Section 3.1 — Arithmetic Instructions (Mazidi)

2025-09-01

Chapter 3 · Section 3.1 — Exercises (Mazidi)

> Problems are paraphrased to respect copyright. This section focuses on **ADDS/ADC** and the **C (carry) / Z (zero)** flags.

---

1) Find C and Z for each case. Also give the result and where it is saved.

(a)

```
MOV   R1,#0x3F
MOV   R2,#0x45
ADDS  R3,R1,R2
```

- **Computation:** 0x3F + 0x45 = 0x84
- **Result:** R3 = 0x00000084
- **Flags:** C=0, Z=0 (no carry out; result not zero).

(b)

```
LDR   R0, =0x95999999
LDR   R1, =0x94FFFF58
ADDS  R1, R1, R0
```

- **Computation:** 0x95999999 + 0x94FFFF58 = 0x12A9998F1 → low 32 bits 0x2A9998F1
- **Result:** R1 = 0x2A9998F1
- **Flags:** C=1 (carry out), Z=0.

(c)

```
LDR   R0, =0xFFFFFFFF
ADDS  R0, R0, #1
```

- **Computation:** 0xFFFFFFFF + 1 = 0x1_0000_0000 → low 32 bits 0x00000000
- **Result:** R0 = 0x00000000
- **Flags:** C=1, Z=1.

(d)

```
LDR   R2, =0x00000001
LDR   R1, =0xFFFFFFFF
ADD   R0, R1, R2     ; does NOT set flags
ADCS  R0, R0, #0     ; adds carry-in and sets flags
```

- After `ADD`: R0 = 0x00000000 (flags **unchanged**).
- `ADCS` uses the **previous C** (not set by the `ADD`). Assuming prior C=0 (typical unless set earlier):
  - **Result:** R0 = 0x00000000
  - **Flags set by ADCS:** C=0, Z=1.
    (If prior C=1, then R0=0x00000001 and Z=0.)

(e)

```
LDR   R0, =0xFFFFFFFE
ADDS  R0, R0, #2
ADC   R1, R0, #0      ; uses carry from ADDS; does not set flags
```

- **Computation:** 0xFFFFFFFE + 2 = 0x1_0000_0000 → R0 = 0x00000000
- **Flags after `ADDS`:** C=1, Z=1
- **Then `ADC`:** R1 = R0 + 0 + C = 0 + 0 + 1 = 0x00000001 (flags unchanged).

---

2) State the three steps in a subtraction (SUB) and apply them.

Three steps (A – B):

1. **One's complement** of B → ~B.
2. **Add 1** to form **two's complement** of B.
3. **Add** to A: A + (~B + 1). In ARM, the **C flag after subtraction** means: **C=1** → **no borrow**, **C=0** → **borrow**.

Apply to 8-bit examples (showing intermediate two's complement):

- **(a)** 0x23 – 0x12
  - ~0x12 = 0xED, +1 → 0xEE; 0x23 + 0xEE = 0x111 → result 0x11, **C=1** (no borrow).
- **(b)** 0x43 – 0x51
  - ~0x51 = 0xAE, +1 → 0xAF; 0x43 + 0xAF = 0xF2 → result 0xF2 (i.e., −0x0E in 8-bit), **C=0** (borrow occurred).
- **(c)** 0x99 – 0x39
  - ~0x39 = 0xC6, +1 → 0xC7; 0x99 + 0xC7 = 0x160 → result 0x60, **C=1** (no borrow).

Notes for learners

- `ADD` vs `ADDS`: only forms with `S` update flags.
- **`ADC`/`ADCS`** add the **carry-in**; `ADCS` also **updates** flags.
- In ARM subtraction, remember: **C = NOT borrow**.