

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

Winter 2021-2022

Instructor: Mahmoud R. El-Sakka

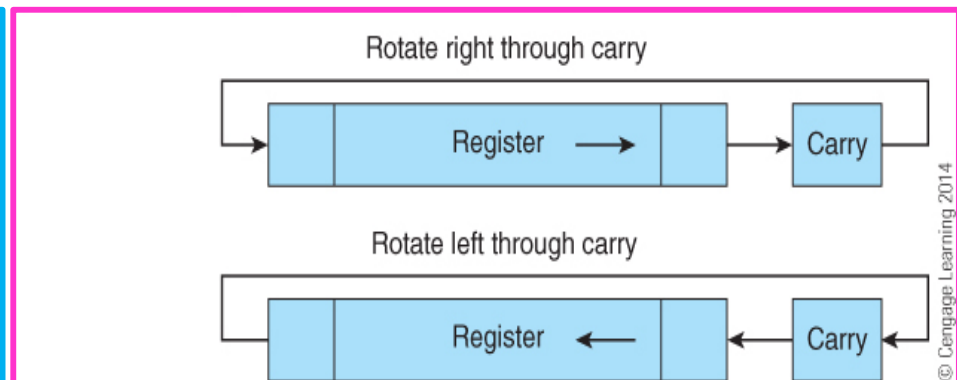
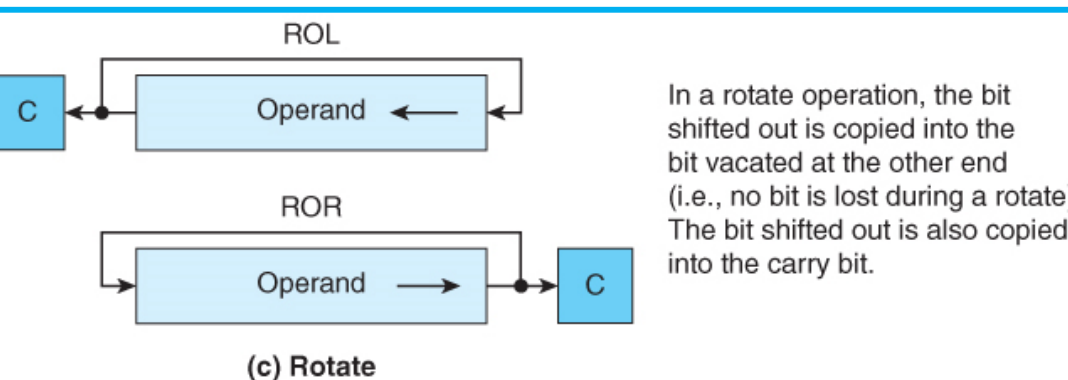
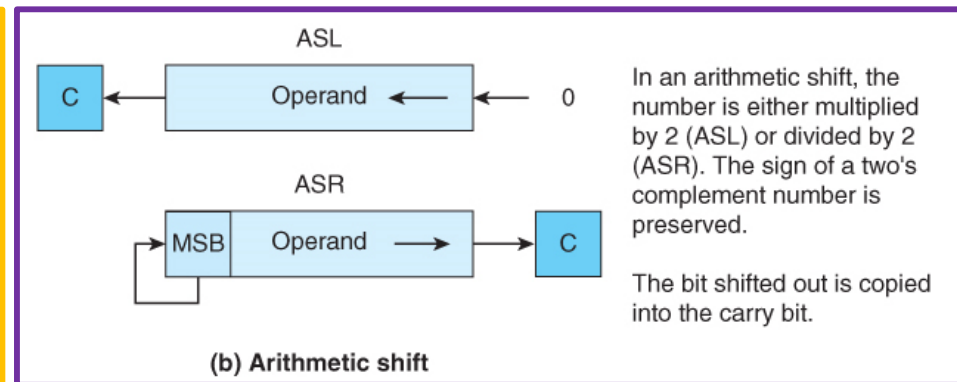
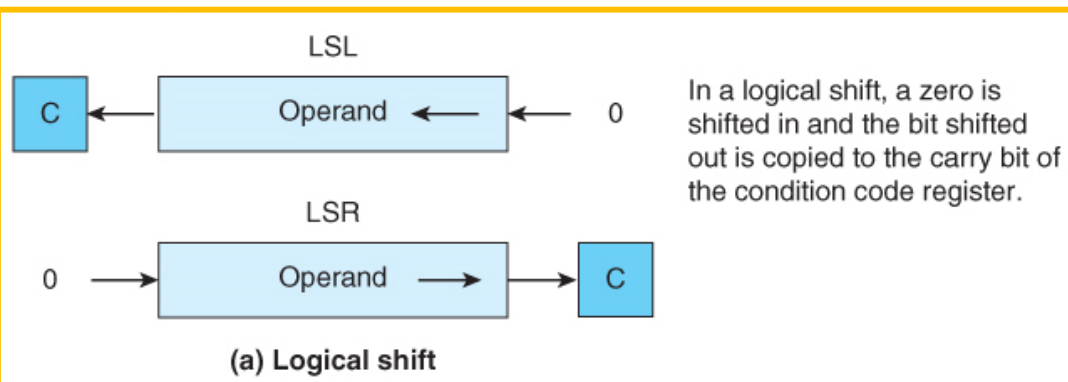
Office: MC-419

