Reminder

You must be familiar with the following formulas:

$$x^a = \prod_{i=1}^{a} x \qquad x \times \ldots \times x,\ a \text{ times} \tag{1}$$

$$x^{-a} = \frac{1}{x^a} \tag{2}$$

$$\sqrt[a]{x} = x^{\frac{1}{a}} \tag{3}$$

$$\log(x^a) = a\log(x) \tag{4}$$

$$x^a x^b = x^{a+b} \tag{5}$$

$$\frac{x^a}{x^b} = x^a x^{-b} \tag{6}$$

$$= x^{a-b} \tag{7}$$

$$(x^a)^b = x^{ab} \tag{8}$$

$$\tag{9}$$

especially when $x = 2$ in computer science

Number Representations: Bases

Base or radix: Number of unique digits (counting 0) used to represent numbers in a positional numeral system.

- Binary: Base 2, i.e. 2 digits (0 and 1)
- Decimal: Base 10
- Hexadecimal: Base 16, 0..f Convenient representation of 4 bits values: 0000 = 0 to 1111 = f (or F)
- Other bases do/did exist:
  - Octal: Base 8 (Digits: 0..7)
  - Duodecimal: Base 12 (Dozens, Grosses...)
  - Vigesimal: Base 20 (Etruscan, Mayan... East-Asian)
  - Sexagesimal: Base 60
  - Base64: Base 64 (a-zA-Z0-9 and two other digits usually '+' and '/')

Number Decomposition on a Base

Let r be an integer greater than 1.
Let A be an positive integer.
There exists a unique $\{b_0, b_1, b_2 ... b_{n-1}\} \in \{0..(r-1)\}^n$ so that

$$A = \sum_{i=0}^{n-1} b_i r^i$$

and A will be represented by $b_0 b_1 b_2 ... b_{n-1}$ in base r which will be denoted by $(b_0 b_1 b_2 ... b_{n-1})_r$.
If there is no ambiguity, the subscript is omitted.
The usual number representation (0, 1...) is purely conventional (some could use א, ב, ג ...).

Number Representations: An example

So...

- '145_{10}' or `145d` or `d145` in base 10;
- '10010001_2' or `b10010001` or `b1001 0001` in binary;
- '221_8' or `0221` in octal;
- '91_{16}' or `'0x91'` in hexadecimal;
- 'cq_{64}' in Base64 (assuming a=0, b=1, ..., A=26,...,0=52,...)

Number Representations

The Number Representation does not change the Number (Value).

$$145_{10} = 1 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

$$\begin{aligned} 10010001_2 &= (1 \times 2^7 + 1 \times 2^4 + 1 \times 2^0)_{10} \\ &= 127_{10} + 16_{10} + 1 \\ &= 145_{10} \end{aligned}$$

... and so on ...

Number Representations in Computer Languages

How to represent the numbers in the various "common" computer-oriented bases is implemented in most languages:

C	`printf("%04x", value)`
C++	`std::cout << std::hex << value`
Python	`"{0:b}".format(value)`
Java	`String.format()`
...	...

According to the context it is more convenient to use one base (and therefore a representation) instead of an other...

Units in Computer Science

- The smallest unit of information is the bit
  - Stands for Binary digIT (and is conveniently meaningful in English!)
  - 0 or 1
- Units of Storage
  - 1 Byte = 8 bits. "0000 0000" to "1111 1111"
- 1 Byte = 8 bits. "0000 0000" to "1111 1111"
- 1 Kilobyte $1024 = 2^{10}$ bytes; Should be denoted by 1 KiB (but sometimes 1 Kb, or 1 kB, or 1kb...)
- 1 Kilobyte is 1000 bytes (should be denoted by kB)
- 1 Megabyte $1024 = 2^{10}$ Kilobytes, that is, $2^{10} \times 2^{10} = 2^{20}$. Should be denoted by 1 MiB
- 1 Megabyte is exactly 1,048,576 bytes (more than 1,000,000 bytes which is 1 megabyte)
- 1 Gigabyte is 1024 Megabytes, i.e. 2^{30} bytes; Denoted by GiB;
- 1 Terabyte is 1024 Gigabytes, i.e. 2^{40} bytes; Denoted by TiB;
- 1 Petabyte is 1024 Terabytes, i.e. 2^{50} bytes; Denoted by PiB;
- 1 Exabyte is 1024 Petabytes, i.e. 2^{60} bytes; Denoted by EiB;

Units in Computer Science: Comments

- Most of the time the 'i' is missing
- Watch out! Commercial/Mundane units are sometimes not the Technical ones: A 800 GB ard drive is only 762 GiB...
- You have to be absolutely familiar with the above (Petabytes systems are "common" now);
- Only powers of 2 and multiple of $2^{10} = 1024$ to go from one multiplicator to another
- In 1980, 1 GiB costed \$1M; Today, less than \$0.10

Counting bytes

- When studying operating systems, we often need to count "chunks" of bytes in some space (memory, file system...)
- Example #1: how many 1 MiB chunks are there in a 8MiB file?
- Example #2: How many 4KiB chunks are there in a 8GiB file?
- Best way to do this: use powers of 2!

