

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*

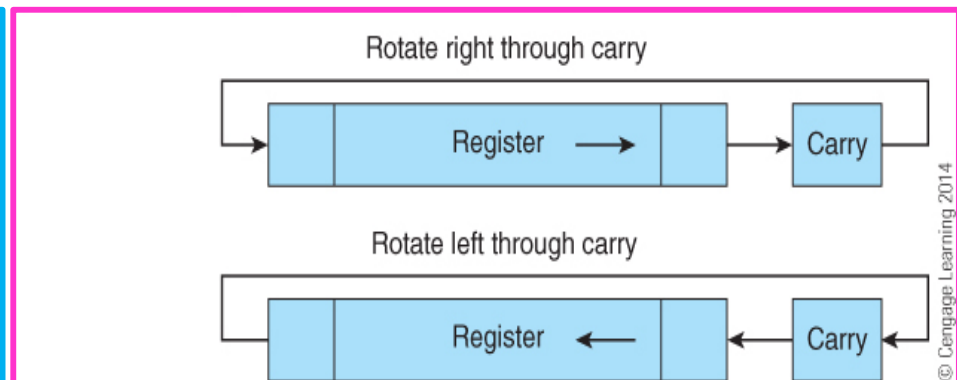
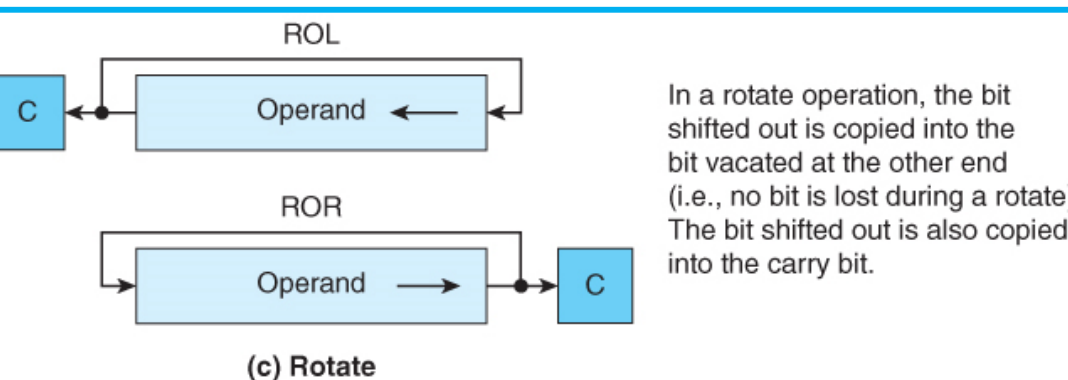
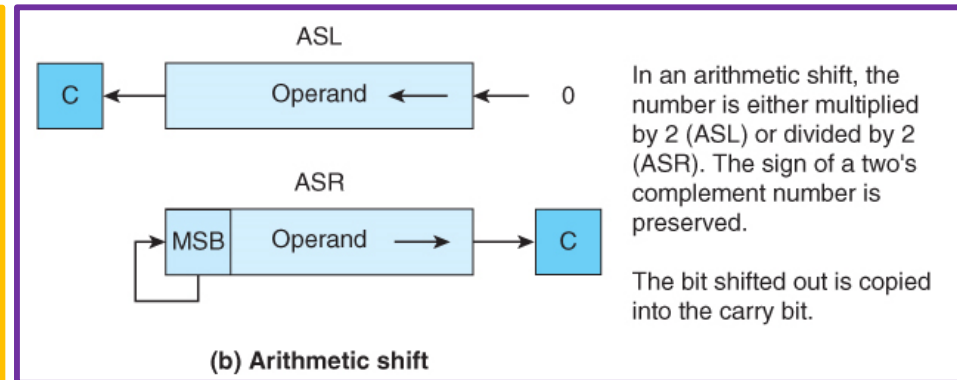
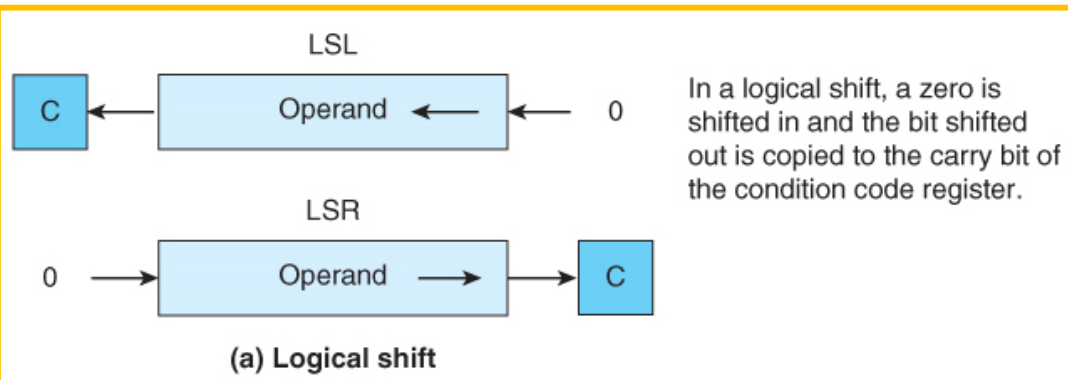
*Office: MC-419*

