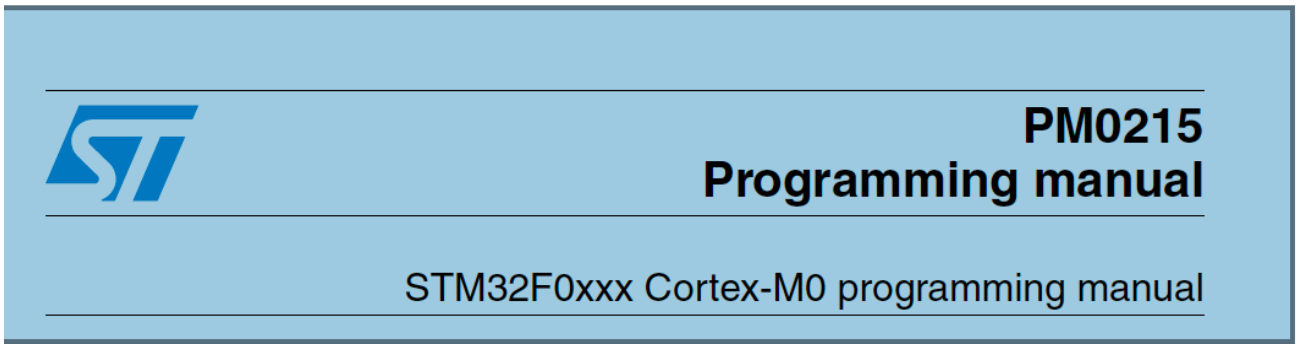
Ch.2 Instruction Set Architecture

Instruction Set Architecture (ISA) Topics :

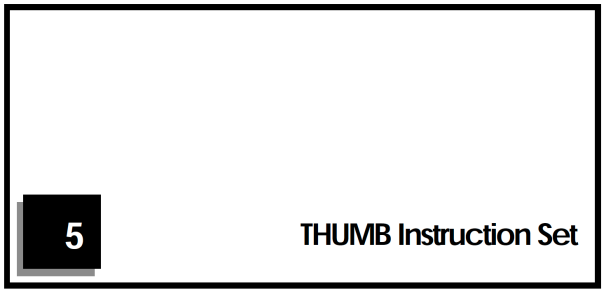
- Data representation: Big- and Little-Endians
- Instructions representation: Opcode (operation) and Operand locations
- Machine instructions
- Program execution: Instruction cycle: fetch and execution
- 2's complement arithmetic and logic operation
- Stacks
- Subroutines

References

- PM: Cortex_M0 programming manual (PM-page#)

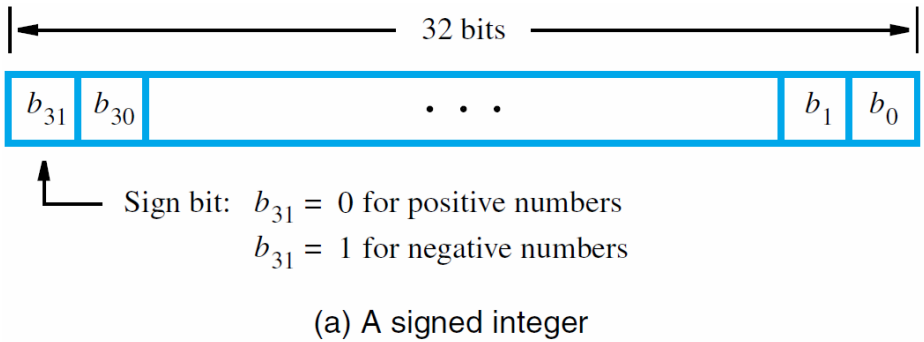


- TH: ARM7-TDMI-manual, Ch5 Thumb Instruction Set (TH-Sec#)



