Chapter 6 · Section 6.4 — Exercises (Mazidi)

Chapter 6 · Section 6.4 — Exercises (Mazidi)

> Problems are paraphrased to respect copyright. Explanations are kept short and practical.

---

47) True or False — Write-back is by default enabled in pre-indexed addressing mode.

Answer: False. In pre-indexed form write-back happens **only** when you add `!` (e.g., `[Rn, Rm]!`). Without `!` it's an **offset** access (no write-back).

---

48) Indicate the addressing mode

- **(a)** `LDR R1,[R5],R2, LSL #2` → **Post-indexed**, register offset with shift (address = `[R5]`, then `R5 += (R2<<2)`).
- **(b)** `STR R2,[R1,R0]` → **Offset / pre-indexed without write-back** (effective addr = `R1 + R0`, `R1` unchanged).
- **(c)** `STR R2,[R1,R0, LSL #2]!` → **Pre-indexed with write-back** (addr = `R1 + (R0<<2)`, then `R1` updated).
- **(d)** `STR R9,[R1],R0` → **Post-indexed** with register offset (store at `[R1]`, then `R1 += R0`).

---

49) What is an ascending stack?

A stack that **grows toward higher addresses** as items are pushed; **SP increases** on push.

---

50) Difference between an empty and a full stack

- **Full stack:** `SP` **points to the last occupied** location. A push **writes before** moving away (with DB or IB depending on direction).
- **Empty stack:** `SP` **points to the next free** location. A push **writes at SP** then moves (with IA or DA depending on direction).

---

51) Store `R0` in a full descending stack

```
PUSH    {{R0}}                  ; alias for STMDB SP!,{R0}  (FD: pre-decrement, store)
; equivalently:
STMDB   SP!, {{R0}}
```

---

52) Load `R9` from an empty descending stack

```
LDMIB   SP!, {{R9}}             ; ED: increment-before on pop, SP increases
; (For comparison: POP {R9} == LDMIA SP!,{R9} is for a full-descending stack.)
```

Notes for learners

- Mapping between stack names and addressing modes (store/push first):
  FD ↔ STMDB / LDMIA, **FA ↔ STMIB / LDMDA**, **ED ↔ STMDA / LDMIB**, **EA ↔ STMIA / LDMDB**.