Chapter 6 · Section 6.1 — Exercises (Mazidi)

Chapter 6 · Section 6.1 — Exercises (Mazidi)

> Problems are paraphrased to respect copyright. Where useful, I show short teaching notes and the arithmetic.

---

1) What is the bus bandwidth unit?

Answer: bytes per second (often expressed as **MB/s**).

---

2) Give the variables that affect bus bandwidth.

Answer:

- **Data-bus width** (bytes per transfer).
- **Bus clock frequency** (transfers per second).
- **Cycles per transfer** (wait states, turnaround, arbitration).
- Optional efficiency factors: **burst length**, **cache/hit ratio**, etc.

Rule of thumb: `Bandwidth = (bus_width_bytes × bus_clock) / (cycles_per_transfer).`

---

3) True/False — One way to increase bus bandwidth is to widen the data bus.

Answer: True. Wider bus ⇒ more bytes moved per cycle.

---

4) True/False — Increasing the number of address-bus pins raises bus bandwidth.

Answer: False. A wider address bus increases the **addressable space**, not the transfer rate.

---

5) Calculate memory-bus bandwidth

Assume a **32-bit data bus (4 bytes)** and that **`cycles per transfer = 1 + wait_states`**.

- **(a)** 100MHz, **0 WS** → transfers/sec = 100M / 1 = **100M**.
  Bandwidth = 4 × 100M = **400MB/s**.
- **(b)** 80MHz, **1 WS** → transfers/sec = 80M / 2 = **40M**.
  Bandwidth = 4 × 40M = **160MB/s**.

---

6) Indicate which addresses are word aligned (address % 4 = 0)

address	aligned?	reason
(a) `0x1200004A`	☐	ends **A** (not 0/4/8/C)
(b) `0x52000068`	☐	ends **8**
(c) `0x66000082`	☐	ends **2**
(d) `0x23FFFF86`	☐	ends **6**
(e) `0x23FFFFF0`	☐	ends **0**
(f) `0x4200004F`	☐	ends **F**
(g) `0x18000014`	☐	ends **4**
(h) `0x43FFFFF3`	☐	ends **3**
(i) `0x44FFFF05`	☐	ends **5**

---

7) Show how data is placed (little- vs big-endian)

```
LDR   R2, =0xFA98E322
LDR   R1, =0x20000100
STR   [R1], R2
```

- **Little-endian** (LSB at lowest address):
  `0x20000100: 0x22`, `0x20000101: 0xE3`, `0x20000102: 0x98`, `0x20000103: 0xFA`.
- **Big-endian** (MSB at lowest address):
  `0x20000100: 0xFA`, `0x20000101: 0x98`, `0x20000102: 0xE3`, `0x20000103: 0x22`.

---

8) True/False — In ARM, instructions are always word aligned.

Answer: False (as a general statement). In **ARM state** they are word-aligned, but in **Thumb** state instructions are **halfword-aligned**.

---

9) True/False — In a word-aligned address the lower hex digit is 0, 4, 8, or C.

Answer: True. (Those correspond to the low two bits `00`.)

---

10)–14) Memory cycles required (32-bit bus)

Assumption: A 64-bit `LDRD` on a 32-bit bus reads **one 32-bit word per cycle**.

- If the starting address is **word-aligned**: **2 cycles**.
- Otherwise the 8-byte window crosses **three** 32-bit words: **3 cycles**.
- Halfword (`LDRH`): **1 cycle** when halfword-aligned (address % 2 = 0).
- Byte (`LDRB`): **1 cycle** at any alignment.

10)

```
LDR   R1, =0x20000004
LDRD  [R1], R2
```

Start **aligned ⇒ 2 cycles**.

11)

```
LDR   R1, =0x20000102
LDRD  [R1], R2
```

Start **unaligned (…02)**; 8 bytes span three words **⇒ 3 cycles**.

12)

```
LDR   R1, =0x20000103
LDRD  [R1], R2
```

Start **unaligned (…03) ⇒ 3 cycles**.

13)

```
LDR   R1, =0x20000006
LDRH  [R1], R2
```

Halfword **aligned (…06) ⇒ 1 cycle**.

14)

```
LDR   R1, =0x20000C10
LDRB  [R1], R2
```

Byte access (any alignment) ⇒ **1 cycle**.

Notes for learners

- **Alignment faults** may occur on some cores for unaligned word/halfword accesses; others handle them in hardware but need extra cycles.
- Effective bandwidth is lowered by **wait states** and **unaligned** accesses—align your data when you can.