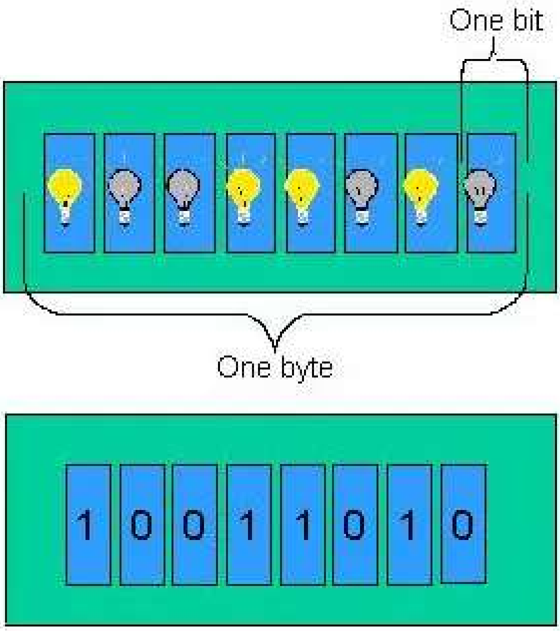
# Concept of Memory and Binary

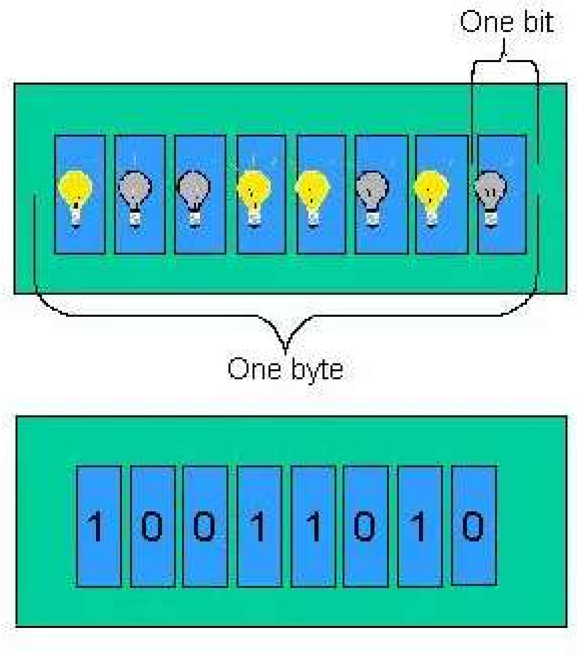
Imagine yourself in a dark room having only one light bulb off. You will just have two options then, either to remain in the dark or turn on the lights. These “on” and “off” conditions can be treated as a signal for a person outside the window of the room. Now if you have two lights in the room, four such signals can be made. “Off and off”, “off and on”, “on and off”, “on and on”.Imagine yourself in a dark room having only one light bulb off. You will just have two options then, either to remain in the dark or turn on the lights. These “on” and “off” conditions can be treated as a signal for a person outside the window of the room. Now if you have two lights in the room, four such signals can be made. “Off and off”, “off and on”, “on and off”, “on and on”.

Computers use the same concept to use store data. The smallest unit of its memory is bit which can be related with the single light bulb in the room. We store “1” in it relating to the “on” condition of the light and “0” for ”off” condition. As explained earlier, four signals can be made then with two bits of memory.

With one light bulb, we can store 1 bit of information. With 8 light bulbs, we could do 8 bits. 8 bits means we could have $2^8$ possible options (Simple P & C right?)

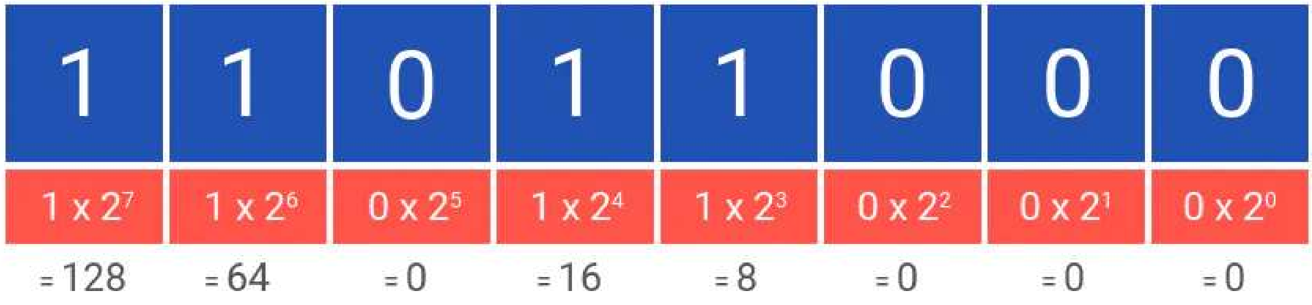


So, if we had enough light bulbs, we could store any amount of data we wanted in a digital form.