*Email: [elsakka@csd.uwo.ca](mailto: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  $\geq 32$ , 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 moving 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

```

# ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE interface. The top toolbar has a button circled in blue, which is the 'Step' button (represented by a single step icon). A blue callout bubble with the text 'Press Step, or F11' points to this button.

The 'Registers' window on the left shows the current state of the ARM registers. The 'Current' register is R0, with a value of 0x00000000. Other registers R1 through R15 are also shown with their values. The CPSR register is also visible, showing flags N, Z, C, V, I, F, T, and M.

The 'Disassembly' window shows the following assembly code:

```

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

```

The 'asm\_for\_tutorial.asm' window shows the source code for 'my\_First\_Example.s':

```

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

```

The 'Memory' window at the bottom shows the memory address 0x00000000 and the corresponding instruction bytes: E3 A0 30 02 E5 9F 10 34 E1 B0 12 81 E1 B0 13 11.



# ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

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

```

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

```
- Source Code Window (asm\_for\_tutorial.asm):**

```

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

```
- Toolbar:** The Step button (a blue square with a white 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 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:** R0 (0x00000000), R1 (0xCCCCCCCC), R2 (0x00000000), R3 (0x00000002), R4 (0x00000000), R5 (0x00000000), R6 (0x00000000), R7 (0x00000000), R8 (0x00000000), R9 (0x00000000), R10 (0x00000000), R11 (0x00000000), R12 (0x00000000), R13 (SP) (0x00000000), R14 (LR) (0x00000000), R15 (PC) (0x00000008), CPSR (0x000000D3).
- Disassembly:**

```

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

```
- asm\_for\_tutorial.asm:**

```

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

```
- Memory 1:** Address: 0, 0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00

**Diagram:** LSL instruction format: C ← Operand ← 0

**Binary Representation:** 1100 1100 1100 1100 1100 1100 1100 1100

**Binary Representation:** 1001 1001 1001 1001 1001 1001 1001 0000

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

```

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

```
- Source Code Window (asm\_for\_tutorial.asm):**