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 uVision4 IDE with the following components:

- Registers Window:** Lists registers R0 through R15 and CPSR. R15 (PC) is highlighted as the current register.
- Disassembly Window:** Shows the following instructions:
 - 3: MOV r3, #2
 - 4: LDR r1, =0xCCCCCCCC ;in binary 1100 1100 1100 1100 1100 1100 1100 1100
 - 5: (blank line)
 - 6: LDR R1, [PC, #0x0034]
 - 7: LSLS r1, r1, #5
 - 8: MOVs R1, R1, LSL #5
 - 9: LSLS r1, r1, r3
 - 10: MOVs R1, R1, LSR #10
 - 11: LSRS r1, r1, r3
- Source Code Window (asm_for_tutorial.asm):** Shows the corresponding assembly code:
 - 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: (blank line)
 - 6: LSLS r1, r1, #5
 - 7: LSLS r1, r1, r3
 - 8: (blank line)
 - 9: LSRS r1, r1, #10
 - 10: LSRS r1, r1, r3
 - 11: (blank line)
- Toolbar:** The Step button (a circular arrow) is circled in blue.
- Callout Bubble:** A blue cloud-shaped bubble with the text "Press Step, or F11" points to the Step button.
- Memory Window:** Shows the address 0 and the memory contents: 0x00000000: E3 A0 30 02 E5 9F 10 34 E1 B0 12 81 E1 B0 13 11.
- 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:**

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 (a blue square with a white right-pointing arrow) is circled in blue.
- Callout:** A blue cloud-shaped bubble with the text "Press Step, or F11" and a mouse cursor icon pointing to the Step button.
- Memory Window:** Shows address 0x00000040 with data CC CC CC CC 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:** A list of registers (R0-R15, CPSR) with their current values. R15 (PC) is highlighted with a value of 0x00000008.
- Disassembly:** A list of assembly instructions with their addresses and values. The instruction at address 0x00000008 is highlighted: `MOV r1, r1, LSL #5`.
- asm_for_tutorial.asm:** A file containing the assembly code. The instruction `LSLS r1, r1, #5` is highlighted.
- Memory 1:** A window showing memory addresses and values. The address 0x00000040 is shown with a value of 00 CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00.
- 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 1100 1100 1100 1100 1100 1100 1100 1100. The result is 1001 1001 1001 1001 1001 1001 1001 0000.
- Annotation:** A blue cloud with the text "Press Step, or F11" is overlaid on the assembly code.

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, =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):**

```

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

```
- 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 `C ← Operand ← 0`.
- 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.
- A yellow box highlights the binary value `1100 1100 1100 1100 1100 1100 1100 1100` in the source code.
- A green box highlights the binary value `1001 1001 1001 1001 1001 1001 1001 0000` in the source 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, =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:**

```

1001 1001 1001 1001 1001 1001 1001 1000 0000
0110 0110 0110 0110 0110 0110 0110 0000 0000

```
- Instruction:** A blue cloud with 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:**

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 E39F1034 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 #5
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

```
- Annotations:**
 - A yellow box at the top right shows the LSL instruction: `C ← Operand ← 0`.
 - Two yellow boxes show the binary value of R1 before and after the LSL instruction:

Before: 1001 1001 1001 1001 1001 1001 1001 1000 0000

After: 0110 0110 0110 0110 0110 0110 0110 0000 0000
 - A red arrow points from the LSL instruction in the disassembly panel to the C flag in the registers panel.
 - A green arrow points from the LSL instruction in the disassembly panel to the C flag in the registers panel.
 - A blue arrow points from the LSL instruction in the disassembly panel to the C flag in the registers panel.

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
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

