 0x00000018, and the CPSR is at 0x000000D3. The C flag is 0.
- Disassembly Window:** Shows the assembly code being executed. The instruction at address 0x00000018 is `ASRS r1,r1,#2`, which is highlighted in yellow.
- Source File Window:** Shows the assembly code for `my_First_Example.s`. The instruction `ASRS r1,r1,#2` is highlighted in green.
- Diagram:** A diagram of the ASR (Arithmetic Shift Right) operation. It shows the MSB (Most Significant Bit) of the operand being shifted into the C (Carry) flag.
- Hexadecimal Values:** Two rows of hexadecimal values are shown, representing the state of the register and the carry flag before and after the shift operation.

0000 0000 0000 0110 0110 0110 0110 0110	0110
0000 0000 0000 0001 1001 1001 1001 1001	1001
- Callout Box:** A blue cloud-shaped box with the text "Press Step, or F11" and a circular arrow icon, indicating the next step in the debugging process.

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	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

```
- asm_for_tutorial.asm Window:**

```

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:

- A red arrow points from the value of R1 (0x00019999) to the instruction `LSL #15` at address 0x0000001C.
- A blue arrow points from the CPSR register value (0x200000D3) to the instruction `LSL #15`.
- A green arrow points from the CPSR register value (0x200000D3) to the instruction `LSL #15`.
- A yellow box highlights the binary value `0000 0000 0000 0001 1001 1001 1001 1001`, which is the result of the LSL operation on R1.
- A red circle with the number 1 is next to the binary value.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows 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

```
- 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

```
- Diagram:** A diagram showing the LSL operation: $C \leftarrow \text{Operand} \leftarrow 0$. The 'C' (Carry) flag is highlighted in a blue box.
- Bit Pattern:** A 32-bit binary representation of the register R1 (0x00019999) is shown: `0000 0000 0000 0001 1001 1001 1001 1001`. The first 16 bits are yellow, and the last 16 bits are green.
- Callout:** A blue cloud-shaped box with the text "Press Step, or F11" and a circular arrow icon.
- Memory Window:** Shows address 0x00000040 with data: `CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00`.
- Status Bar:** Simulation, t1: 0.00000000 sec, L13.

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 ARM registers. R1 contains the value 0xCCCC8000. Other registers like R0, R2, R3, etc., contain zeros. The CPSR register shows flags: N=1, Z=0, C=0, V=0, I=1, F=1, T=0, M=0x13.
- Disassembly Window:** Shows the disassembled instructions for the assembly file 'asm_for_tutorial.asm'. The instructions are:
 - 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:** Shows the original assembly code for 'my_First_Example.s':
 - 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
- 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.

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 0 in the binary representation, which corresponds to the least significant bit of the register value.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:**

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 Panel:**

```

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:
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 File Panel (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 of 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 displays the uVision4 IDE interface for an ARM assembly project. The **Registers** window on the left shows the current state of the processor registers. Register R1 contains the value 0xFFFFCCCC. The **Disassembly** window shows the assembly code being executed. The instructions are as follows:

Address	Instruction	Comment
0x00000013	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	

A green box highlights the value **1111 1111 1111 1111 1100 1100 1100 1100**, which is the result of the ASRS instruction at address 0x00000016. A green arrow points from this box to the ASRS instruction. A blue arrow points from the ASRS instruction to the CPSR register in the Registers window, which shows the value 0xA00000D3. The **Command** window at the bottom shows the command **ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet**.

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

```
- 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

```
- Diagram:** A block diagram of the ASR (Arithmetic Shift Right) instruction. It shows an 'ASR' block with an input 'MSB' (Most Significant Bit) and an 'Operand'. The output of the ASR block is connected to a box labeled 'C' (Carry flag).
- Hexadecimal Values:**
 - Initial value of R1: 0xFFFFCCCC (1111 1111 1111 1111 1100 1100 1100 1100)
 - Value after ASRS r1, r1, r3: 0xFFFFCCCC (1111 1111 1111 1111 1111 0011 0011 0011)
- Callout:** A blue cloud-shaped box with the text "Press Step, or F11" pointing to the Step button in the IDE toolbar.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the uVision4 IDE interface for an ARM assembly project. The **Registers** window on the left shows the current state of the processor registers. Register R1 contains the value 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 at address 0x00000028, **RORS r1,r1,#4**, is highlighted in yellow. A yellow box to the right of this instruction shows the binary representation of the value 0xFFFF333 after a right rotate by 4 bits: **1111 1111 1111 1111 1111 0011 0011 0011**. A red arrow points from register R1 to this box, and a green arrow points from the RORS instruction to it. A blue arrow points from the CPSR register to the source code window.

The **Source** window shows the assembly code for **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

```

The instruction **RORS r1,r1,#4** at line 18 is highlighted in green. A green arrow points from this instruction to the yellow box showing the binary result. The **Command** window at the bottom shows the simulation status: **Simulation**, **t1: 0.00000000 sec**, and **L:18**.

