Mr. Bellavia

# LESSON 01 - YOUR COMPUTER, A BINARY MACHINE

In this lesson we will discuss a computer's primary purpose of input, processing, and output. We will also look at how information is represented on a computer by use of binary code. It is important to note that a computer is not limited to just a box on a desk. Currently, computers come in many forms: desktops, laptops, phones, tablets, etc. Computers are even built into much of the technology we use today: cars, appliances, watches, etc.

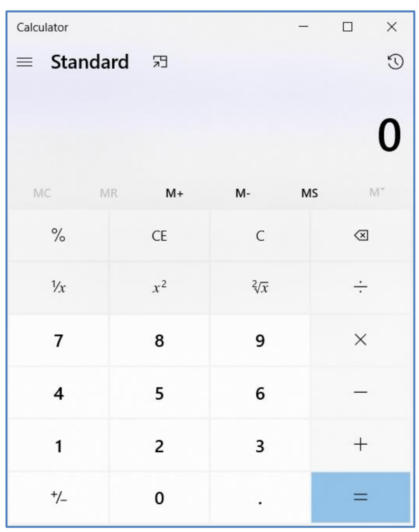
**Sections:**

I. HOW DOES A COMPUTER WORK? ........................................................................ PAGE 1
II. HOW A COMPUTER INTERPRETS & STORES INFORMATION (BINARY CODE) ............... PAGE 4

## I. HOW DOES A COMPUTER WORK?

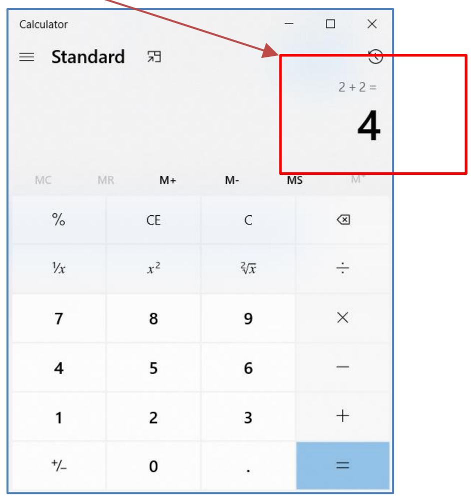
To understand how computers/phones/tablets/etc., work is a complex matter! A simple way to view a computer is as an **input/output machine.** First you **input** something into a computer using an input device, for example, with a keyboard, microphone, mouse, etc. Second, your computer **processes** that information you just inputted. Third, your computer will give you feedback by **outputting** that processed information to an output device, for example to a screen, or through speakers, or to a printer, etc.

Think about a simple calculator program:



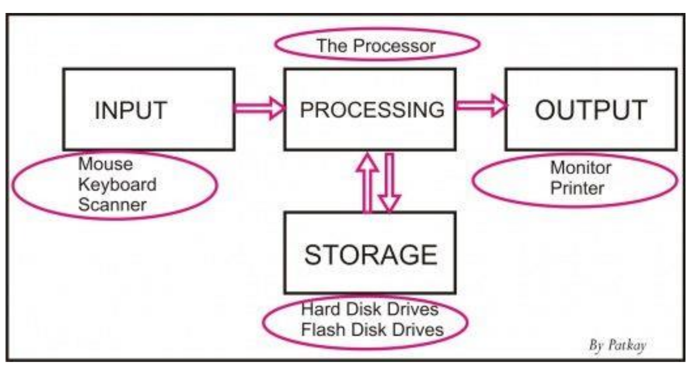
Mr. Bellavia

If we **input** *'2 + 2'* followed by the *'Enter'* key, the computer will **process** an answer of *'4'* then **output** that answer to the screen:




This is a very abstract representation of how the above calculator program works. **Abstraction** means hiding the details that do not matter, and in this case, we do not need to know how the computer calculated the answer we just need to know how to use the calculator program. In fact, you face abstraction everyday. For example, if you drive a car, you do not need to know how the engine works along side the transmission, you just need to know how to operate the vehicle without having to worry about the mechanical make up of the car. Hence, you have an **abstract** understanding of how a car works. The same goes for a computer that runs computer programs (like the calculator program above). We do not need to understand how the electric circuits inside the computer processes information when we use a computer program, we just need to know how to use a computer program so that we can perform a specific task (in this case, using the calculator program to calculate *'2 + 2'*).

The calculator program above demonstrates the basic computer principles mentioned earlier: **input, processing,** and **output.** Consider the following diagram:




This is a very abstract view of how a computer works. We are not going to get into all the details of each component of a computer. That is not the point of this course. Just know that when we use a computer we **input** information, the information is **processed**, and feedback/**output** is given back to us (we'll jump into more of these diagram details in the upcoming lessons). Here is a breakdown of this process:

**INPUT:** Any information that is given to a computer is considered input. This is usually done with an input device like a keyboard, mouse, scanner, etc. For example, the calculator program above was given the input: *'2 + 2'*.

**PROCESSING & STORAGE:** Information that is inputted into a computer is then processed by the **Central Processing Unit (CPU)** and stored into memory. For example, the calculator program above processed *'2 + 2'* resulting with the answer *'4'* (more on this step in the next lesson).

**OUTPUT:** Once information is processed it is then outputted to the user in a meaningful way. This is usually done with an output device like a monitor, printer, speakers, etc. For example, the calculator program above outputted the answer of *'4'* to the screen.

## II. HOW A COMPUTER INTERPRETS & STORES INFORMATION (BINARY CODE):

Think about all the information/media you view on your computer/phone/tablet/etc. Whether it be simple text, an image, a sound clip, the information/programs on a computer are represented the same. How? Using **binary code.** Binary code consists of patterns of zeros **'0'** and ones **'1'** which represent information.