```
- Assembly 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 value of R1: 0110 0110 0110 0110 0110 0110 0000 0000
 - Result after LSR #10: 0000 0000 0001 1001 1001 1001 1001 1001
- Instruction:** A blue cloud contains the text "Press Step, or F11".
- Memory Window:** Shows address 0x00000040 with value 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 shows the uVision4 IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. Below the menu is a toolbar with various icons for file operations, editing, and debugging. 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 blue background and shows the value 0x00000014. CPSR is highlighted with a blue background and shows the value 0x200000D3. The C flag in CPSR is set to 1.
- Disassembly:** A list of assembly instructions. The instruction at address 0x00000010 is highlighted with a green background: `10: LSRS r1, r1, r3`. The instruction at address 0x00000014 is highlighted with a red background: `14: LSRS r1, r1, r3`. A green box highlights the CPSR register, and a red box highlights the R1 register. A green arrow points from the CPSR register to the Disassembly window, and a red arrow points from the R1 register to the Disassembly window.
- asm_for_tutorial.asm:** A file containing assembly code. The instruction `LSRS r1, r1, r3` at line 10 is highlighted with a green background. A green arrow points from the CPSR register to this instruction, and a red arrow points from the R1 register to this instruction.
- Memory 1:** A panel showing memory addresses and their values. The address 0x00000040 is highlighted, and the value is `CC CC CC CC 00 00 00 00 00 00 00 00 00 00`.

The bottom status bar shows the simulation time as 0.00000000 sec and the instruction pointer as L:10.

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) register.
- 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)

[illegible]

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. CPSR is at 0x000000D3. The C flag is 0.
- Disassembly Window:** Shows the assembly code for the current instruction. The instruction at address 0x00000018 is `ASRS r1,r1,#2`. The instruction at address 0x0000001C is `LSLS r1,r1,#15`.
- ASR Diagram:** A diagram showing the ASR instruction. It takes an operand and shifts it right by a specified amount (MSB). The result is stored in the C flag.
- Assembly Code Window:** Shows the source code for the assembly. The instruction at line 12 is `ASRS r1,r1,#2`. The instruction at line 13 is `LSLS r1,r1,#15`. The instruction at line 14 is `ASRS r1,r1,#16`.
- 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 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 µVision4 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 in the image:

- A red arrow points from the `LSRS r1,r1,r3` instruction to the R1 register value.
- A green arrow points from the `LSL #15` instruction to the CPSR register value.
- A yellow box highlights the binary value `0000 0000 0000 0001 1001 1001 1001 1001`, which is the result of the `LSL #15` instruction.
- A blue circle with the number 1 is next to the memory dump.

1

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 register value is shown: `0000 0000 0000 0001 1001 1001 1001 1001`. The first 16 bits are yellow, and the last 16 bits are green. A purple arrow points from the 15th bit (1) to the 16th bit (1), indicating a shift.
- Callout:** A blue cloud-shaped box with the text "Press Step, or F11" and a circular arrow icon.
- Command Panel:** Shows "ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet".
- Memory Panel:** Shows "Memory 1" with address 0 and a hex dump starting with `0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00`.
- Status Bar:** Shows "Simulation", "t1: 0.00000000 sec", and "L:13".

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,#16
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      RRXS r1,r1

```
- 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 R1 register value to the ASRS instruction, a green arrow pointing from the R1 register value to the ASRS instruction, and a blue arrow pointing from the R1 register value to the ASRS instruction. A yellow box highlights the binary representation of the R1 value: 1100 1100 1100 1100 1000 0000 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	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:
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      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 MOVSR R1,R1,LSL #15
14:      ASRS r1,r1,#16
15:
0x00000020 E1B01841 MOVSR R1,R1,ASR #16
16:      ASRS r1,r1,r3
17:
0x00000024 E1B01351 MOVSR R1,R1,ASR R3
18:      RORS r1,r1,#4
0x00000028 E1B01261 MOVSR R1,R1,ROR #4
19:      RORS r1,r1,r3
20:
0x0000002C E1B01371 MOVSR R1,R1,ROR R3
21:      RRRXS r1,r1
22:      RRRXS r1,r1
23:      RRRXS r1,r1

```
- asm_for_tutorial.asm 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      RRRXS r1,r1
22      RRRXS r1,r1
23      RRRXS r1,r1

```
- 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 ASRS instruction in the Disassembly window to the R1 register in the Registers window.
- A blue arrow points from the ASRS instruction in the Disassembly window to the CPSR register in the Registers window.
- A green arrow points from the ASRS instruction in the Disassembly window to the CPSR register in the Registers window.
- A green box highlights the binary value 1111 1111 1111 1111 1100 1100 1100 1100.
- A red circle with the number 1 is next to the green box.

1

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:**

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 Panel:**

```

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
    
```
- Binary Representation of ASR:**

```

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

```
- Code Editor:**

```

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 Panel:**

```

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet

```
- Memory Panel:**

```

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

```

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 ARM registers. R1 contains 0xFFFF333. The CPSR register shows the C flag (Carry) set to 0.
- Disassembly Window:** Shows the disassembled instructions. The instruction at address 0x00000028 is highlighted: `RORS r1,r1,#4`. A yellow box highlights the binary value `1111 1111 1111 1111 1111 0011 0011 0011` associated with this instruction.
- 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
      
