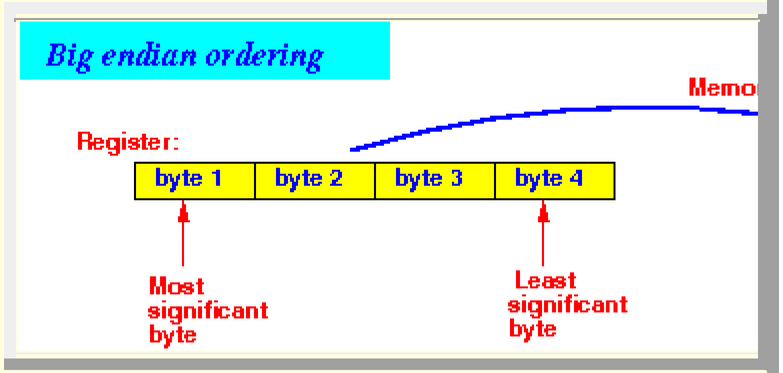
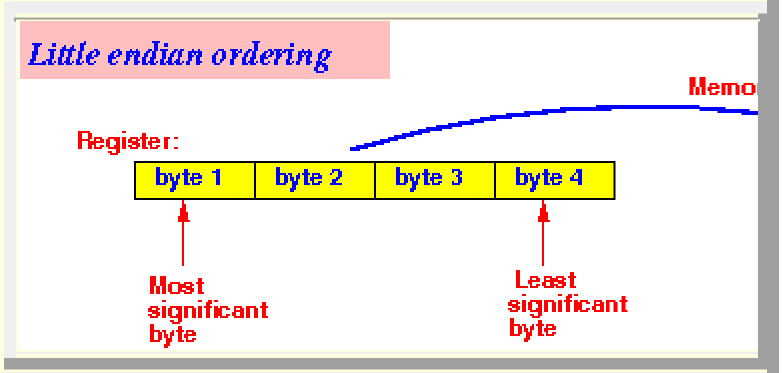
Store-ordering in memory: big endian and littel endian

- **Store ordering**
  - When **more than 1 bytes** in a **register** is **stored in memory**, there are **2 store orderings** possible:

    1. **Big endian ordering** where the **most-significant byte** is stored ***first***:

    2.
    3. **Little endian ordering** where the **least-significant byte** is stored ***first***:

  -
  -
  - The **endianness** in use is determined by the **CPU type**.

    For example:

- The **INTEL processors (x86)** are **little endian**
- The **ARM processor** is **little endian** by default; and can be **programmed** to operate as **big endian**
- Many older processors were **big endian**, such as: **Motorola M68000** and **SPARC**

Endianness is initially an arbitrary decision by the semiconductor vendor that can have a long-term impact on a line of products.

When vendors update their technology, they keep the existing endianness to help maintain backward compatibility.

For example, the designers of the Motorola 68000 and the Intel 8086 (predecessor of the x86 family) chose their endianness in the 1970s and continue to use their respective endianness today.

- **Which ordering is better ?**
  - The **ordering** has **absolutely** no effect of **processor performance** and any other performance factor.

    - As long as the **processor** retrieve the **bytes stored in the memory** in the ***same* ordering** as the processor **stores the bytes**, the **`load`** and **`store`** instructions will operate **correctly** and with the ***same* efficiency**

In fact, the term **Big endian** and **Little endian** is taken from the book **Gulliver's travels** and describes a **meaningless squabble** between 2 faction in the nation **Lilliput**:

- The Big-Endians, who broke their boiled eggs at the big end, rebelled against the king, who demanded that his subjects break their eggs at the little end.

The people in Lilliput were very tiny (compared) to Gulliver and their eggs is even **smaller**.

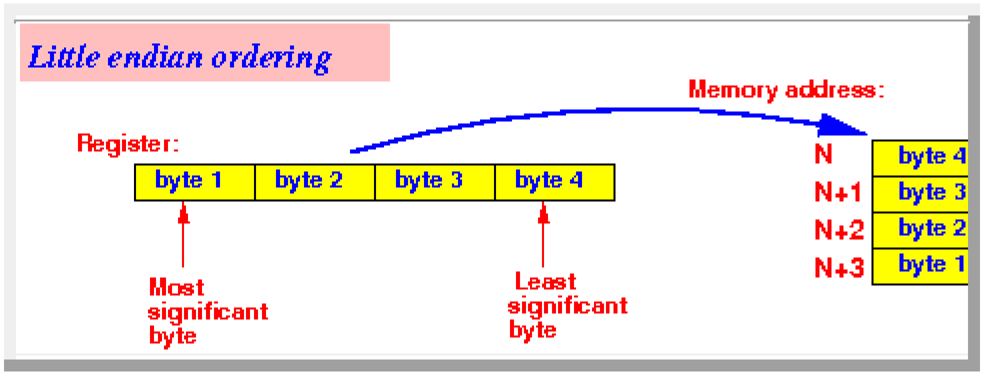
Gulliver can't tell which end was the big end and which was the little end. So to Gulliver, this squabble was meaningless...

So it does not matter which store-ordering a processor uses...

See: click here

-
-
-
-
- **ARM's processor store ordering**
  - The **ARM processor** uses **little endian** (by default)

    (We did not change the default.)



    That's why you had to **reverse** the **byte ordering** in this webpage: click here