```

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

```
- Annotations:**
  - A yellow box labeled "LSL" shows a diagram: `C ← Operand ← 0`.
  - Red arrows point from the "C" in the LSL diagram to the "C" bit in the CPSR register (value 1) and to the 31st bit of R1 (value 1).
  - Two binary strings are shown in yellow boxes:
    - Top: 1100 1100 1100 1100 1100 1100 1100 1100
    - Bottom: 1001 1001 1001 1001 1001 1001 1001 0000
  - A green circle with the number "1" is placed over the 31st bit of the bottom binary string.

# 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: a box labeled 'C' (Carry) is connected to a box labeled 'Operand', which is connected to a box labeled '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 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 Window:** Shows the current state of registers. R1 contains 0x66666600. The CPSR (CPSR) register shows the C flag (Carry) as 0.
- Disassembly Window:** Shows the assembly code being executed. The instruction at address 0x00000004 is `LSLS r1, r1, #5`, which shifts the value in R1 left by 5 bits.
- Assembly Source Window:** Shows the source code for `my_First_Example.s`. The instruction at line 9 is `LSRS r1, r1, #10`, which shifts the value in R1 right by 10 bits.
- Memory Window:** Shows the memory address 0x00000040 with the value 00 00 00 00.

Annotations in the image illustrate the effect of the `LSL` instruction:

- A diagram in the top right shows the `LSL` instruction taking an `Operand` and a shift count `0`, resulting in a new value `C` (Carry flag).
- A red arrow points from the `LSLS r1, r1, #5` instruction to the register R1, indicating the shift operation.
- A green arrow points from the `LSRS r1, r1, #10` instruction to the register R1, indicating the shift operation.
- Two binary strings are shown, representing the value in R1 before and after the shift operations:
  - Before: `1001 1001 1001 1001 1001 1001 1001 1000 0000`
  - After: `0110 0110 0110 0110 0110 0110 0110 0000 0000`

# ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers 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 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 (Carry flag).
- 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 Panel:** Address 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:9 C



# ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the Keil 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 panes:

- Registers:** A list of registers (R0 to R15, CPSR) with their current values. R1 is highlighted with a value of 0x00199999. The CPSR register shows flags: N=0, Z=0, C=1, V=0, I=1, F=1, T=0, M=0x13.
- Disassembly:** A list of assembly instructions with their addresses and opcodes. The instruction at address 0x00000014 is highlighted: `0x00000014 E1B01331 MOVs R1,R1,LSR R3`. A red arrow points from this instruction to the R1 register in the Registers pane.
- Source Code:** A list of source code lines corresponding to the assembly instructions. The line `10: | LSRS r1,r1,r3` is highlighted in green. A green arrow points from this line to the CPSR register in the Registers pane.
- Command Window:** Shows the simulation status: "Simulation" and "t1: 0.00000000 sec".
- 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`.

Handwritten annotations include a red arrow pointing from the assembly instruction at address 0x00000014 to the R1 register, and a green arrow pointing from the source code line `10: | LSRS r1,r1,r3` to the CPSR register. A yellow box highlights the instruction at address 0x00000014, and a green box highlights the source code line `10: | LSRS r1,r1,r3`.

# 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 by the value in the 'C' (Carry) register. The result is shown as a sequence of bits: 0000 0000 0001 1001 1001 1001 1001 1001. The final bit '1' is highlighted in red.
- Callout Box:** A blue cloud-shaped box with the text "Press Step, or F11".
- Memory Window:** Shows memory address 0x00000040 with 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 Keil uVision4 IDE with the following components:

- Registers Window:** Shows the current state of registers. R1 is highlighted with a value of 0x00066666. R15 (PC) is at 0x00000018. CPSR is at 0x000000D3.
- Disassembly Window:** Shows the assembly instructions for the current address. The instruction at 0x00000018 is highlighted in yellow: `0x00000018 E1B01141 MOVs R1,R1,ASR #2`. Other instructions include `LSRS r1,r1,#10`, `ASRS r1,r1,#2`, `LSLS r1,r1,#15`, and `ASRS r1,r1,#16`.
- Source Code Window:** Shows the assembly code for the current file. The instruction at line 12 is highlighted in green: `12 ASRS r1,r1,#2`. Other instructions include `9 LSRS r1,r1,#10`, `10 LSRS r1,r1,r3`, `13 LSLS r1,r1,#15`, `14 ASRS r1,r1,#16`, `16 ASRS r1,r1,r3`, `18 RORS r1,r1,#4`, and `19 RORS r1,r1,r3`.
- 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`.
- Command Window:** Shows the command `ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet`.

Annotations in the image include:

- A red arrow pointing from the assembly instruction `0x00000018 E1B01141 MOVs R1,R1,ASR #2` to the source code instruction `12 ASRS r1,r1,#2`.
- A green arrow pointing from the source code instruction `12 ASRS r1,r1,#2` to the assembly instruction `0x00000018 E1B01141 MOVs R1,R1,ASR #2`.
- A yellow box highlighting the assembly instruction `0x00000018 E1B01141 MOVs R1,R1,ASR #2`.
