CASPER

CS305: Computer Architecture

Endianness and Alignment

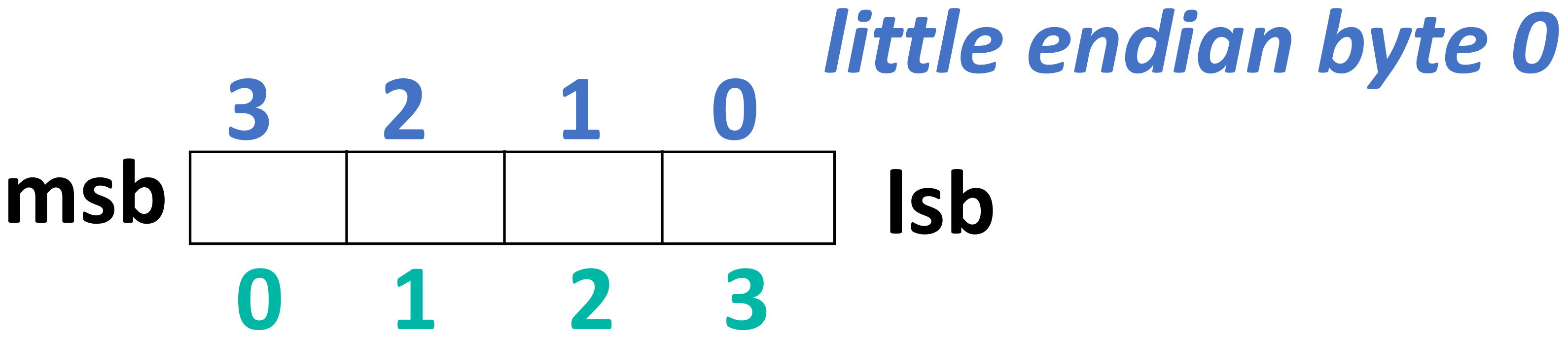
https://www.cse.iitb.ac.in/~biswa/courses/CS305/main.html

https://www.cse.iitb.ac.in/~biswa/

Endianness (Byte ordering within a word)

- Big Endian: address of most significant byte = word address (xx00 = Big end of word), MIPS
- Little Endian: address of least significant byte = word address (xx00 = Little end of word), x86

Think about an egg ☺



little endian byte 0

3 2 1 0

msb lsb

0 1 2 3

big endian byte 0

Example

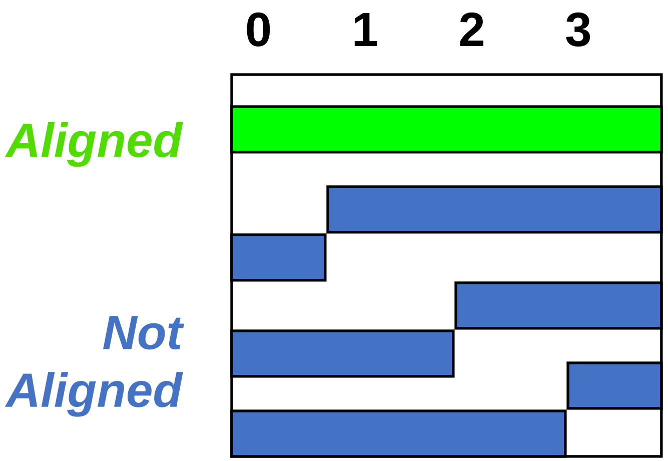
```
unsigned int i = 1;
char *c = (char*)&i;  // reading the LSB
Printf (“%d”, *c);

unsigned int i = 12345678;
char *c = (char*)&i;
Printf (“%d”, *c);
```

```
unsigned int i = 1;
char *c = (char*)&i;  // reading the LSB
Printf (“%d”, *c);
Little endian: 1
Big endian: 0
```

```
unsigned int i = 12345678;
char *c = (char*)&i;
Printf (“%d”, *c);
Little endian: 78
Big endian: 12
```

Instruction Alignment: Why we need it?



Aligned:

x-byte access starting from an address y: y % x must be zero.

MIPS vs X86

MIPS does not allow unaligned accesses

x86 does not enforce alignment ☺

Whose job is to generate aligned/unaligned accesses?

MIPS vs X86

MIPS does not allow unaligned accesses

x86 does not enforce alignment ☺

Whose job is to generate aligned/unaligned accesses?

Compiler

Let’s go a bit deeper

Object of size s bytes at byte add. A is aligned if A mod s = 0

Alignment for faster transfer of data ?

Why fast ??

Think about memory (caches if you know).

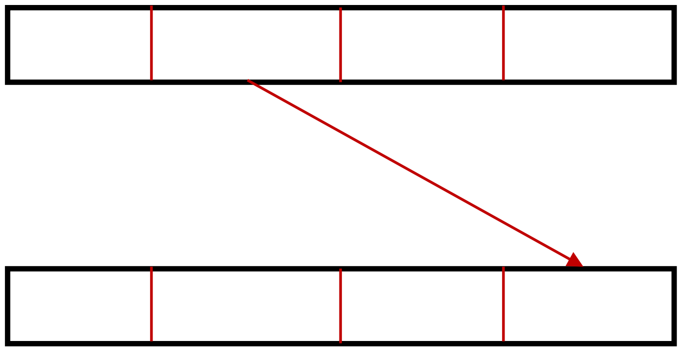
Memory operations and alignment network

LOADs and STOREs need an alignment network that makes sure data loaded/written are aligned.

lb R1, 1($s3)

4-byte chunk

Register R1



For the Curious ones

https://lemire.me/blog/2012/05/31/data-alignment-for-speed-myth-or-reality/

Dhanyavaadaalu