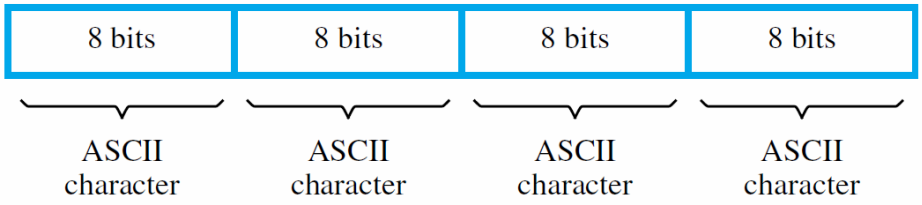
2.1 Data Encoding

- 32-bit word examples, Fig. 2.2:
  - A 32-bit integer (a)



(a) A signed integer

  - Four 8-bit bytes (b)
  - (e.g., ASCII characters with b_7=0)



(b) Four characters

2.1 Memory Organization

- **Binary Structure:**
- **Cell:** one bit 0 or 1
- **Nibble:** 4 bits
- **Byte:** 8 bits
- **Word:** n bits
  - **Word length (*n*):** 8, 16, 32, 64, or 128 bits

What is an *n*-bit processor?

↓

Handles data (operands) of *n* bits

Fig. 8.2 Memory Chip Organization

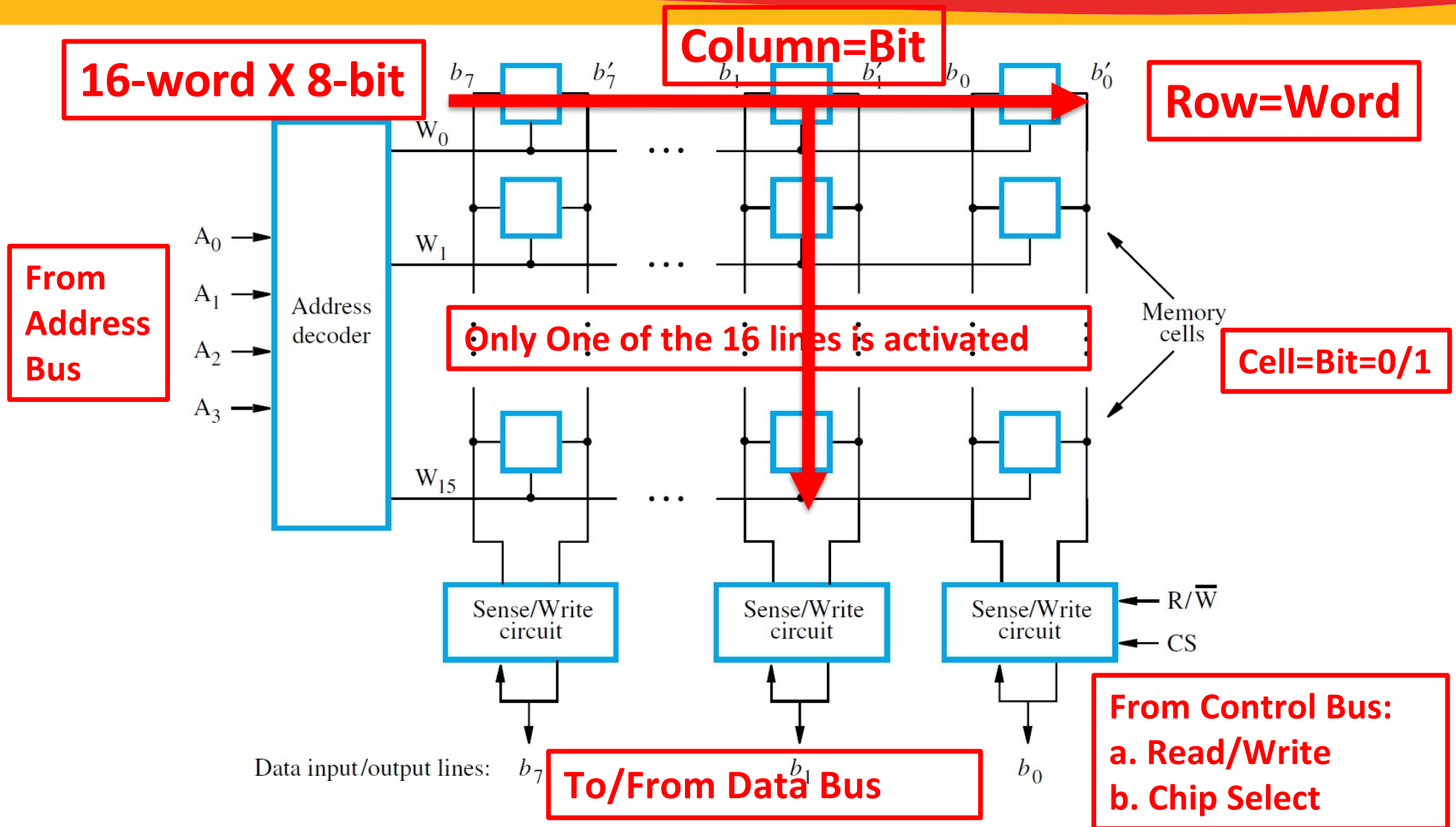
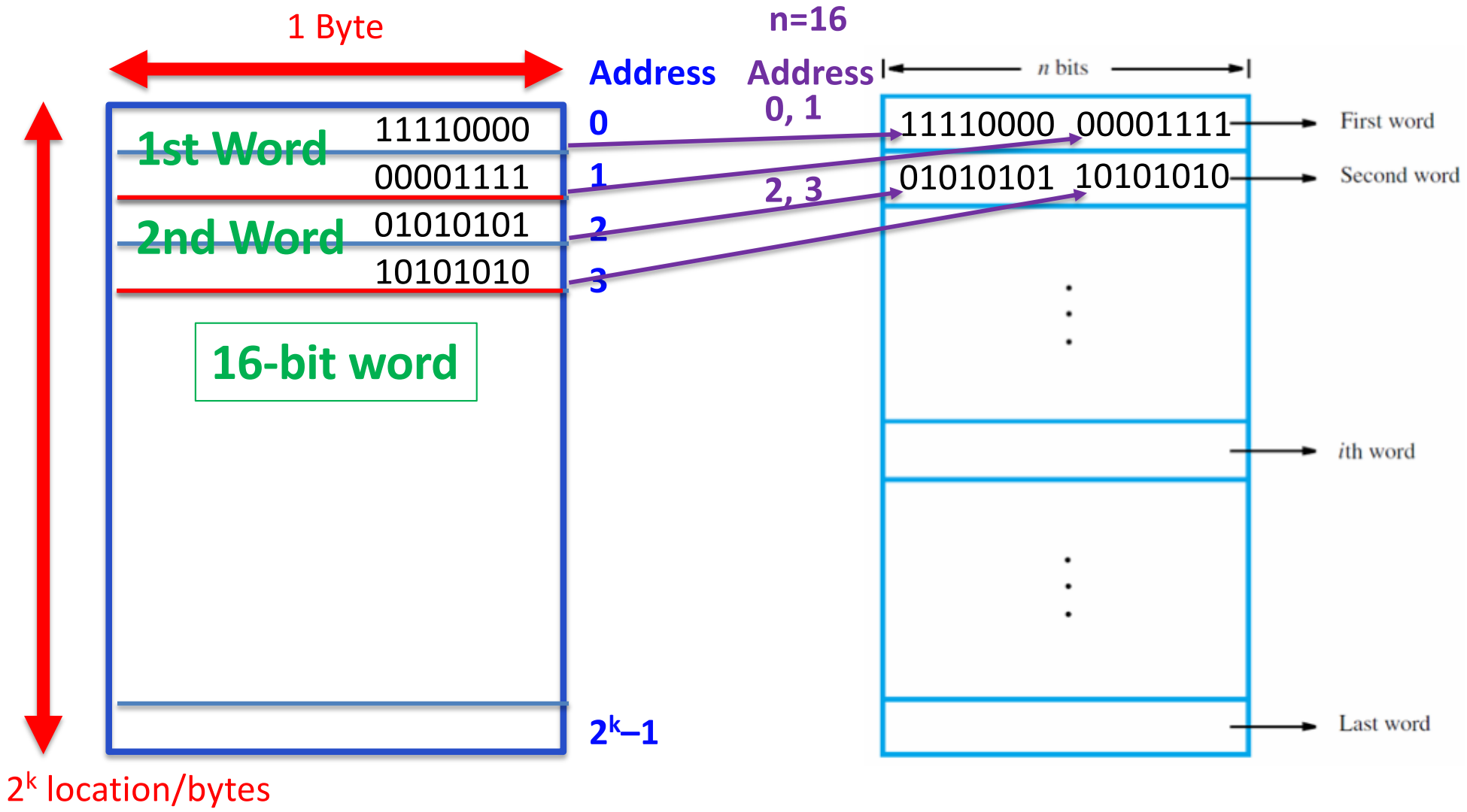