```
- 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:** 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 in a yellow box:
 - 1111 1111 1111 1111 1111 0011 0011 0011
 - 0011 1111 1111 1111 1111 1111 0011 0011
- Call to Action:** A blue cloud-shaped bubble with the text "Press Step, or F11" and a circular arrow icon.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays 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 E1B01271 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}

```
- 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 the assembly instruction at address 0x0000002C to the R1 register value, and a green arrow pointing from the assembly instruction at address 0x00000040 to the memory window. A binary representation of the value 0011 1111 1111 1111 1111 1111 0011 0011 is shown, with a circled '0' at the end.

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 STCGTL   p12,CR12,[R12],{204}

```
- 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 flow from "Operand" to a circular shift register, which then outputs to "C" (Carry flag).
- Hexadecimal Values:**
 - Initial value of R1: 0011 1111 1111 1111 1111 1111 0011 0011
 - Value after RORS r1, r1, #4: 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 Panel:**

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 Panel:**

```

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}
  
```
- Source Code Panel (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 Panel:**

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 `RRXS r1, r1` instruction to the R1 register.
- A green arrow points from the `RRXS r1, r1` instruction to the 'C' flag in the CPSR register.
- A yellow box highlights the binary value `1100 1111 1111 1111 1111 1111 1100 1100` for the R1 register, with a circled '1' next to it.

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 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:**

Register	Value
R0	0x00000000
R1	0xE7FFFE6
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)	0x00000034
CPSR	0x800000D3
N	1
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 STCGTL    p12,CR12,[R12],{204}

```
- 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

A binary representation of the R1 register value is shown as: 1110 0111 1111 1111 1111 1111 1111 0110. A green circle with the number 0 is also present on the right side of the image.

ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
R0	0x00000000
R1	0xE7FFFE6
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)	0x00000034
CPSR	0x800000D3
N	1
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
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}

```
- 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 red box highlights the "Rotate right through carry" operation. It shows a "Register" box with an arrow pointing to a "Carry" box, and another arrow pointing from the "Carry" box back to the "Register" box.
- Bit Patterns:** Two rows of bit patterns are shown:

1110 0111 1111 1111 1111 1111 1111 1110 0110

0111 0011 1111 1111 1111 1111 1111 1111 0011
- Annotation:** A blue cloud contains the text "Press Step, or F11".

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 the ARM registers. The **Disassembly** window in the center shows the assembly code being executed. The **asm_for_tutorial.asm** file is open, showing the following instructions:

```

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 **Registers** window shows the following values:

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

The **Disassembly** window shows the following instructions:

```

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}

```

The **asm_for_tutorial.asm** file shows the following instructions:

```

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 **Memory** window shows the address 0x00000040 with the value 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.

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
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
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 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 in the image:

- A red arrow points from the highlighted value **0x39FFFFFF** in the Registers window to the instruction **RRXS r1, r1** in the Disassembly window.
- A green arrow points from the highlighted value **CPSR.C = 1** in the Registers window to the instruction **RORS r1, r1, r3** in the Disassembly window.
- A blue arrow points from the instruction **ASRS r1, r1, r3** in the Source Code window to the instruction **ASRS r1, r1, r3** in the Disassembly window.
- A yellow box highlights the binary value **0011 1001 1111 1111 1111 1111 1111 1001**, which is the hexadecimal value 0x39FFFFFF.
- A red circle with the number **1** is located near the bottom right of 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	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:** A list of registers with their current values. R1 is highlighted with a value of 0x9CFFFFFF. R15 (PC) is 0x00000044. CPSR is 0xA00000D3.
- Disassembly:** A list of assembly instructions. The instruction 'RRXS r1, r1' is highlighted at address 0x00000030. The instruction 'RRXS r1, r1' is also highlighted at address 0x00000034.
- asm_for_tutorial.asm:** A file containing assembly code. The instruction 'RRXS r1, r1' is highlighted at line 21. The instruction 'RRXS r1, r1' is also highlighted at line 22.
- Memory 1:** A window showing memory addresses and values. The address 0x00000040 is highlighted, showing a value of 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.

The binary value **1001 1100 1111 1111 1111 1111 1111 1100** is shown in a yellow box. A red arrow points from the R1 register value (0x9CFFFFFF) to the assembly code line 'RRXS r1, r1'. A green arrow points from the '1' in the binary value to the '1' in the assembly code line 'RRXS r1, r1'. A blue arrow points from the '1' in the binary value to the '1' in the assembly code line 'RRXS r1, r1'.

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”

