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

Chapter 4
ARM Arithmetic and Logic Instructions
Exercises

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/

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.

Review

Carry and Overflow Flags w/ Arithmetic Instructions

Carry flag C = 1 (Borrow flag = 0) upon an **unsigned** addition if the answer is wrong (true result > 2^n-1)

Carry flag C = 0 (Borrow flag = 1) upon an **unsigned** subtraction if the answer is wrong (true result < 0)

Overflow flag V =1 upon a **signed** addition or subtraction if the answer is wrong (true result > 2^{n-1}-1 or true result < -2^{n-1})

Overflow may occur when adding 2 operands with the same sign, or subtracting 2 operands with different signs; Overflow cannot occur when adding 2 operands with different signs or when subtracting 2 operands with the same sign.

Tip: Convert subtraction to addition with Two's complement. If two operands have same sign, and the result has opposite sign, then V = 1; else V = 0

	Unsigned Addition	Unsigned Subtraction	Signed Addition or Subtraction
Carry flag	true result > 2^n-1 ➔ Carry flag=1 Borrow flag=0 (Result incorrect)	true result < 0 ➔ Carry flag=0 Borrow flag=1 (Result incorrect)	N/A
Overflow flag	N/A	N/A	true result > 2^{n-1}-1 or true result < -2^{n-1} ➔ Overflow flag=1 (Result incorrect)

Bit Manipulations

- Compute register values after each instruction
- MOV R0, #0xABC
- MOV R1, #0xDEF
- AND R2, R0, R1
- ORR R3, R0, R1
- EOR R4, R0, R1
- ORN R5, R0, R1
- BIC r6, R0, R1

Bit Manipulations

- Find the Register Value to Complement, CLEAR & SET 5th, 7th, 12th bit of the given value and also find the result: 0xDECB.

Clearing a Register

- What are the different ways by which all bits in register r12 can be cleared? No other register is to be used.

Set bits

- Write an instruction that sets bits 0, 4, and 12 in register r6 and leave the remaining bits unchanged
- Write an instruction that clears bits 0, 4, and 12 in register r6 and leave the remaining bits unchanged

Add two 128-bit numbers

- Add two 128-bit numbers, assuming one number is stored in r4, r5, r6, r7 registers and the other stored in r8, r9, r10, r11. Store the result in r0, r1, r2, r3.

Absolute value

- Write a program to calculate the absolute value of a number by using only two instructions (HINT: Check CMP and RSB)

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

Assembly Programming

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

Shift LSL

- Compute register values:
  - LDR R1, =0X11223344
  - MOV R2, R1, LSL #4
  - MOV R3, R1, LSL #8
  - MOV R4, R1, LSL #16
  - MOV R5, R1, LSL #6

Shift ASR

- Compute register values:
- LDR R1, =0x81223344
- MOV R2, R1, ASR #4
- MOV R3, R1, ASR #8
- MOV R4, R1, ASR #16

Multiply without MUL

- Without using MUL instruction, give instructions that multiply a register, r3 by
  - 135
  - 153
  - 255
  - 18
  - 16384

Count number of ones

- Write a program to count the number of ones in a 32-bit register r0.

Count the number of zeros

- Based on the program that counts 1's, modify it to count the number of zeros a 32-bit register r0.

Flags ANDS

- What are value of r2, and NZCV flags after execution, assuming all flags are initially 0.

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

Flags ADDS

- What are value of r2, and NZCV flags after execution, assuming all flags are initially 0.

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

Flags

r0	0xffffffff
r1	0x00000001
r2	0x00000003
r3	0xfffffff0

- Suppose registers have the following values:
- What are value of r4, and NZCV flags after execution, assuming all flags are initially 0. (Each instruction runs individually.)
  - (a) ADD r4, r0, r2, ASR #3
  - (b) ADDS r4, r0, r1
  - (c) LSRS r4, r0, #1
  - (d) ANDS r4, r0, r3
  - (e) CMP r2, #3

r0	0xFFFFFFFF
r1	0x00000001
r2	0x00000000

Flags

- Suppose registers have the following values:
- What are value of r3, and NZCV flags after execution, assuming all flags are initially 0. (Assume each instruction runs individually, not sequentially.)
  - ADD r3, r0, r2
  - SUBS r3, r0, r0
  - ADDS r3, r0, r2
  - LSL r3, r0, #1
  - LSRS r3, r1, #1
  - ANDS r3, r0, r2