Endianness

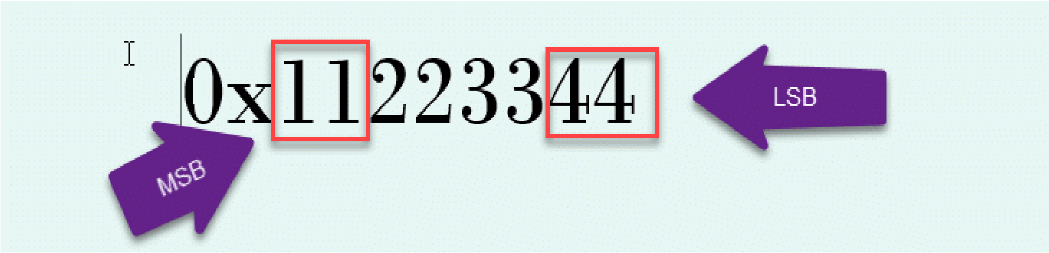
- Endianness refers to the bytes order in which data stored in the memory and also describes the order of byte transmission over a digital link.

- Basically, Endianness comes in two varieties **Little Endian** and **Big Endian** and in which order data will be stored in memory it depends on the endianness.

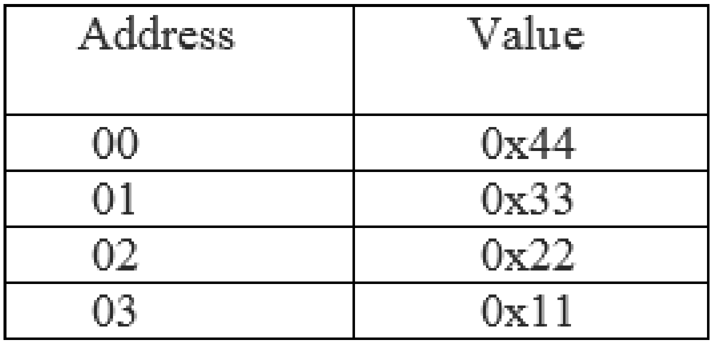
- In Little endian machines, LSB byte store first (means at lower address) or in order words last byte of binary representation of the multibyte datatype is stored first.

- In Big endian machines, MSB byte store first (means at lower address) or in order words first byte of binary representation of the multibyte datatype is stored first.

- Let us assume, we have a 32bit processor and need to store a value in memory. This value is **0x11223344**.

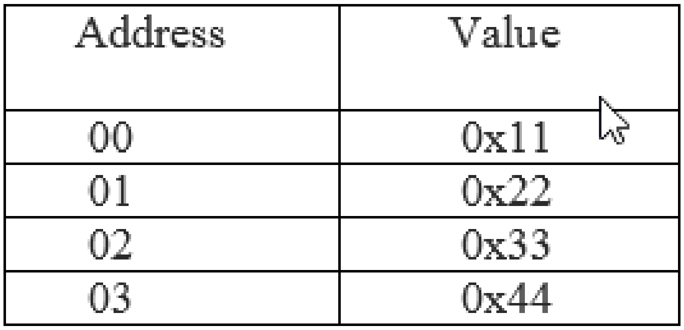


- **Little Endian:**
  - In the Little-Endian machine, LSB byte will store first. So, the LSB Byte will store at the lowest memory address.
  - See the table,



Address	Value
00	0x44
01	0x33
02	0x22
03	0x11

- **Big Endian:**
  - As we know that in Big-Endian MSB Byte will store first. It means the MSB Byte will store at the lowest memory address.
  - See the table,



Address	Value
00	0x11
01	0x22
02	0x33
03	0x44

- **Note:** Some processor has the ability to switch one endianness to other endianness using the software that means it can perform like both big endian or little endian at a time. This processor is known as the **Bi-Endian**. Here are some architecture (ARM version 3 and above, Alpha, SPARC) who provide the switchable endianness (support bi-endianness) feature.

- **C code to verify that system is Little Endian or Big Endian.**

```
#include <stdio.h>

int main()
{
        unsigned int i = 1;

        char *c = (char*) &i;

        if (*c)

                printf ("Little endian");

        else

                printf ("Big endian");

        return 0;

}
```

- ## Which Endianness is better?

  - There is no meaning to say who is the better big endian or little endian, it only arranges the data in a predefined order.

  - **Little Endian:** In the case of little-endian, we can access the first bytes of data with zero offsets because LSB stored at the lower address. Hence in case of the little-endian relation between address and offset is 1:1 and easy to access the data.

  - **Big Endian:** In the case of big-endian MSB byte stored at the lower address, so we can access the MSB byte with zero offsets and check the sign of data without knowing the length of data.

  - **Note:** Endianness is not associated with the register that means register is neither big endian and little endian. Register only stores the data in bit format and its leftmost bit is MSB bit and the rightmost bit is LSB bit.

  - **Conclusion:** We find there is no advantage of using one endianness over the other, both only define the byte sequence order. Today mostly personal computers and desktop come in little-endian architecture. Endian also do not affect the single byte it only affects the multi-byte data because we consider byte is the atomic unit from the storage point of view.