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`. Below it, the binary representation of the instruction is shown: `1111 1111 1111 1111 1111 0011 0011 0011` (hex 00000028) and `0011 1111 1111 1111 1111 1111 0011 0011` (hex 0000002C).
- Source File 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 'C' box has an arrow pointing back to the 'Operand' box, indicating a circular shift.
- Callout:** A blue cloud-shaped callout with the text "Press Step, or F11" pointing to the Step button in the toolbar.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the uVision4 IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The main workspace is divided into several panels:

- Registers:** A list of registers (R0 to R15, CPSR) with their current values. R15 (PC) is highlighted with a value of 0x0000002C.
- Disassembly:** A list of instructions with their addresses and assembly code. The instruction at address 0x0000002C is highlighted in yellow: `0x0000002C E1B01371 MOVs R1,R1,ROR #3`. A red arrow points from this instruction to the assembly view.
- Assembly:** A list of instructions with their addresses and assembly code. The instruction at address 0x0000002C is highlighted in green: `19 | RORS r1,r1,#3`. A green arrow points from this instruction to the registers window.
- Memory:** A window showing the memory address 0x00000040 containing the value `CC CC CC CC`.

Annotations include a red arrow pointing from the disassembly view to the assembly view, and a green arrow pointing from the assembly view to the registers window. A yellow box highlights the instruction at address 0x0000002C in the disassembly view, and a green box highlights the instruction at address 0x0000002C in the assembly view.

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 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 STCGTL p12,CR12,[R12],{204}
  
```
- Assembly Source 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 flow: "Operand" → "C" (Carry flag). Arrows show the carry flag feeding back into the operand.
- 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" points to the assembly source window.

ARM's Data-Processing Instructions (Shift Operations)

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

- Registers Window:** Displays the current state of registers. R1 is highlighted with a value of 0xCFFFFFFC. R15 (PC) is 0x00000030. CPSR is 0xA00000D3. Status flags N, Z, C, V, I, F, T, M are also shown.
- Disassembly Window:** Shows the machine code for the assembly file. The instruction at address 0x00000030 is highlighted in yellow: `RRXS r1, r1`. A red arrow points from this instruction to the assembly window, and a green arrow points from the assembly window to this instruction. A yellow box highlights the instruction, and a red circle with the number '1' is placed next to it.
- Assembly Window:** Shows the source code for 'my_First_Example.s'. The instruction `RRXS r1, r1` is highlighted in green. A red arrow points from this instruction to the disassembly window, and a green arrow points from the disassembly window to this instruction.
- Memory Window:** Shows the memory address 0x00000040 with a value of 0x00000000.

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
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}
  
```

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 displays the uVision4 IDE interface with the following components:

- Registers Window:** Shows the current state of registers. R1 contains the value 0xE7FFFE6. Other registers like R0, R2, R3, etc., are at 0x00000000. The CPSR register shows flags: N=1, Z=0, C=0, V=0, I=1, F=1, T=0, M=0x13.
- Disassembly Window:** Shows the disassembled instructions for the assembly file. The instructions for R1 are:
 - 0x00000028: E1B01261 MOVN R1,R1,ROR #4
 - 19: RORS r1,r1,r3
 - 20: 0x0000002C: E1B01371 MOVN R1,R1,ROR R3
 - 21: RRXS r1,r1
 - 22: 0x00000030: E1B01061 MOVN R1,R1,RRX
 - 23: RRXS r1,r1
 - 24: 0x00000034: E1B01061 MOVN R1,R1,RRX
 - 25: RRXS r1,r1
 - 26: 0x00000038: E1B01061 MOVN R1,R1,RRX
 - 27: RRXS r1,r1
 - 28: 0x0000003C: E1B01061 MOVN R1,R1,RRX
 - 29: RRXS r1,r1
 - 30: 0x00000040: CCCCCCCC STCGTL p12,CR12,[R12],{204}
 - 31: 0x00000044: CCCCCCCC UNDEF R10,R1,R1
- Source Code Window:** Shows the assembly code for the file. The instructions for R1 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 callout box shows the binary representation of the value in R1: 1110 0111 1111 1111 1111 1111 1111 0110. A green arrow points from the R1 register value to this binary representation.
- Command Window:** Shows the command prompt with the text: 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:** Shows the current state of registers. R1 contains 0xE7FFFE6. The CPSR (Current Program Status Register) is shown with the Carry flag (C) set to 0.
- Disassembly Window:** Shows the assembly code being executed. The instruction at address 0x00000034 is `RRXS r1, r1`, which is highlighted in yellow.
- Diagram:** A diagram titled "Rotate right through carry" shows a "Register" box connected to a "Carry" box. An arrow points from the Register to the Carry, and another arrow points from the Carry back to the Register, indicating a circular shift.
- Bit Patterns:** Two bit patterns are shown, representing the state of the register and carry before and after the shift operation. The top pattern is `1110 0111 1111 1111 1111 1111 1110 0110` with a red '0' at the end. The bottom pattern is `0111 0011 1111 1111 1111 1111 1111 0011` with a red '0' at the end.
- Assembly Code:** The code in the bottom window includes instructions like `ASRS r1, r1, r3`, `RORS r1, r1, #4`, `RORS r1, r1, r3`, `RRXS r1, r1`, and `END`.
- Call Stack + Locals:** Shows the current state of the call stack and local variables.
- 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`.
- Simulation Status:** Shows the simulation is running at `tl: 0.00000000 sec` and `L:22`.

