

Section 2.2 — The ARM Memory Map (Mazidi)

Chapter 2 · Section 2.2 — Exercises (Mazidi)

Problems are paraphrased to respect copyright. For theory and examples, see **Mazidi, Ch. 2 §2.2**.

11) True or False. R13 and R14 are special function registers.

Answer: True.

Why: In the programmer's model, R13 = SP (stack pointer) and R14 = LR (link register) — both are **special-purpose** registers.

12) True or False. The peripheral registers are mapped to memory space.

Answer: True.

Why: ARM microcontrollers use **memory-mapped I/O**; peripheral registers occupy regions in the **address space**.

13) True or False. The on-chip Flash is the same size in all members of ARM.

Answer: False.

Why: Flash size **varies by device/family** (e.g., 8KB ... MBs).

14) True or False. The on-chip data SRAM is the same size in all members of ARM.

Answer: False.

Why: SRAM size is **device-dependent**.

15) What is the difference between the EEPROM and data SRAM space in the ARM?

Answer: EEPROM is **non-volatile** (retains contents with power off, slower writes, limited endurance); SRAM is **volatile** (contents lost on power-off, fast read/write).

16) Can we have an ARM chip with no EEPROM?

Answer: Yes.

Why: Many ARM MCUs have **no true EEPROM**; Flash (or emulated EEPROM) is used instead.

17) Can we have an ARM chip with no data RAM?

Answer: No.

Why: Practical execution requires **RAM** for stack/variables.

18) What is the maximum number of bytes that the ARM can access?

Answer: 4GB = 2^{32} bytes = 0x00000000–0xFFFFFFFF (architectural 32-bit address space; specific MCUs may implement less).

19) Find the address of the last location of on-chip Flash for each case (first location = 0).

- (a) 32KB → last = 0x7FFF
- (b) 8KB → last = 0x1FFF
- (c) 64KB → last = 0xFFFF
- (d) 16KB → last = 0x3FFF
- (e) 128KB → last = 0x1FFFF

- (f) **256KB** \rightarrow last = **0x3FFFF**
Reasoning: last address = **size** $- 1$ (bytes).
-

20) Show the lowest and highest values (in hex) that the ARM program counter can take.

Answer: Lowest = **0x00000000**, Highest = **0xFFFFFFFF**.

Note: Alignment/state bits may constrain actual fetch addresses, but the architectural range is as above.

21) A given ARM has 0x7FFF as the last location of its on-chip ROM. What is the size?

Answer: **0x8000 bytes = 32KB**.

Why: Size = last $-$ first $+ 1 = 0x7FFF - 0x0000 + 1$.

22) Repeat Question 21 for 0x3FFF.

Answer: **0x4000 bytes = 16KB**.

23) Find the on-chip program memory size (in KB) for these address ranges (inclusive):

- (a) **0x0000–0x1FFF** \rightarrow size = **0x2000 = 8KB**
 - (b) **0x0000–0x3FFF** \rightarrow size = **0x4000 = 16KB**
 - (c) **0x0000–0x7FFF** \rightarrow size = **0x8000 = 32KB**
 - (d) **0x0000–0xFFFF** \rightarrow size = **0x10000 = 64KB**
 - (e) **0x0000–0x1FFFF** \rightarrow size = **0x20000 = 128KB**
 - (f) **0x0000–0x3FFFF** \rightarrow size = **0x40000 = 256KB**
-

24) Find the on-chip program memory size (in KB) for these address ranges (inclusive):

- (a) **0x000000–0xFFFFFFFF** \rightarrow size = **0x1000000 = 16MB = 16384KB**
 - (b) **0x000000–0x7FFFFF** \rightarrow size = **0x80000 = 512KB**
 - (c) **0x000000–0x7FFFFFFF** \rightarrow size = **0x800000 = 8MB = 8192KB**
 - (d) **0x000000–0x0FFFFFFF** \rightarrow size = **0x100000 = 1MB = 1024KB**
 - (e) **0x000000–0x1FFFFFFF** \rightarrow size = **0x200000 = 2MB = 2048KB**
 - (f) **0x000000–0x3FFFFFFF** \rightarrow size = **0x400000 = 4MB = 4096KB**
-

Notes for learners

- **Inclusive ranges:** If the first address is 0, the last address is (**size** $- 1$).
- Use powers of two: **1KB = 1024 bytes**, **1MB = 1024KB**.
- Program counter range shown is the architectural 32-bit span; concrete devices map only a subset.