Endianness

There are two kinds of endianness: position endianness and consistency endianness. The first is a concern of hardware designers. The second is a concern for software developers.

Position endianness

Position endianness relates to the ordering of bytes within data busses. When two sockets of the same data bus width, but different endianness, access the same address then the bytes of the bus are seen in reversed order.

When data is transferred from a narrow source data bus to a wider little endian destination data bus then the first narrow word is presented at the destination in the lower addressed byte(s). When data is transferred from a narrow source data bus to a wider big endian destination data bus then the first narrow word is presented at the destination in the higher addressed byte(s).

When data is transferred from a wide little endian source data bus to a narrow destination data bus then lower addressed byte(s) are presented at the destination first. When data is transferred from a wide big endian source data bus to a narrow destination data bus then higher addressed byte(s) are presented at the destination first.

In a system with devices of differing endianness, no configuration of position endianness on hardware interfaces can avoid the need for programmers to consider consistency endianness in their software design.

Consistency endianness

Consistency endianness relates to how software represents data within a byte-invariant memory. Specifically, consistency endianness is the mapping of data bytes to memory byte addresses. Ensuring consistency endianness is a concern of software developers and can not be resolved by hardware design. To understand consistency endianness, it can be helpful to visualize all hardware bus interfaces as being 1 byte wide.

Consider this 8-byte data structure.

```
struct A {
   int x = 0x12345678;
   char y[4] = "mnpq";
}
```

The big and little endian mappings of data to byte addresses are as follows.

	data	
byte address	big endian order	little endian order
0	0x12	0x78
1	0x34	0x56
2	0x56	0x34
3	0x78	0x12
4	m	m
5	n	n
6	p	p
7	q	q

An 8-byte (64-bit) data bus interface of a NoC can transfer the data structure in 1 cycle, but does not know about the ordering of data within a software data structure.