Counting bytes: Examples

- 4KiB $= 4 \times 2^{10}$ bytes $= 2^2 \times 2^{10}$ bytes $= 2^{12}$ bytes
- 8GiB $= 8 \times 2^{30}$ bytes $= 2^3 \times 2^{30}$ bytes $= 2^{33}$ bytes
- So there are $2^{33}/2^{12} = 2^{33-12} = 2^{21}$ chunks of 4KiB in a 8GiB file (that is a bit more than 4 millions and not exactly 4 millions).

In-class Exercises

- How many 8KiB chunks in a 2MiB file?
- How many 32-byte elements in a 128KiB array?
- How many 4MiB images in a 256GiB digital library?
- How many 1GiB memory zones in a 16EiB memory?
- How many 4KiB pages in a 2GiB virtual address space?

Note: You don't need to know what "images", "digital library", "memory zones", "pages", "virtual address space"... are to answer!
Solution: 2^8, 2^{12}, 2^{16}, 2^{34}, 2^{19} (please verify them)

Partitioning

Partitioning a set is dividing into **disjoint subsets**:

- Partition a pie into slices;
- Partition a supermarket in aisles;
- Partition a federal republic into states;
- Partition the set of integers between even and odd ones;
- Partition a file into blocks;
- Partition a disk into sectors;
- Partition an address space into pages
- ...

The set of all the subsets is called a **partition** of the initial set.

Partition: Properties

Note: There is a mathematical definition of a partition

- We want the union of all parts to be the whole set (nothing is left behind);
- We want any element to belong to exactly one part;

Note however that a set can be partitioned differently and that the partitions are not necessarily compatible (the USA can be split into states but also into timezones for instance there are parts of the same state that are on different timezones)
Why partitioning?
Whatever is partitioned should be usually easier (that is faster) to "manage" (politically, economically, logically...)
(But the "right" partitioning schema has to be chosen)

Partition: Addressing

One a partition is created, we often want to refer to the subset an item belongs to or to restrict an operation to a subset.
We therefore need to label the subset. The way the address is created is the choice of the partitioner (Florida, "Aisle 4"...).
For computers, subsets/chunks of elements are just identified by numbers: 0, 1, 2... stored in binary formats: their `address`.

Addressing

- Key question: What is the range of addresses that we need to address all chunks (uniquely)?
- We also want the smallest range not to waste address bits by having large addresses that are not used (There are two houses on that street: one is at 123456788, the other at 12346789. It could be simplified: one is at 1, the other at 2)
- E.g.:
  - I have 7 mansions in Honolulu
  - I want to "address" them: 0, 1, 2, 3, 4, 5, 6
  - 1 bit and 2 bits are not enough to address them all
  - By binary addresses: 000, 001, 010, 011, 100, 101, 110
  - 3 bits are necessary (but one slot is wasted: 111)
  - If 4 bits were used, 1 bit would always be set to 0 (and 9 slots would never been used)

Addressing

- To address 2 items: 1 bit is necessary; To address 4 items: 2 bits are necessary; To address 8 items: 3 bits... To address 2^p items: p bits are necessary
- To address n items, $\lceil \log_2(n) \rceil$ where $\log_2(x) = \frac{\log(x)}{\log(2)}$ and $\lceil . \rceil$ is the function that rounds a value up to the nearest integer (e.g. $\lceil 1.5 \rceil = 2$, $\lceil \pi \rceil = 3$, $\lceil 3 \rceil = 3$
- Most of the time n will always be a power of 2 (and you can check that if $n = 2^p$, $\log_2(n) = p$

Addressing: In-class exercises

- How many address bits do you need to address each...
  - byte in a 2MiB memory?
  - 4-byte word in a 1MiB memory?
  - 4KiB page in a 16MiB address space?
  - 1MiB file in a 4GiB file system?
- Count how many items you have, take its $\lceil \log_2 \rceil$
- Solutions: 21,18, 12, 12

Conclusion

- You must be absolutely comfortable with all this since we'll soon be doing counting/addressing all the time
- Besides, counting and addressing are what computer science is about :)
- Simple quiz about this (You won't need a calculator)