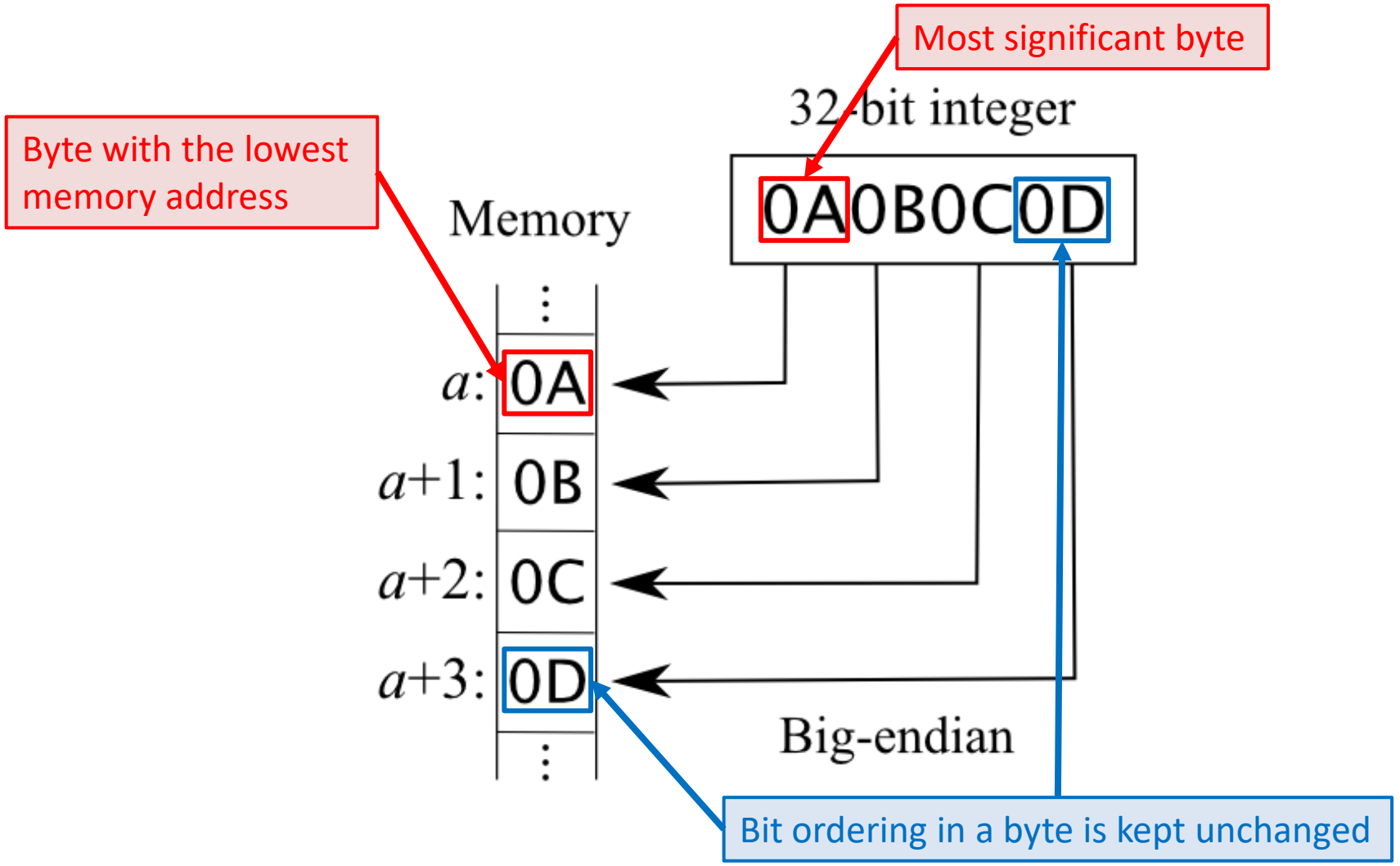
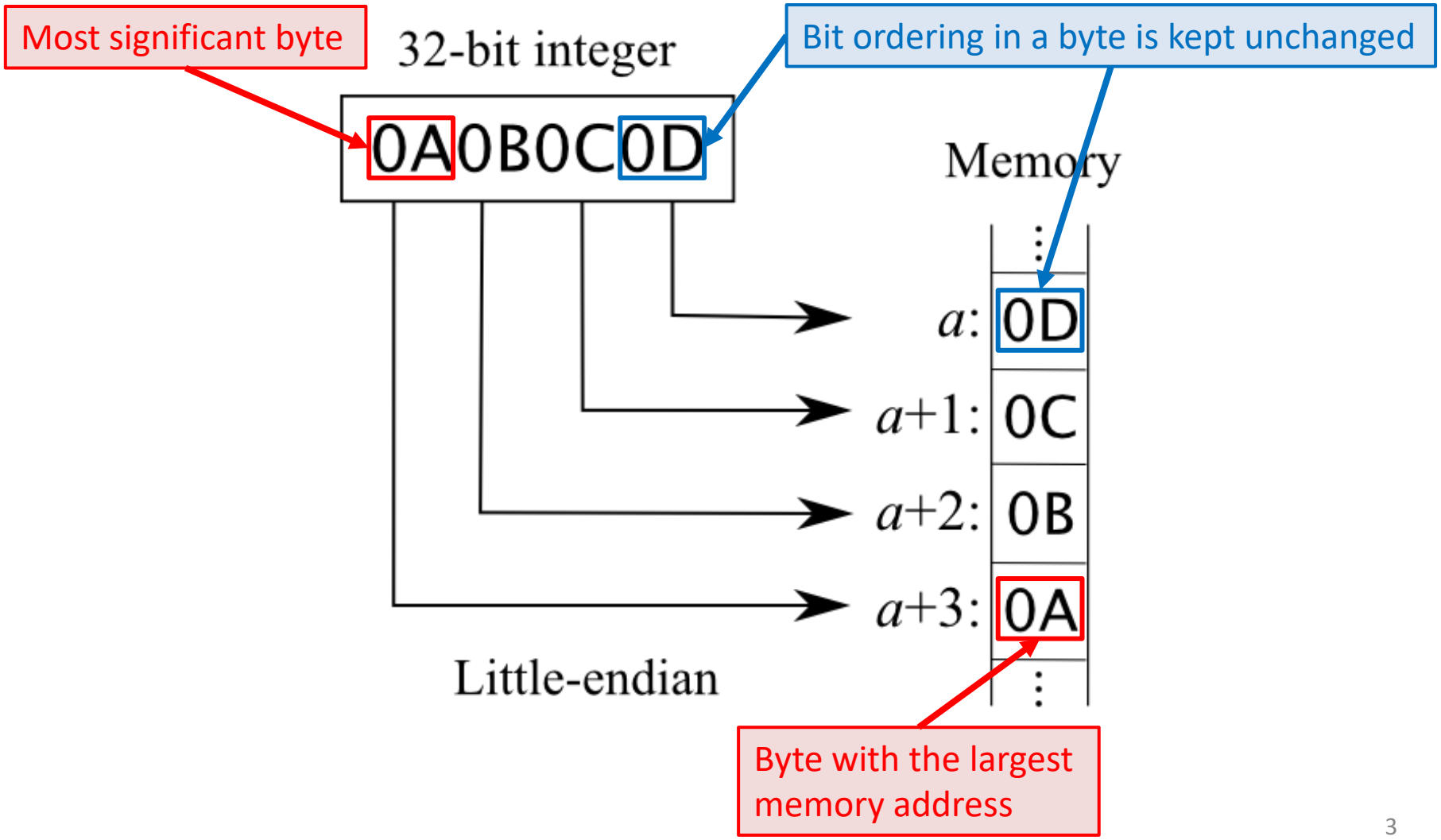
Endianness

