Chapter 4 · Section 4.1 — Exercises (Mazidi)

Chapter 4 · Section 4.1 — Exercises (Mazidi)

> Problems are paraphrased to respect copyright. When helpful, results are shown in **decimal** and **hex**.

---

1) In ARM, looping action using a single register is limited to ______ iterations.

Answer: 4,294,967,296 iterations (2^32).
Why: A loop counter can be held in one 32-bit register and decremented with `SUBS ..., #1` and `BNE` until zero.

---

2) If a conditional branch is not taken, what instruction executes next?

Answer: The **next sequential (fall-through)** instruction (i.e., the one at `PC+4` in ARM state).

---

3) In calculating the branch target, a displacement is added to register ______.

Answer: `PC` (R15) — branches are **PC-relative**.

---

4) The mnemonic BNE stands for ________.

Answer: Branch if Not Equal (i.e., `Z == 0`).

---

5) What is the advantage of using BX over B?

Answer: `BX` **branches to an address in a register and can switch instruction set state** (ARM↔□Thumb based on bit0), whereas `B` is PC-relative and does not change state.

---

6) True or False. The target of a BNE can be anywhere in the 4 GB address space.

Answer: False. The PC-relative range of ARM `B{{cond}}` is limited (see Q8).

---

7) True or False. All ARM branch instructions can branch to anywhere in the 4 GB byte space.

Answer: False. Branch ranges are **finite** (PC-relative immediates).

---

8) Dissect the B instruction: how many bits are for the operand vs. the opcode, and how far can it branch?

Answer: In ARM state, `B{{cond}}` uses **24 bits** for the **signed immediate operand** (`imm24`), and **8 bits** for the **opcode/condition** (`cond[31:28]` + `101` + `L`). The target is `PC + sign_extend(imm24 << 2)`, so the range is approximately **±32MB** (±2^25 bytes).

---

9) True or False. All conditional branches are 2-byte instructions.

Answer: False. In **ARM (A32)** they are **4 bytes**; only **Thumb** has 16-bit conditional branches.

---

10) Show code for a nested loop that performs an action 10,000,000,000 times.

```
        ; Outer = 10,000  (0x2710), Inner = 1,000,000 (0x0F4240)
        AREA    |.text|, CODE, READONLY
        EXPORT  _start
        THUMB
_start:
        LDR     r2, =0x00002710      ; outer count = 10,000
Outer:
        LDR     r1, =0x000F4240      ; inner count = 1,000,000
Inner:
        ; ---- ACTION HERE (one time per inner iteration) ----
        NOP                          ; replace with your code
        ; ----------------------------------------------------
        SUBS    r1, r1, #1
        BNE     Inner                ; run inner exactly 1,000,000 times
        SUBS    r2, r2, #1
        BNE     Outer                ; repeat outer 10,000 times
        B       .
        END
```

Total iterations: `10,000 × 1,000,000 = 10,000,000,000`.

---

11) Show code for a nested loop that performs an action 200,000,000,000 times.

```
        ; Outer = 20,000 (0x4E20), Inner = 10,000,000 (0x00989680)
        AREA    |.text|, CODE, READONLY
        EXPORT  _start
        THUMB
_start:
        LDR     r2, =0x00004E20      ; outer = 20,000
Outer2:
        LDR     r1, =0x00989680      ; inner = 10,000,000
Inner2:
        ; ---- ACTION HERE ----
        NOP
        ; ---------------------
        SUBS    r1, r1, #1
        BNE     Inner2
        SUBS    r2, r2, #1
        BNE     Outer2
        B       .
        END
```

Total iterations: `20,000 × 10,000,000 = 200,000,000,000`.

---

12) How many times is the loop body executed?

```
MOV   R0,#0x55
MOV   R2,#40
L1:   LDR   R1,=10000000      ; ten million per outer pass
L2:   EOR   R0,R0,#0xFF       ; loop body (the "action")
      SUB   R1,R1,#1
      BNE   L2
      SUB   R2,R2,#1
      BNE   L1
```

Answer: 400,000,000 times (`40 × 10,000,000`).

---

13) Status of Z and C after `CMP`

Recall: `CMP Rn,Op2` computes `Rn - Op2`.

- **Z = 1** if equal.
- **C = 1** if **no borrow** (i.e., `Rn ≥ Op2` as **unsigned**).
- **(a)** `R0=0x32`, `R1=0x28` → `0x32 - 0x28` → **Z=0, C=1**.
- **(b)** `R1=0xFF`, `R2=0x6F` → **Z=0, C=1**.
- **(c)** `R2=0x34`, `R3=0x88` → **Z=0, C=0**.
- **(d)** `R1=0`, `R2=0` → **Z=1, C=1**.
- **(e)** `R2=0`, `R3=0xFF` → **Z=0, C=0**.

- **(f)** `R0=0`, `R1=0` → **Z=1, C=1**.
- **(g)** `R4=0x78`, `R2=0x40` → **Z=0, C=1**.
- **(h)** `R0=0xAA & 0x55 = 0x00`, compare with `#0` → **Z=1, C=1**.

14) Rewrite "Program 4-1" to find the lowest grade

Assume an array of **N unsigned bytes** at `GRADES`, result in `R2`.

```
        AREA    |.text|, CODE, READONLY
        EXPORT  find_min
        THUMB
GRADES  EQU     0x20000000
N       EQU     40

find_min:
        LDR     r0, =GRADES          ; r0 = base
        LDR     r1, =N               ; r1 = count
        LDRB    r2, [r0], #1         ; r2 = current minimum (first element)
        SUBS    r1, r1, #1           ; remaining

loop_min:
        CBZ     r1, done
        LDRB    r3, [r0], #1
        CMP     r3, r2               ; if r3 < r2 update min
        BHS     skip                 ; BHS: r3 >= r2 (unsigned) → keep old min
        MOV     r2, r3               ; new min
skip:   SUBS    r1, r1, #1
        BNE     loop_min
done:   BX      lr
        END
```

15) The target of a BNE is backward if the relative offset is ______.

Answer: negative (sign-extended `imm24 << 2` is < 0).

16) The target of a BNE is forward if the relative offset is ______.

Answer: positive.

Notes for learners

- `B{{cond}}` targets are **PC-relative**; the assembler converts labels to signed offsets.
- For **very long jumps**, use an absolute branch via a register: `LDR rX,=dest ; BX rX`.
- Flag meanings for `CMP`: think **unsigned** for **C** (borrow/no-borrow) and **equality** for **Z**.