Section 2.5 — ARM Data Format and Directives (Mazidi)

Chapter 2 · Section 2.5 — Exercises (Mazidi)

> Problems are paraphrased to respect copyright. For theory and examples, see **Mazidi, Ch. 2 §2.5**.

---

38) State the hex value for each of the following EQU constants

Name	Definition	Value (hex)
MYDAT_1	EQU 55	0x37
MYDAT_2	EQU 98	0x62
MYDAT_3	EQU 'G'	0x47
MYDAT_4	EQU 0x50	0x50
MYDAT_5	EQU 200	0xC8
MYDAT_6	EQU 'A'	0x41
MYDAT_7	EQU 0xAA	0xAA
MYDAT_8	EQU 255	0xFF
MYDAT_9	EQU 2_10010000	0x90
MYDAT_10	EQU 2_01111110	0x7E
MYDAT_11	EQU 10	0x0A
MYDAT_12	EQU 15	0x0F

Notes: `2_` denotes **binary**; character constants (e.g., `'G'`) use ASCII.

---

39) State the hex value for each of the following EQU constants

Name	Definition	Value (hex)
DAT_1	EQU 22	0x16
DAT_2	EQU 0x56	0x56
DAT_3	EQU 2_10011001	0x99
DAT_4	EQU 32	0x20
DAT_5	EQU 0xF6	0xF6
DAT_6	EQU 2_11111011	0xFB

---

40) Show a simple code to load the value `0x10102265` into locations `0x40000030–0x4000003F`.

Approach: That range is **16 bytes**, i.e., **four words** at `0x30, 0x34, 0x38, 0x3C`. Use `STR` (word store) with a 4-iteration loop.

```
        AREA    |.text|, CODE, READONLY
        EXPORT  _start
        THUMB
_start:
        LDR     r0, =0x40000030          ; start (word-aligned)
        LDR     r1, =0x10102265          ; value to store
        MOVS    r2, #4                   ; four words = 16 bytes

store_loop40:
        STR     r1, [r0]                 ; *(uint32_t*)r0 = 0x10102265
        ADDS    r0, r0, #4               ; next word address
        SUBS    r2, r2, #1
        BNE     store_loop40

        B       .
        END
```

Explanation: Using `STR` avoids byte-by-byte stores and respects word alignment.

---

41) (a) Load the value `0x23456789` into locations `0x40000060–0x4000006F`, and (b) add them together, placing the result in `R9` as values are added. Use `EQU` to name the locations `TEMP0–TEMP3`.

Approach: Define the four word addresses with `EQU`. Store the word at each address and accumulate the sum in `R9`.

```
        AREA    |.text|, CODE, READONLY
        EXPORT  _start
        THUMB

TEMP0   EQU     0x40000060
TEMP1   EQU     0x40000064
TEMP2   EQU     0x40000068
TEMP3   EQU     0x4000006C

_start:
        LDR     r1, =0x23456789          ; word to replicate
        MOVS    r9, #0                   ; accumulator = 0

        ; -- store to TEMP0..TEMP3 and accumulate --
        LDR     r0, =TEMP0
        STR     r1, [r0]
        ADDS    r9, r9, r1

        LDR     r0, =TEMP1
        STR     r1, [r0]
        ADDS    r9, r9, r1

        LDR     r0, =TEMP2
        STR     r1, [r0]
        ADDS    r9, r9, r1

        LDR     r0, =TEMP3
        STR     r1, [r0]
        ADDS    r9, r9, r1

        ; Now r9 = 4 * 0x23456789 = 0x8D159E24 (mod 2^32)
        B       .
        END
```

Explanation: The range `0x60–0x6F` covers **four words** (16 bytes). Each store is word-aligned. The final sum is `0x8D159E24` (no wrap in 32-bit math).

Notes for learners

- `EQU` defines a **symbolic constant**; it does **not** allocate memory.
- Bases: `0x..` = hex, `2_..` = binary, decimal is default.
- For aligned word ranges, prefer **`STR`** with a **4-byte stride**; for byte ranges use **`STRB`**.