- A green box highlighting the source code instruction `12 ASRS r1,r1,#2`.
- A green box highlighting the memory address `0x00000040` and its value.

# ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:** Shows the current state of registers. R15 (PC) is at 0x00000018. CPSR is 0x000000D3. The C flag is 0.
- Disassembly Panel:** Shows the assembly code being executed. The instruction at address 0x00000018 is `ASRS r1,r1,#2`, which is highlighted in yellow.
- Source Code Panel:** Shows the corresponding assembly code in `asm_for_tutorial.asm`. Line 12 is `ASRS r1,r1,#2`, also highlighted in green.
- ASR Diagram:** A diagram in the top right shows the ASR instruction taking an operand and shifting it right by a specified amount (MSB). The result is stored in the C flag.
- Bit Patterns:** Two 32-bit bit patterns are shown, representing the state of the registers before and after the shift operation. The first pattern is `0000 0000 0000 0110 0110 0110 0110 0110` and the second is `0000 0000 0000 0001 1001 1001 1001 1001`.
- Callout:** A blue cloud-shaped callout bubble with the text "Press Step, or F11" is overlaid on the source code panel.

# ARM's Data-Processing Instructions (Shift Operations)

The screenshot displays the uVision4 IDE interface during a simulation. The **Registers** window on the left shows the current state of ARM registers. Register R1 contains the value 0x00019999, and the CPSR (Current Program Status Register) contains 0x200000D3, with the Carry flag (C) set to 1. The **Disassembly** window shows the instructions being executed, including several shift operations on register R1. The **ASM** window shows the corresponding assembly code. A yellow box highlights the bit pattern 0000 0000 0000 0001 1001 1001 1001 1001, which is the result of the LSL #15 instruction. A green arrow points from this bit pattern to the CPSR register, indicating the update of the Carry flag (C) to 1.

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 a blue box.
- Bit Patterns:**

Initial value of R1: 0000 0000 0000 0001 1001 1001 1001 1001

Result after LSL #15: 1100 1100 1100 1100 1000 0000 0000 0000
- Callout:** A blue cloud-shaped box with the text "Press Step, or F11".

# ARM's Data-Processing Instructions (Shift Operations)

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

- Registers Window:** Shows the current state of ARM registers. R1 contains the value 0xCCCC8000. 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 assembly code being executed. The instructions are:
  - 13: LSLS r1,r1,#15
  - 14: MOVS R1,R1,LSL #15
  - 15: ASRS r1,r1,#16
  - 16: MOVS R1,R1,ASR #16
  - 17: ASRS r1,r1,r3
  - 18: MOVS R1,R1,ASR R3
  - 19: RORS r1,r1,#4
  - 20: MOVS R1,R1,ROR #4
  - 21: RORS r1,r1,r3
  - 22: MOVS R1,R1,ROR R3
- Source Code Window:** Shows the assembly code for 'my\_First\_Example.s':
  - 11: ASRS r1,r1,#2
  - 12: LSLS r1,r1,#15
  - 13: ASRS r1,r1,#16
  - 14: ASRS r1,r1,r3
  - 15: RORS r1,r1,#4
  - 16: RORS r1,r1,r3
  - 17: 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 red circle highlights the '0' in the instruction 'ASRS r1,r1,#16' in the disassembly window. A green arrow points from this '0' to the 'C' flag in the CPSR register. A blue arrow points from the 'C' flag to the 'C' flag in the source code window. A yellow box highlights the binary representation of the R1 register 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 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 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 window is divided into several panes:

- Registers:** Located on the left, it shows the current values of registers R0 through R15 and the CPSR. R1 is highlighted with a blue background and contains the value 0xFFFFCCCC. R15 (PC) contains 0x00000024. CPSR contains 0xA00000D3. The CPSR flags are also shown: N=1, Z=0, C=1, V=0, I=1, F=1, T=0, M=0x13.
- Disassembly:** Located in the center, it shows the assembly code for the current file, 'asm\_for\_tutorial.asm'. The instructions are:
  - 13: LSLs r1, r1, #15
  - 14: ASRS r1, r1, #16
  - 15: MOVs R1, R1, ASR #16
  - 16: ASRS r1, r1, r3
  - 17: MOVs R1, R1, ASR R3
  - 18: RORS r1, r1, #4
  - 19: RORS r1, r1, r3
  - 20: MOVs R1, R1, ROR R3
  - 21: RORS r1, r1
  - 22: RORS r1, r1
  - 23: RORS r1, r1
- Registers (Current):** Located on the right, it shows the current values of registers R0 through R15 and the CPSR. R1 is highlighted with a blue background and contains the value 0xFFFFCCCC. R15 (PC) contains 0x00000024. CPSR contains 0xA00000D3. The CPSR flags are also shown: N=1, Z=0, C=1, V=0, I=1, F=1, T=0, M=0x13.
- Command:** Located at the bottom left, it shows the command 'ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet'.
- Memory:** Located at the bottom right, it shows the memory address 0x00000040 and the value 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.

Annotations in the image include:

- A red arrow pointing from the 'ASRS r1, r1, #16' instruction to the R1 register.
- A green arrow pointing from the 'ASRS r1, r1, r3' instruction to the R3 register.
- A yellow box highlighting the instruction 'ASRS r1, r1, r3' and the value '1111 1111 1111 1111 1100 1100 1100 1100' in the Disassembly window.
- A red circle with the number '1' next to the value '1100'.



# ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

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

