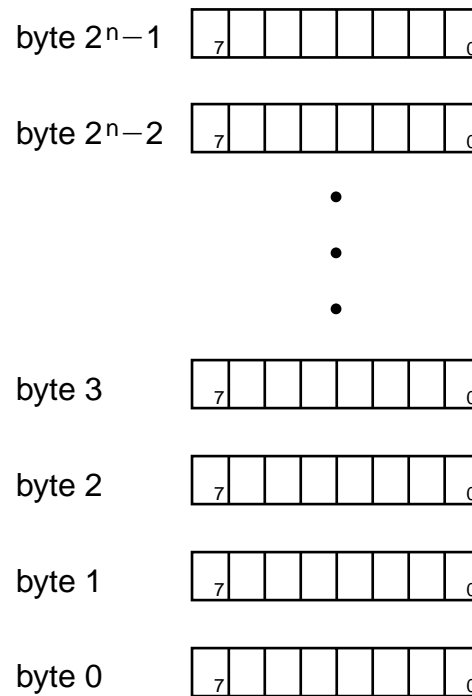


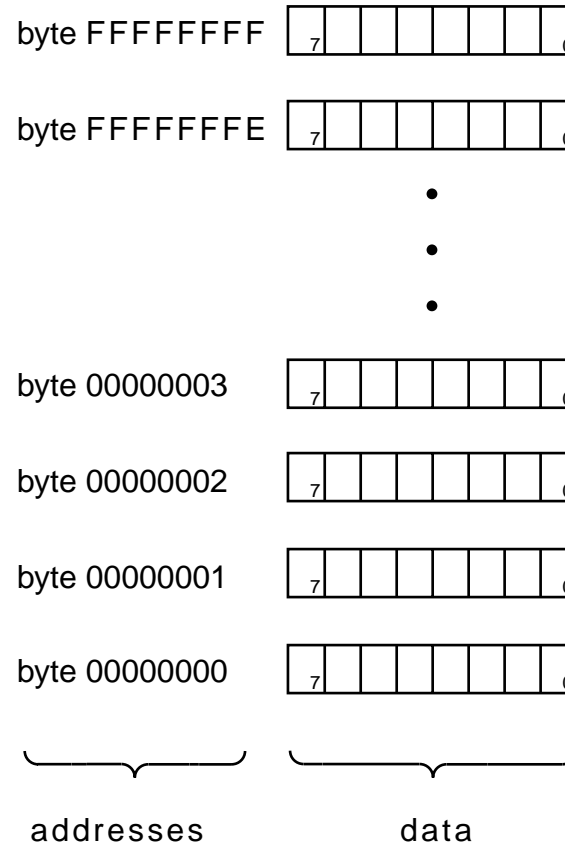
## MEMORY ADDRESSING

- Logical structure of a computer's random-access memory (RAM)
  - ▷ The generic term for the smallest unit of memory that the CPU can read or write is **cell**
    - In most modern computers, the size of a cell is 8 bits (1 byte)
  - ▷ Hardware-accessible units of memory larger than one cell are called **words**
    - Currently (1998) the most common word sizes are 32 bits (4 bytes) and 64 bits (8 bytes)
  - ▷ Every memory cell has a unique integer **address**
    - The CPU accesses a cell by giving its address
    - Addresses of logically adjacent cells differ by 1
    - The **address space** of a processor is the range of possible integer addresses, typically  $(0 : 2^n - 1)$