- A 16bit or 32bit integer may have different byte arrangement under different **HARDWARE**
- Big-Endian (IBM System 360, Motorola 6800, etc)
  - **LOWEST** memory address stores **MOST** significant **BYTE**
- Little-Endian (Intel CPU, DEC Alpha, Atmel AVR , etc)
  - **LARGEST** memory address stores **MOST** significant **BYTE**
- Network order of UDP and TCP **IS ALWAYS** big-endian
- **Within a byte**, most significant byte **IS ALWAYS** located at the far right hand side in C Programming.

Big-Endian


Most significant byte
32-bit integer
0A0B0C0D
Byte with the lowest memory address
Memory
a: 0A
a+1: 0B
a+2: 0C
a+3: 0D
Big-endian
Bit ordering in a byte is kept unchanged

Little-Endian



Why Endianness matters

- In µTCP header, the first 4 bit is the mode of packet and the remaining 28 bit is the sequence number.
- Since sequence number in bound by $2^{28} - 1$, the 4 most significant bit must be zero.
- The most significant byte of the sequence number should have a lowest memory address such that the originally empty 4 most significant bit is at the beginning of the header and can hold the mode code.
- Sequence number is in big-endian (network order).

Encoding a µTCP header

Given:
unsigned int seq (sequence number)
unsigned char mode (mode code)
unsigned char buffer[4] (header byte array)

1. Convert sequence number to network order
seq = htonl(seq);

2. Copy sequence number to buffer
memcpy(buffer, &seq, 4);

3. Add mode code to the first byte of the buffer
buffer[0] = buffer[0] | (mode << 4);

Decoding a µTCP header

Given:
unsigned int seq (sequence number)
unsigned char mode (mode code)
unsigned char buffer[4] (header byte array)

1. Get the mode number
mode = buffer[0] >> 4;

2. Remove the mode code from 1st byte of header
buffer[0] = buffer[0] & 0x0F;

3. Copy sequence number to buffer
memcpy(&seq, buffer, 4);

4. Convert sequence number to local order
seq = ntohl(seq);