Fig. 2.1 Memory Layout



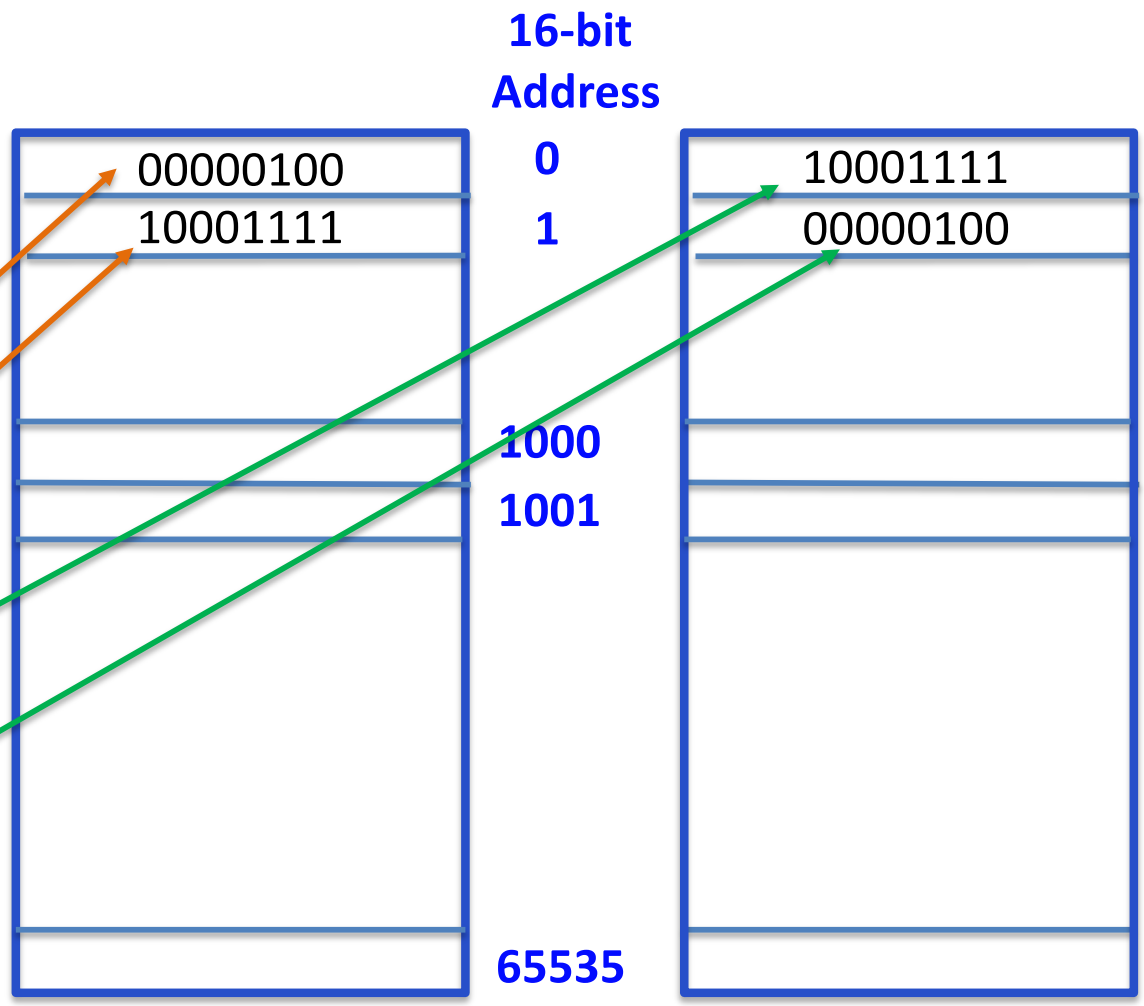
Byte Placement

- If data 0x048F (00000100 10001111) is to be placed at locations 0000 and 0001