# BYTE-ADDRESSED MEMORY



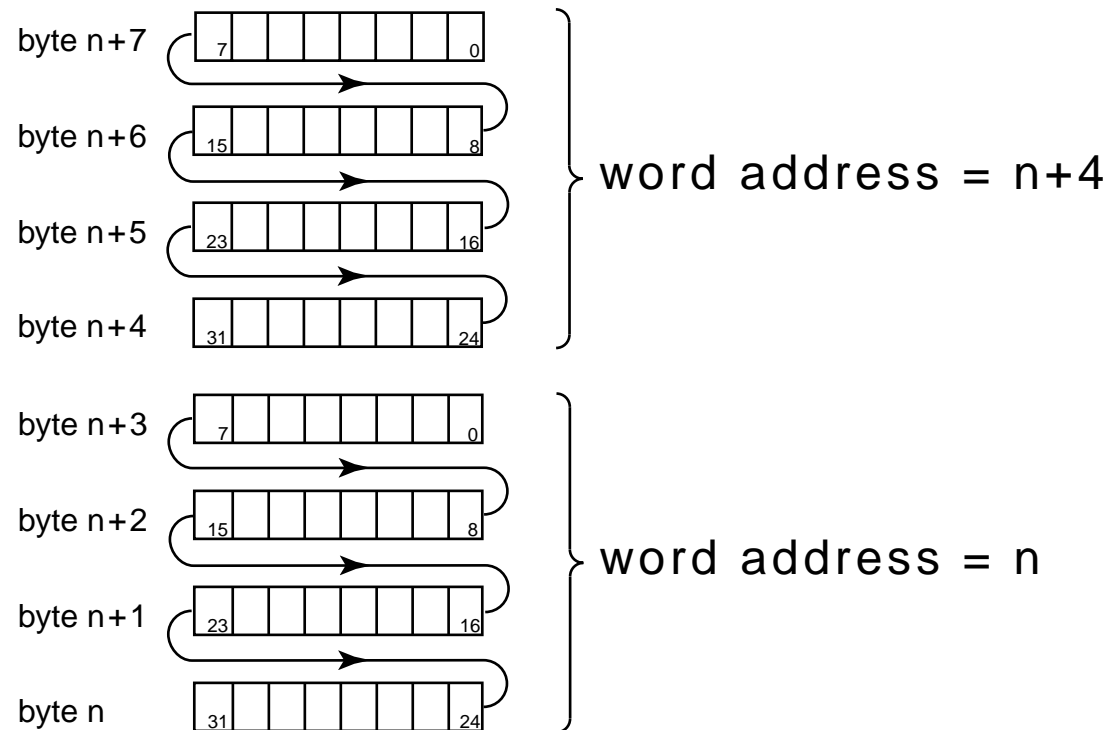
# BYTE-ADDRESSED MEMORY



32-bit addressing: The address of a byte is a 4-byte (32-bit) word

# WORD ADDRESSING

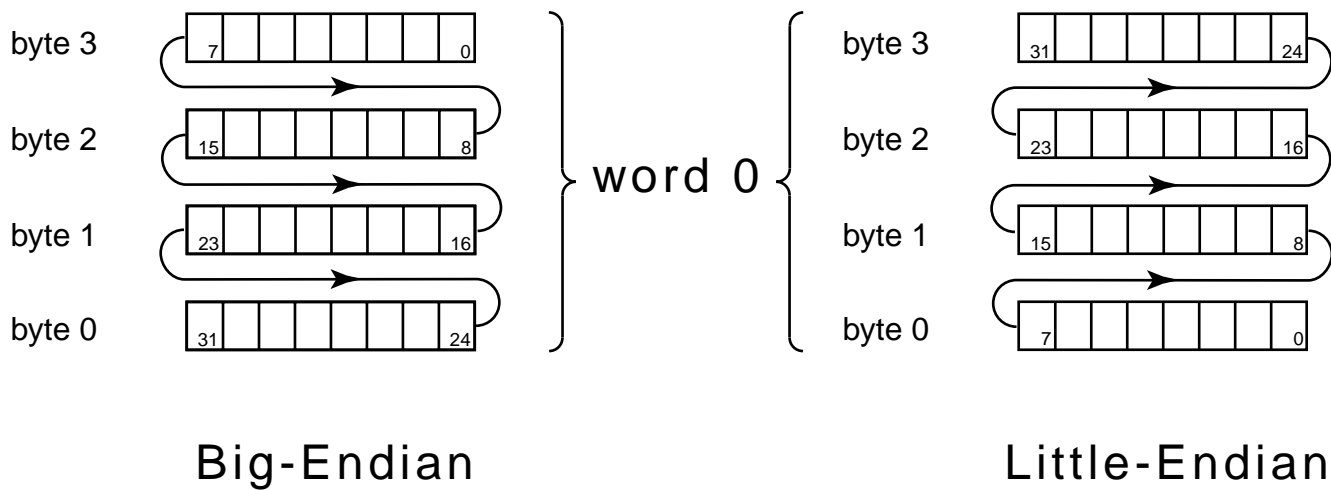
- In a byte-addressed memory, the addresses of successive words differ by the number of bytes in a word:



## BYTE ORDERING

- Big-endian byte ordering
  - ▷ **Most** significant (leftmost) byte has the lowest address
  - ▷ The address of a word is the address of its most significant byte
  - ▷ Default byte ordering in MIPS, DEC Alpha, HP PA-RISC and IBM/Motorola/Apple PowerPC architectures
  - ▷ Only available byte ordering in SPARC and IBM 370 architectures
- Little-endian byte ordering
  - ▷ **Least** significant (rightmost) byte has the lowest address
  - ▷ The address of a word is the address of its least significant byte
  - ▷ Only available byte ordering in Intel 80x86, National Semiconductor NS 32000 and DEC Vax architectures

# BYTE ORDERING CONVENTIONS



The arrows point in the direction of increasingly significant digits

**LITTLE-ENDIAN vs BIG-ENDIAN BYTE ORDERING**

- Affects the *interpretation* of multi-byte structures (4-byte words, etc.)
- Examples:

▷ Strings:

$$\begin{aligned} \text{"MIPS"} &= 4D\ 49\ 50\ 53 \quad (\text{big-endian}) \\ &= 53\ 50\ 49\ 4D \quad (\text{little-endian}) \end{aligned}$$

But

$$53\ 50\ 49\ 4D = \text{"SPIM"} \quad (\text{big-endian})$$

▷ Unsigned 32-bit integer (such as an IP address):

$$81\ 6E\ 10\ 53_{16} = 2,171,474,003_{10} \quad (\text{big-endian})$$

But

$$53\ 10\ 6E\ 81_{16} = 1,393,585,793_{10} \quad (\text{little-endian})$$

- A problem for data transfer from one device to another!

## ALIGNMENT

- In RISC ISAs, the address of the low-address byte of a block of memory that holds a data type must be a multiple of the data type's size
  - ▷ The address of a byte can be any unsigned integer within the processor's address space
  - ▷ Assume that a word is 4 bytes (32 bits)
    - The address of a word (4 bytes) must be a multiple of 4
    - A word address ends with 2 zero bits (00)
    - Possible last hexadecimal digits in a word address: 0, 4, 8, C
  - ▷ The address of a doubleword (8 bytes) must be a multiple of 8
    - A doubleword address ends with 3 zero bits (000)
    - Possible last hexadecimal digits in a word address: 0, 8
- The MIPS directive `.align n` aligns block addresses on multiples of `n`