A blue cloud bubble with the text "Press Step, or F11" is overlaid on the bottom right of the IDE 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 ARM registers. R1 contains the value 0x73FFFFFF. The CPSR register shows flags: N=0, Z=0, C=0, V=0, I=1, F=1, T=0, M=0x13.
- Disassembly Window:** Shows the disassembled instructions. The instruction at address 0x00000038 is highlighted: `MOV R1, R1, RRX`. 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 to the source code. A blue arrow points from the `RRXS r1, r1` instruction to the CPSR register. A green circle with the number 0 is next to the last '0' in the binary representation.

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 contains 0x73FFFFFF. R15 (PC) contains 0x00000038.
- Disassembly Panel:** Shows the assembly code for the current instruction. The instruction at address 0x00000038 is `RRXS r1, r1`, which is highlighted in yellow.
- Source Code Panel:** Shows the assembly code for the current file, `my_First_Example.s`. The instruction `RRXS r1, r1` is highlighted in green.
- Diagram:** A diagram titled "Rotate right through carry" shows a "Register" box with an arrow pointing to a "Carry" box. The diagram illustrates the rotation of the register's value to the right, with the carry bit being shifted into the register's least significant bit.
- Bit Pattern:** A bit pattern is shown in a yellow box, representing the register's value after the rotation. The bit pattern is `0111 0011 1111 1111 1111 1111 1111 0011`. The last bit, `1`, is highlighted in red.
- Annotation:** A blue cloud-shaped callout with the text "Press Step, or F11" points 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	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 STCGTL    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:

- A red arrow points from R1 (0x39FFFFFF) to the MOV instruction at address 0x0000002C.
- A green arrow points from the CPSR C flag (1) to the RORS instruction at address 0x00000018.
- A yellow box highlights the binary value 0011 1001 1111 1111 1111 1111 1111 1001, which is the result of the RRX instruction.
- A blue box highlights the instruction RRXS r1, r1 at address 0x0000003C.

1

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 0x0000003C. CPSR is 0x200000D3. The 'C' (Carry) flag is set to 1.
- Disassembly Window:** Shows assembly instructions. The instruction at address 0x0000003C is `RRXS r1, r1`, which is highlighted in yellow.
- Source File Window:** Shows the assembly code for `my_First_Example.s`. The instruction `RRXS r1, r1` at line 24 is highlighted in green.
- Diagram:** A red box highlights the 'Rotate right through carry' operation. It shows a 'Register' box connected to a 'Carry' box. An arrow points from the right side of the Register to the Carry box, and another arrow points from the Carry box back to the left side of the Register.
- Bit Patterns:** Two bit patterns are shown in yellow boxes:
 - Top: 0011 1001 1111 1111 1111 1111 1111 1111 1001
 - Bottom: 1001 1100 1111 1111 1111 1111 1111 1111 1100
- Call to Action:** A blue cloud contains the text "Press Step, or F11".

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers:**

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:**

```

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 STCGT     p12,CR12,[R12],{204}
0x00000044 00000000 ANDEQ     R0,R0,R0
0x00000048 00000000 ANDEQ     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

```
- Memory 1:**

Address: 0

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

A red arrow points from register R1 (0x9CFFFFFF) to the assembly code. A green arrow points from the assembly code to the CPSR register. A yellow box highlights the binary value 1001 1100 1111 1111 1111 1111 1111 1100, which is the value of register R1. A blue arrow points from this binary value to the assembly code. A green arrow points from the assembly code to the CPSR register. A red circle with the number 1 is next to the binary value.

1

ARM's Data-Processing Instructions (Shift Operations)

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

```
ENTRY
```

```
MOV r3, #2
```

```
LDR r1, =0xCCCCCCCC ;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”

