Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C

Chapter 4
ARM Arithmetic and Logic Instructions
Exercises ANS

Z. Gu

Fall 2025

Acknowledgement: Lecture slides based on Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, University of Maine https://web.eece.maine.edu/~zhu/book/

ANDS

- What are value of r2, and NZCV flags after execution?

```
LDR r0, =0xFFFFFF00
LDR r1, =0x00000001
ANDS r2, r1, r0, LSL #1
```

- ANS: r2 = ?
- N = ?, Z = ?, C = ?,V = ?

ANDS ANS

- What are value of r2, and NZCV flags after execution?

```
LDR r0, =0xFFFFFF00
LDR r1, =0x00000001
ANDS r2, r1, r0, LSL #1
```

- ANS: r2 = 0x00000000
- N = 0, Z = 1, C = 1, V = "unchanged"
  - Left shift (LSL #1) moves the bits in r0 left by 1 bit. The bitwise AND is then computed: r2 = r1 & (r0 << 1).
  - N (Negative flag): Set to 0 because result r2 = 0x00000000, which is not negative.
  - Z (Zero flag): Set to 1 because the result is zero (r2 = 0).
  - C (Carry flag): Set to 1. The carry flag is updated based on the last bit shifted out during the left shift operation on r0. Since the left shift of 0xFFFFFF00 by 1 bit causes a '1' to be shifted out, the carry flag is set to 1.
  - V (Overflow flag): Unchanged by AND operation.

ADDS

- What are value of r2, and NZCV flags after execution?

```
LDR r0, =0xFFFFFF00
LDR r1, =0x00000001
ADDS r2, r1, r0, LSL #1
```

- ANS: r2 = ?
- N = ?, Z = ?, C = ?, V = ?

ADDS ANS

- What are value of r2, and NZCV flags after execution?

```
LDR r0, =0xFFFFFF00
LDR r1, =0x00000001
ADDS r2, r1, r0, LSL #1
```

- ANS: r2 = 0xFFFFFE01
- N = 1, Z = 0, C = 0, V = 0
  - Left shift (LSL #1) moves the bits in r0 left by 1 bit, so r0 = 0xFFFFFE00. The ADDS is then computed: r2 = r1 + (r0 << 1) = 0xFFFFFE01
  - N (Negative): Since result r2 = 0xFFFFFE01 when interpreted as signed 32-bit (two's complement) is negative (most significant bit is 1), N = 1.
  - Z (Zero): Result is not zero, Z = 0.
  - C (Carry): Carry is set if there is an unsigned overflow. Here carry flag C = 0.
  - V (Overflow): Overflow is set if there is a signed overflow. Here:
    - $r1$ is positive; $r0 << 1$ is negative since high bit is set; 2 operands have different signs, no overflow, so V = 0.
    - Recall "Overflow cannot occur when adding 2 operands with different signs or when subtracting 2 operands with the same sign."

Review

Barrel Shifter: Explanations

- LSL (logical shift left): shifts left, fills zeros on the right; C gets the last bit shifted out of bit 31. This is multiply by 2^n for non-overflowing values.
- LSR (logical shift right): shifts right, fills zeros on the left; C gets the last bit shifted out of bit 0. This is unsigned division by 2^n.
- ASR (arithmetic shift right): shifts right, fills the sign bit on the left to preserving the sign; C gets the last bit shifted out of bit 0. This is signed division by 2^n with sign extension
- ROR (rotate right): rotates bits right with wraparound; bits leaving bit 0 re-enter at bit 31, and C receives the bit that wrapped. This is a pure rotation without data loss.
- RRX (rotate right extended): rotates right by one through the carry flag, treating C as a 33rd bit; new bit 31 comes from old C, and C receives old bit 0.

Arithmetic with Shifts

- Assuimg 32-bit registers:
- Q1:
  - LDR r0, =0x00000007
  - MOV r0, r0, LSL 7
- Q2:
  - LDR r0, =0x00000400
  - MOV r0, r0, LSR 2
- Q3:
  - LDR r0, =0xFFFFC000
  - MOV r0, r0, LSR 2
- Q4:
  - LDR r0, =0xFFFFC000
  - MOV r0, r0, ASR 2
- Q5:
  - LDR r0, =0x00000007
  - MOV r0, r0, ROR 2

Q1 ANS

- Q1:
  - LDR r0, =0x00000007
  - MOV r0, r0, LSL 7
- ANS:
  - Original r0 = 0000 0000 0000 0000 0000 0000 0000 0111
  - After LSL 7, r0 = 0000 0000 0000 0000 0000 0011 1000 0000 = 0x0380 (896 in decimal)
  - In decimal: 7 × 2^7 = 7 × 128 = 896 (Not required for exam)

Q2 ANS

- Q2:
  - LDR r0, =0x00000400
  - MOV r0, r0, LSR 2
- ANS:
  - Original r0 = 0000 0000 0000 0000 0000 0100 0000 0000
  - After LSR 2, r0 = 0000 0000 0000 0000 0000 0001 0000 0000 = 0x00000100 (256 in decimal)
  - In decimal: 1024 ÷ 2^2 = 256 (Not required for exam)

Q3 ANS

- Q3:
  - LDR r0, =0xFFFFC000
  - MOV r0, r0, LSR 2
- ANS:
  - Original r0 = 1111 1111 1111 1111 1111 1100 0000 0000
  - After LSR 2, r0 = 0011 1111 1111 1111 1111 1111 0000 0000 = 0x3FFFF000
  - In decimal: 4,294,951,424 ÷ 4 = 1,073,737,728 (Not required for exam)

Q4 ANS

- Q3:
  - LDR r0, =0xFFFFC000
  - MOV r0, r0, ASR 2
- ANS:
  - Original r0 = 1111 1111 1111 1111 1111 1100 0000 0000
  - After ASR 2, r0 = 1111 1111 1111 1111 1111 1111 0000 0000 = 0xFFFFFF00 (-256 in decimal)
  - In decimal: -16384 ÷ 2^2 = -4096 (Not required for exam)

Q5 ANS

- Q4:
  - LDR r0, =0x00000007
  - MOV r0, r0, ROR 2
- ANS:
  - Original r0 = 0000 0000 0000 0000 0000 0000 0000 0111
  - After ROR r0 = 1100 0000 0000 0000 0000 0000 0000 0001 = 0xC0000001

Assembly Programming

- Write ARMv7 assembly for pseudocode
  - r1 = (r0 >> 4) & 15

Assembly Programming ANS

- Write ARMv7 assembly for pseudocode
  - r1 = (r0 >> 4) & 15
- ANS:
  - MOV r1, r0, LSR #4
  - AND r1, r1, #15