- Location 0000 stores 04 and 0001 stores 8F ?

or

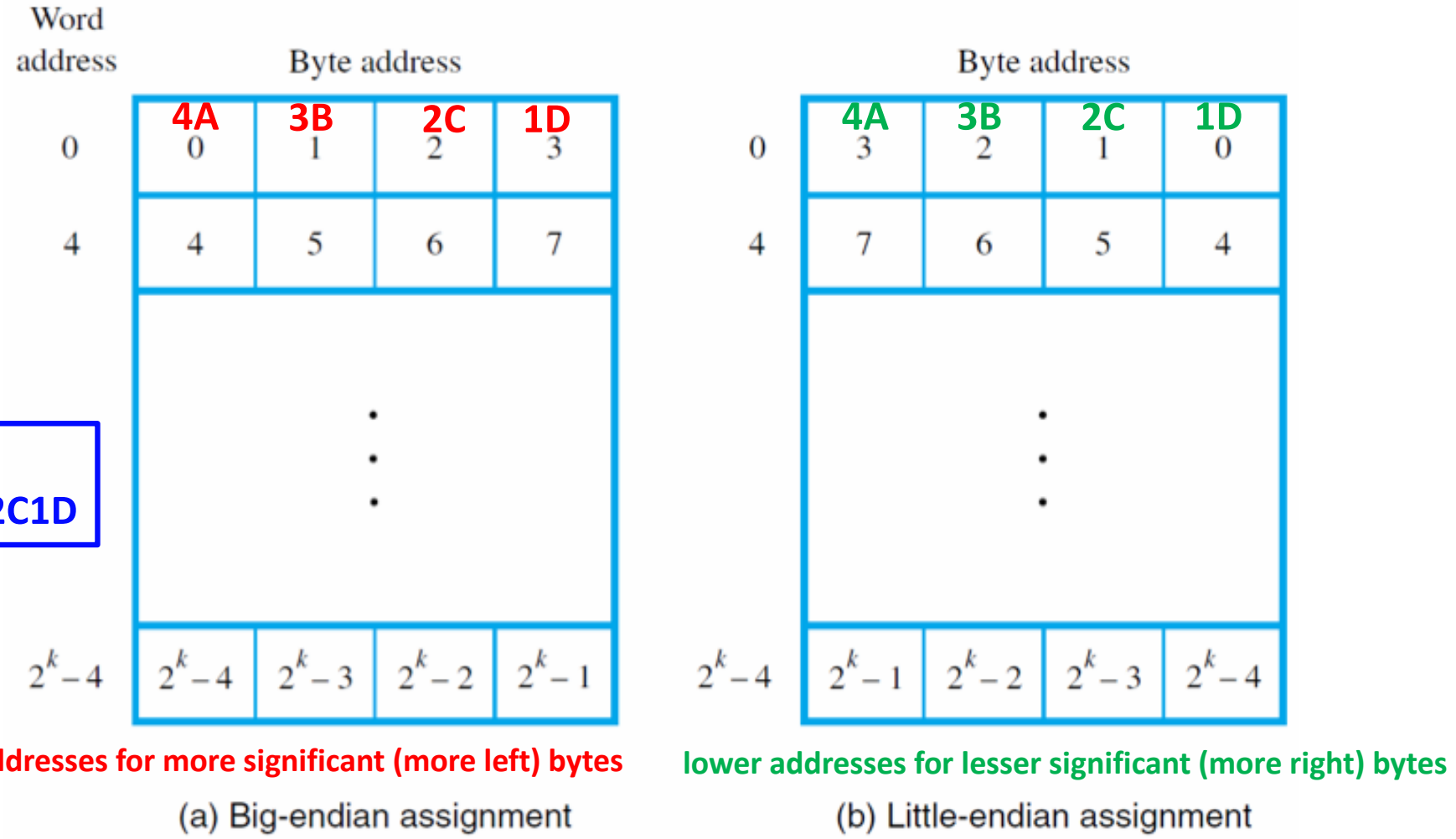
- Location 0000 stores 8F and 0001 stores 04 ?