```

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

```
- ASR Diagram:**

```

graph LR
    MSB[MSB] --> C[C]
    Operand[Operand] --> C
    ASR[ASR] --- MSB
    ASR --- Operand
    
```
- Binary Examples:**

Example 1: 1111 1111 1111 1111 1100 1100 1100 1100

Example 2: 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 Window:**

```

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet

```
- Memory Window:**

Address: 0

0x00000040: CC CC CC CC 00 00 00 00 00 00 00 00 00 00 00 00
- Simulation Status:**

Simulation t1: 0.00000000 sec L:16

**Press Step, or F11**

# 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 0xFFFF333. The CPSR (Current Program Status Register) shows the Carry flag (C) is 0.
- Disassembly Window:** Shows the assembly code being executed. Instruction 18, `RORS r1, r1, #4`, is highlighted in yellow. A red arrow points from the R1 register value to this instruction. A yellow box shows the binary result of the rotation: `1111 1111 1111 1111 1111 0011 0011 0011`. A green arrow points from the 'C' flag in the CPSR to this instruction.
- Assembly Window:** Shows the source code file `my_First_Example.s`. The corresponding assembly instructions are visible: `ASRS r1, r1, r3`, `RORS r1, r1, #4`, and `RORS r1, r1, r3`.
- 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 displays the uVision4 IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, editing, and debugging. The 'Registers' window on the left shows the current state of registers R0 through R15, with R15 (PC) at 0x00000028 and CPSR at 0x800000D3. The 'Disassembly' window shows the following assembly code:

```

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 'asm\_for\_tutorial.asm' window shows the source 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

```

A diagram in the top right corner illustrates the ROR (Rotate Right) operation. It shows an 'Operand' box with an arrow pointing to a 'C' (Carry) box. The arrow from the 'C' box loops back to the input of the 'Operand' box, indicating a rightward rotation.

Two hexadecimal values are shown in a yellow box, representing the result of the ROR operation:

```

1111 1111 1111 1111 1111 0011 0011 0011
0011 1111 1111 1111 1111 1111 0011 0011

```

A blue cloud-shaped callout with the text 'Press Step, or F11' is positioned over the assembly code.

The 'Command' window at the bottom shows the following commands:

```

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet

```

The 'Memory 1' 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.

The status bar at the bottom indicates 'Simulation' mode, with a time of 0.00000000 sec and a line number of L:18.

# ARM's Data-Processing Instructions (Shift Operations)

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

- Registers Window:** Shows the current state of registers. R1 is highlighted with a value of 0x3FFFFFF3. R15 (PC) is at 0x0000002C. CPSR is 0x000000D3. Other registers (R0-R14) are at 0x00000000.
- Disassembly Window:** Shows the assembly code being executed. The instruction at address 0x0000002C is `MOVs R1,R1,ROR R3`. The instruction at address 0x00000028 is `MOVs R1,R1,ROR #4`. The instruction at address 0x00000030 is `MOVs R1,R1,RRX`. The instruction at address 0x00000034 is `MOVs R1,R1,RRX`. The instruction at address 0x00000038 is `MOVs R1,R1,RRX`. The instruction at address 0x0000003C is `MOVs R1,R1,RRX`. The instruction at address 0x00000040 is `CCCCCCCC STCGTL p12,CR12,[R12],{204}`.
- Source 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`.

A binary representation of the value in R1 is shown as `0011 1111 1111 1111 1111 1111 0011 0011`. A green arrow points from the binary value to the R1 register in the Registers window. A blue arrow points from the R1 register to the `RORS r1,r1,r3` instruction in the Disassembly window. A green arrow points from the `RORS r1,r1,r3` instruction to the `RORS r1,r1,r3` instruction in the Source window. A green arrow points from the `RORS r1,r1,r3` instruction to the `RORS r1,r1,r3` instruction in the Source window.

0

# 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}
  
```
- Source Code Window (asm\_for\_tutorial.asm):**

```

15
16 ASRS r1,r1,r3
17
18 RORS r1,r1,#4
19 RORS r1,r1,r3
20
21 RRXS r1,r1
22 RRXS r1,r1
23 RRXS r1,r1
24 RRXS r1,r1
25 END
  
```
- Diagram:** A box labeled "ROR" contains a diagram showing an "Operand" being rotated right into a register "C".
- Hexadecimal Values:**
  - Initial value (R1): 0011 1111 1111 1111 1111 1111 0011 0011
  - Value after RORS r1,r1,r3: 1100 1111 1111 1111 1111 1111 1100 1100
- Callout:** A blue cloud-shaped box with the text "Press Step, or F11" and a circular arrow icon.



# ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers:** A list of registers (R0-R15, CPSR) with their current values. R1 is highlighted with a value of 0xCFFFFFFC.
- Disassembly:** A list of assembly instructions. The instruction at address 0x00000030 is highlighted in yellow: `RRXS r1, r1`. A red arrow points from the R1 register value to this instruction.
- asm\_for\_tutorial.asm:** A file containing assembly code. The instruction `RRXS r1, r1` at address 0x00000021 is highlighted in green. A green arrow points from this instruction to the `RRXS r1, r1` instruction at address 0x00000030.
- Binary Value:** A yellow box highlights the binary value `1100 1111 1111 1111 1111 1111 1100 1100`, which is the result of the shift operation. A red circle with the number 1 is next to the binary value.
- Command:** A text area for entering commands. The text `ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet` is visible.
- Memory 1:** A window showing memory addresses and values. The address 0x00000040 is shown with a value of `CC CC CC CC 00 00 00 00 00 00 00 00 00 00`.

