Endianness Tutorial

Endianness can be a confusing topic when you first learn it. This tutorial should help to illustrate that it's really quite simple once you wrap your head around it.

Cells

Recall that memory is made of cells, which are the smallest addressable units in a computer. For instance, in a byte-addressable system, memory is divided into cells one byte in length. When we want to read a value from memory, we must read at least one byte – we cannot read, for instance, half of a byte. This is what we mean when we say that a cell is the smallest addressable unit.

Endianness

The endianness of a system refers to the order in which our data is stored in the memory cells of the computer. Suppose that we store a value in memory. In a little-endian system, the low order byte of the value is stored at a lower address, while in a big-endian system, the low-order byte is stored at a higher address.

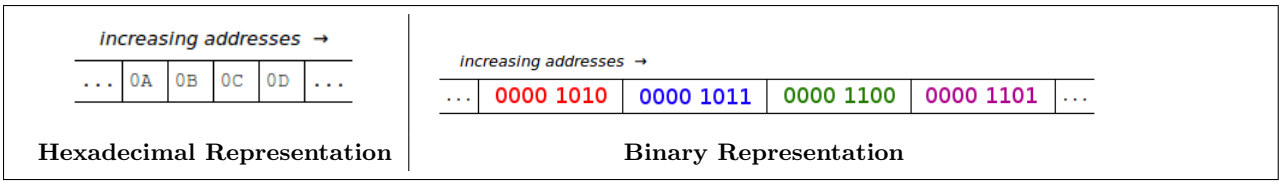
What does this mean? Let's use an example to illustrate the idea. Suppose we are using a byte-addressable computer and we wish to store the following 4-byte (32-bit) value in memory (shown both in binary and hexadecimal):

Base 2	0000 1010	0000 1011	0000 1100	0000 1101
Base 16	0A	0B	0C	0D

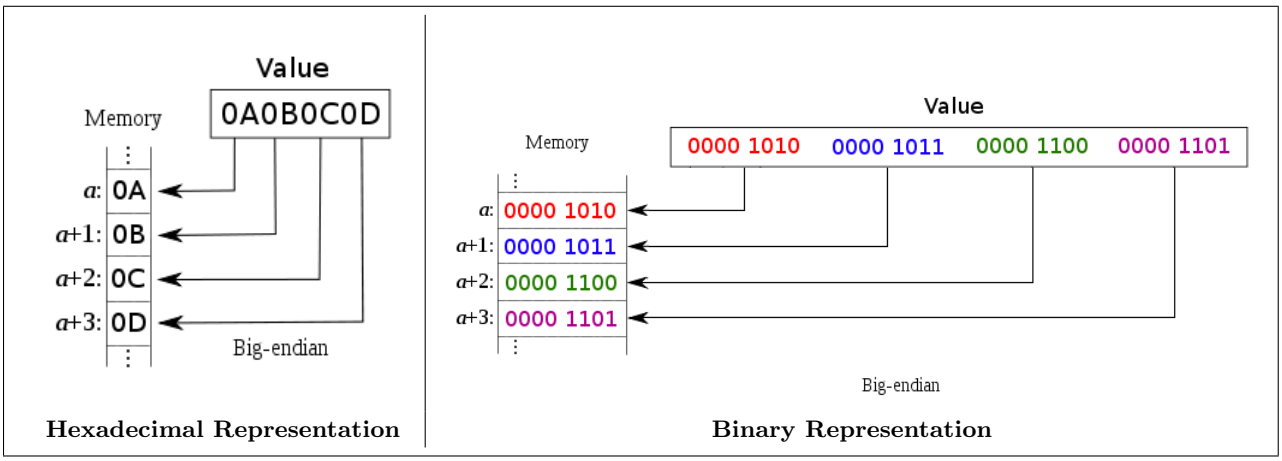
Let's see how this value would be stored in memory in a big-endian system first, followed by a little-endian system. For each system, we will look at how the number is stored in memory using both hexadecimal and binary to represent the number.