Here, one cell represents one bit.
Now, not many light bulbs would fit inside our computer! So computers store bits of data by holding electrons in capacitors, for example. This technology is used in the RAM memory. However, we shall continue using this metaphor of light bulbs as it is simpler to visualize.
So, how many such light bulbs fit into your RAM exactly?
Let's assume your computer has 4GB RAM, for example. If you convert 4 GB according to the above chart, it will be approximately equal to 32000000000 bits. That means your 4GB of RAM holds 32 billion bits. That's 32 billion light bulbs!

*But, what does it actually mean to have 32 billion bits to store and represent information?*
**To understand what these bits can do for us, let's take a closer look at the binary number system. It only uses 0's and 1's to represent any further numeral values – and other types of data, too.**
Have a look at how the number 216 is stored in the memory.



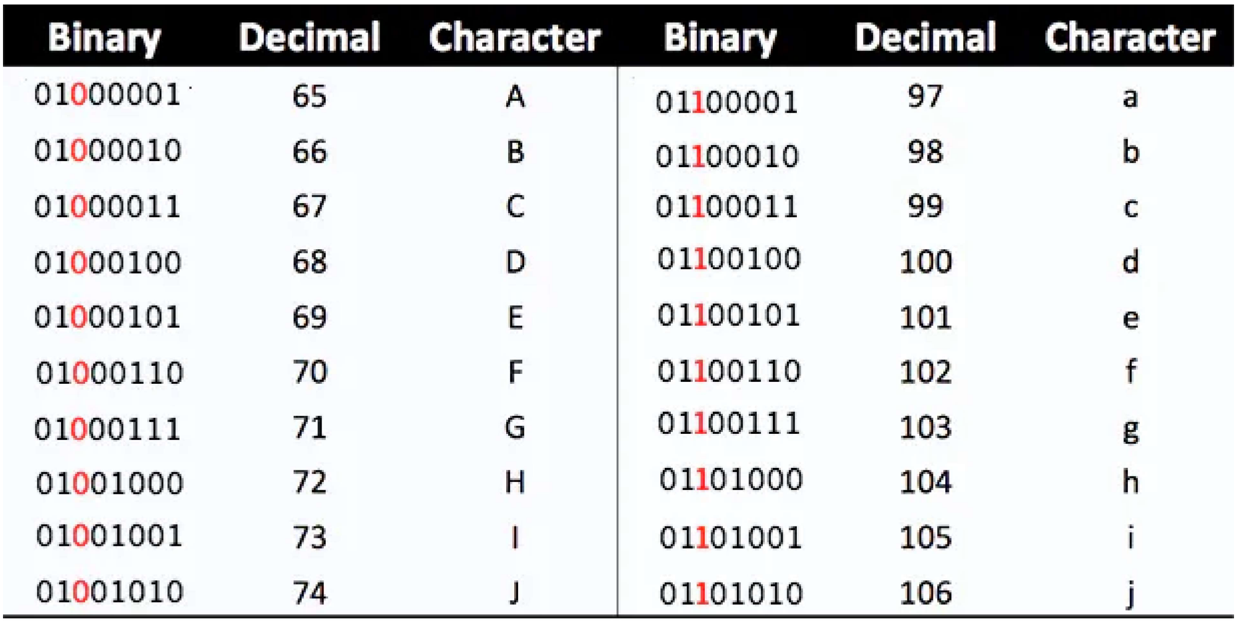
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| $1 \times 2^7$ | $1 \times 2^6$ | $0 \times 2^5$ | $1 \times 2^4$ | $1 \times 2^3$ | $0 \times 2^2$ | $0 \times 2^1$ | $0 \times 2^0$ |
| = 128 | = 64 | = 0 | = 16 | = 8 | = 0 | = 0 | = 0 |

**128 + 64 + 16 + 8 = 216**

The blue boxes represent a single bit of memory in the RAM. Computers store 216 input by the user in its memory in the following way. This decoded form is obtained by a decimal to binary conversion. To learn about more conversions click here. (complete the link)

We have learned about how numbers are stored in memory, but what about characters?