# 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
<b>R1</b>	<b>0xE7FFFE6</b>
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
<b>R15 (PC)</b>	<b>0x00000034</b>
<b>CPSR</b>	<b>0x800000D3</b>
N	1
Z	0
<b>C</b>	<b>0</b>
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

Annotations in the image include a red arrow pointing from the R1 register value to the assembly code, a blue arrow pointing from the CPSR register value to the assembly code, and a green arrow pointing from the CPSR register value to the assembly code. A yellow box highlights the binary representation of the R1 value: 1110 0111 1111 1111 1111 1111 1111 0110. A small 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:** Shows the current state of registers. R15 (PC) is at 0x00000034. CPSR is 0x800000D3. The 'C' (Carry) flag is 0.
- Disassembly Window:** Shows assembly instructions. Line 22 is highlighted: `RRXS r1, r1`. The instruction list includes:
  - 0x00000028: `MOVNS R1, R1, ROR #4`
  - 19: `RORS r1, r1, r3`
  - 20: (blank)
  - 0x0000002C: `MOVNS R1, R1, ROR R3`
  - 21: `RRXS r1, r1`
  - 0x00000030: `MOVNS R1, R1, RRX`
  - 22: `RRXS r1, r1` (highlighted)
  - 0x00000034: `MOVNS R1, R1, RRX`
  - 23: `RRXS r1, r1`
  - 0x00000038: `MOVNS R1, R1, RRX`
  - 24: `RRXS r1, r1`
  - 0x0000003C: `MOVNS R1, R1, RRX`
  - 0x00000040: `CCCCCCCC STCGTL p12, CR12, [R12] {204}`
  - 0x00000044: `CCCCCCCC STCGTL p12, CR12, [R12] {204}`
- Source Code Window:** Shows the assembly file `asm_for_tutorial.asm` with instructions:
  - 15: (blank)
  - 16: `ASRS r1, r1, r3`
  - 17: (blank)
  - 18: `RORS r1, r1, #4`
  - 19: `RORS r1, r1, r3`
  - 20: (blank)
  - 21: `RRXS r1, r1`
  - 22: `RRXS r1, r1` (highlighted)
  - 23: `RRXS r1, r1`
  - 24: `RRXS r1, r1`
  - 25: `END`
- Diagram:** A red box labeled "Rotate right through carry" shows a flow from a "Register" to a "Carry" flag.
- Bit Patterns:** Two rows of 32-bit binary values are shown:
  - Row 1: 1110 0111 1111 1111 1111 1111 1111 1110 0110
  - Row 2: 0111 0011 1111 1111 1111 1111 1111 1111 0011
- Annotation:** A blue cloud 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:** A list of registers (R0-R15, CPSR) with their current values. R1 is highlighted with a value of 0x73FFFFFF.
- Disassembly:** A list of assembly instructions with their addresses and opcodes. The instruction at address 0x00000038 is highlighted in yellow: `MOV R1, R1, RRX`.
- asm\_for\_tutorial.asm:** A file containing assembly code. The instruction at address 0x00000016 is highlighted in green: `ASRS r1, r1, r3`. The instruction at address 0x00000018 is highlighted in green: `RORS r1, r1, #4`.
- Memory:** A window showing memory addresses and values. The address 0x00000040 is highlighted with a value of 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.

Annotations:

- A red arrow points from the R1 register value (0x73FFFFFF) to the MOV instruction at address 0x00000038.
- A blue arrow points from the ASRS instruction at address 0x00000016 to the R1 register.
- A green arrow points from the RORS instruction at address 0x00000018 to the R1 register.
- A yellow box highlights the binary value 0111 0011 1111 1111 1111 1111 1111 0011, which is the result of the ASRS instruction.
- A green circle with the number 0 is visible on the right side of the image.

# 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
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:23

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

```

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

```

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

Address: 0

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

A red arrow points from the R1 register value (0x39FFFFFF) to the assembly instruction 'RRXS r1, r1'. A green arrow points from the 'C' flag value (1) to the 'RRXS r1, r1' instruction. A yellow box highlights the binary value '0011 1001 1111 1111 1111 1111 1111 1001'.

1



# 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 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:24

**Press Step, or F11**

# ARM's Data-Processing Instructions (Shift Operations)

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

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

```

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

```
- 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 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 sequence '1001 1100 1111 1111 1111 1111 1111 1100' 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, =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”*