Big-Endian System

When we store the value $0A\ 0B\ 0C\ 0D$ in memory in a big-endian system, the high order byte (most significant byte) is stored at a lower address in memory, while the lower order byte is stored at a higher address. So, the *big end* of the number comes first in memory. This is illustrated below, represented both in hexadecimal and binary.



To better illustrate this, let's see how the original value maps to the individual cells in memory:

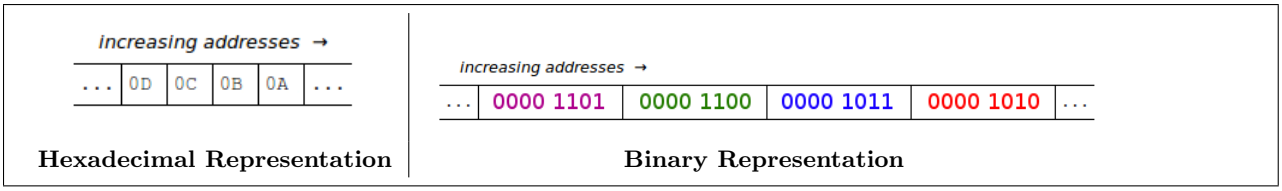


Notice that the most significant byte ($0A$ or, equivalently, 0000 1010) is stored at address a – the lower address – while the least significant byte ($0D$ or, equivalently, 0000 1101) is stored at address $a + 3$: a higher address.

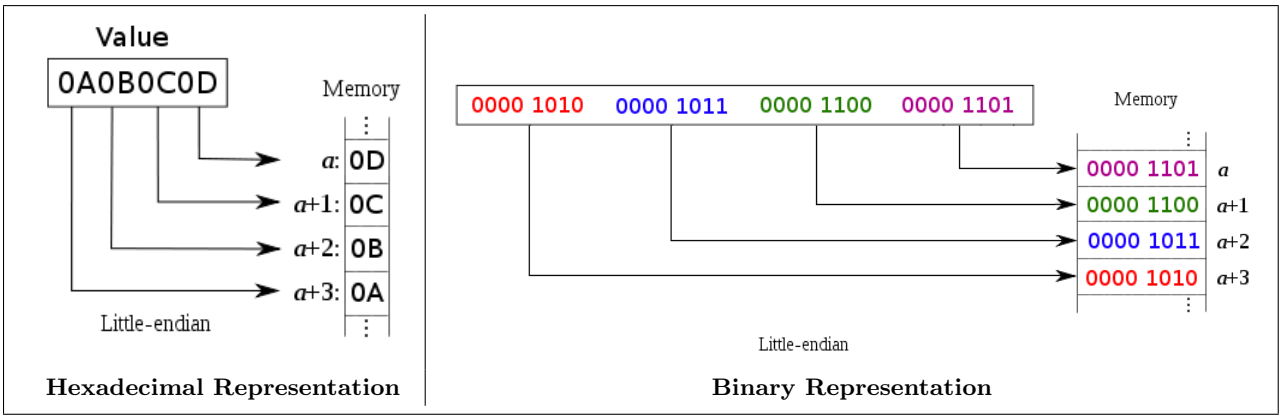
Take a moment to examine the figures above and convince yourself that the big end of the number (the most significant byte) comes first in memory in a big-endian system.

Little-Endian System

When we store the value $0A\ 0B\ 0C\ 0D$ in memory in a little-endian system, the high order byte (most significant byte) is stored at a higher address in memory, while the lower order byte is stored at a lower address. So, the *little end* of the number comes first in memory. This is illustrated below, represented both in hexadecimal and binary.



To better illustrate this, let's see how the original value maps to the individual cells in memory:



Notice that the most significant byte ($0A$ or, equivalently, 0000 1010) is stored at address $a + 3$ – the higher address – while the least significant byte ($0D$ or, equivalently, 0000 1101) is stored at address a: a lower address.

Take a moment to examine the figures above and convince yourself that the little end of the number (the least significant byte) comes first in memory in a little-endian system.