At this point of time,we are capable of representing any number in binary form. We will use this to store alphabets. We will assign every character with a number. Let us understand this by looking at how "A" is stored in memory.

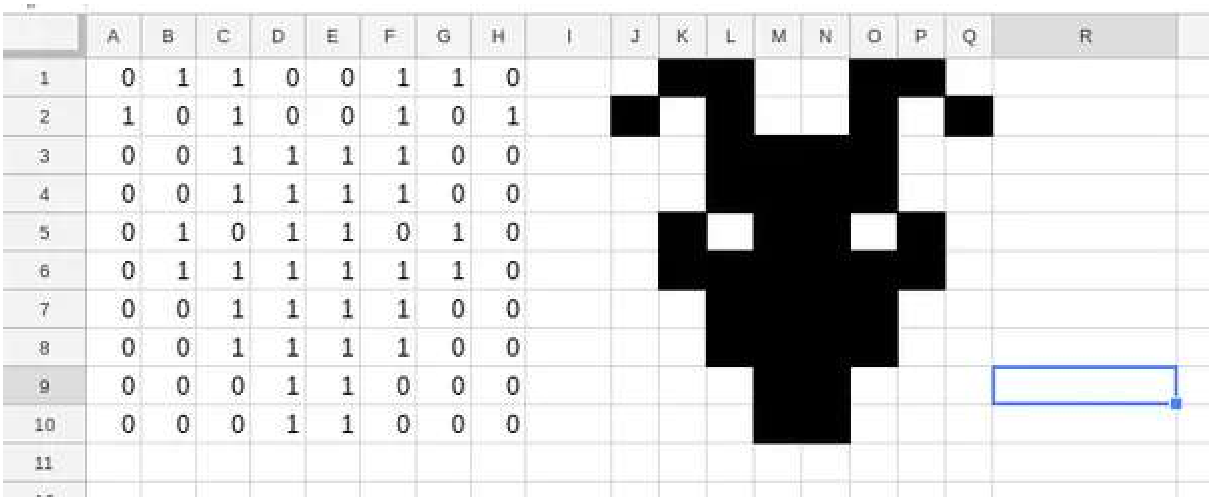


| Binary | Decimal | Character | Binary | Decimal | Character |
|---|---|---|---|---|---|
| 01000001 | 65 | A | 01100001 | 97 | a |
| 01000010 | 66 | B | 01100010 | 98 | b |
| 01000011 | 67 | C | 01100011 | 99 | c |
| 01000100 | 68 | D | 01100100 | 100 | d |
| 01000101 | 69 | E | 01100101 | 101 | e |
| 01000110 | 70 | F | 01100110 | 102 | f |
| 01000111 | 71 | G | 01100111 | 103 | g |
| 01001000 | 72 | H | 01101000 | 104 | h |
| 01001001 | 73 | I | 01101001 | 105 | i |
| 01001010 | 74 | J | 01101010 | 106 | j |

This image gives a clear view of how different alphabets are stored in the binary form.

Take a look at “A”. Its ASCII code is 65 and when we convert 65 to binary form, we get “01000001”. So, the computer stores “A” as “01000001” in its memory. But how does it differentiate between integer 65 and “A” then? This is done with the int and char data types. We will discuss them in-depth later. Click on the link to view the ASCII codes of other characters.

Now, let us look at how computers store images in memory. We will understand this with an example.



|  | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 7 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 8 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 9 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

Assume the rows and columns of this excel sheet to be the pixels of the image. Now the computer will give every pixel a special bit in its ram and then specify their value as 0 or 1 according to the black or white color we want in our image. 0 for white and 1 for black. This is how simple it is to store a black and white image in memory.

Now,

Don't worry, it isn't your typical boring one!

You want to save the below image on your computer. Suggest a suitable way for the computer to store such an image in its memory.

Hint: Try applying the concepts of storing a black and white image discussed above.