2.1.2 Big- and Little-Endian

- Bits in each byte labeled b_7 … b_0, high to low, left to right,
  Most Significant Bit to Least Significant Bit
  - Example 0x048F: b_{15} … b_8 b_7 … b_0, high to low, left to right
    - Most Significant Byte in 0x048F : 0x04 or $0000 0100
    - Least Significant Byte in 0x048F : 0x8F or $1000 1111
- **Big-endian:** lower addresses store more significant (leftmost) bytes of data (04 in 0x048F)
- **Little-endian** : lower addresses store lesser significant bytes (8F in 0x048F)

Fig. 2.3 Big- and Little-Endian Details



Data= 0x4A3B2C1D

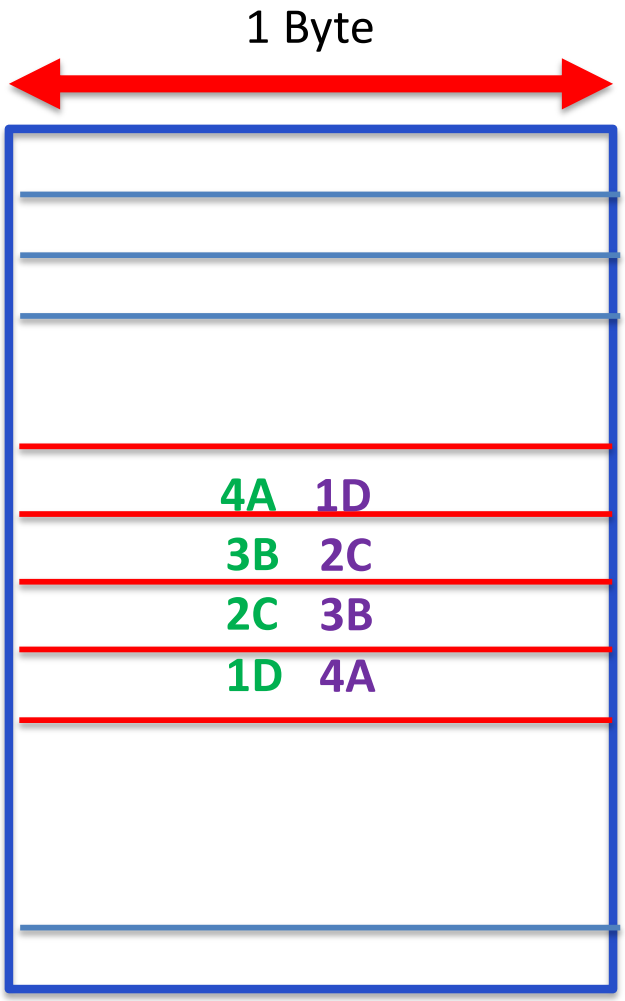
lower addresses for more significant (more left) bytes

(a) Big-endian assignment

lower addresses for lesser significant (more right) bytes

(b) Little-endian assignment

Big Endian



Data=
0x4A3B2C1D
To location 100

2^k location/bytes

Big-endian assigns lower addresses to store more significant (more left) bytes

Address
0
1
.
.
100
101
102
103

Little-endian assigns lower addresses to store lesser significant (more right) bytes

ARM Little Endian (PM21)

Little-endian format

In little-endian format, the processor stores the least significant byte (lsbyte) of a word at the lowest-numbered byte, and the most significant byte (msbyte) at the highest-numbered byte. See *Figure 7* for an example.

Figure 7. Little-endian example