Let us demonstrate with an example. Consider when we type text into a computer with a keyboard. This text can be represented as binary code. Way back when computers were first being developed a standard was created to represent all characters on your keyboard using binary patterns. This standard is called **ASCII** *(American Standard Code for Information Interchange)* and is still use today. Consider the following ASCII table:

**ASCII Code: Character to Binary**

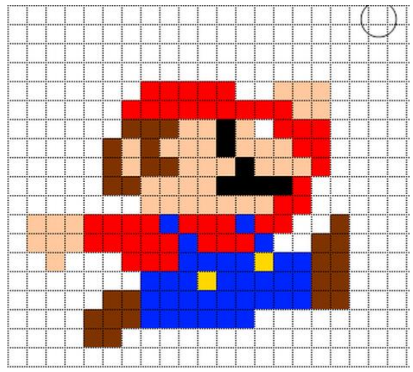
| | | | | | |
|---|---|---|---|---|---|
| 0 | 0011 0000 | O | 0100 1111 | m | 0110 1101 |
| 1 | 0011 0001 | P | 0101 0000 | n | 0110 1110 |
| 2 | 0011 0010 | Q | 0101 0001 | o | 0110 1111 |
| 3 | 0011 0011 | R | 0101 0010 | p | 0111 0000 |
| 4 | 0011 0100 | S | 0101 0011 | q | 0111 0001 |
| 5 | 0011 0101 | T | 0101 0100 | r | 0111 0010 |
| 6 | 0011 0110 | U | 0101 0101 | s | 0111 0011 |
| 7 | 0011 0111 | V | 0101 0110 | t | 0111 0100 |
| 8 | 0011 1000 | W | 0101 0111 | u | 0111 0101 |
| 9 | 0011 1001 | X | 0101 1000 | v | 0111 0110 |
| A | 0100 0001 | Y | 0101 1001 | w | 0111 0111 |
| B | 0100 0010 | Z | 0101 1010 | x | 0111 1000 |
| C | 0100 0011 | a | 0110 0001 | y | 0111 1001 |
| D | 0100 0100 | b | 0110 0010 | z | 0111 1010 |
| E | 0100 0101 | c | 0110 0011 | . | 0010 1110 |
| F | 0100 0110 | d | 0110 0100 | , | 0010 0111 |
| G | 0100 0111 | e | 0110 0101 | : | 0011 1010 |
| H | 0100 1000 | f | 0110 0110 | ; | 0011 1011 |
| I | 0100 1001 | g | 0110 0111 | ? | 0011 1111 |
| J | 0100 1010 | h | 0110 1000 | ! | 0010 0001 |
| K | 0100 1011 | I | 0110 1001 | ' | 0010 1100 |
| L | 0100 1100 | j | 0110 1010 | " | 0010 0010 |
| M | 0100 1101 | k | 0110 1011 | ( | 0010 1000 |
| N | 0100 1110 | l | 0110 1100 | ) | 0010 1001 |
| | | | | space | 0010 0000 |

As you can see, all numbers, lowercase characters, uppercase characters and symbols can be represented by a unique pattern of zeros '0' and ones '1'. **Note:** The above chart has been expanded over the years to contain many more characters.

All computers in the world use ASCII or they would not be able to communicate with each other. Based on the above ASCII chart we can form English words. For example, the word *'Computer'* can be represented in binary as:

01000011 01101111 01101101 01110000 01110101 01110100 01100101 01110010 00001101 00001010

Even images and sound clips can be stored as binary. Images, for example, consist of small dots (pixels) on a screen where each dot has a colour:



For example, we can say that the pixel that holds Mario's thumb above is at (x, y) coordinate equal to **(3, 6)** and that the colour has a hex code of **#ffc79e** (hex codes are common for representing colours on the web). In theory we could transform all this information to binary:

**3** = 00110011

**6** = 00110110

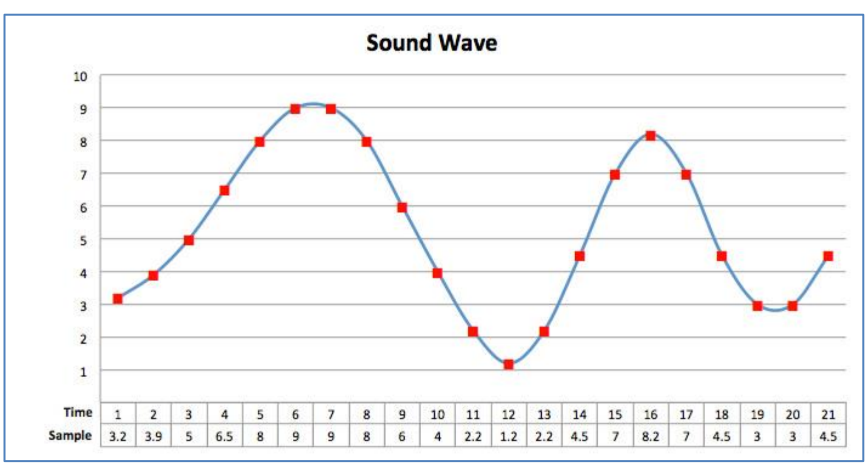
**ffc79e** = 01100110 01100110 01100011 00110111 00111001 01100101

We can do this for every pixel on the screen and we will end up with a large sequence of zeros and ones:

00110000 00110000 01100110 01100110 01100110 01100110 01100110 01100110 00110001 00110001 01100110 01100110 01100110 01100110 01100110 01100110 …

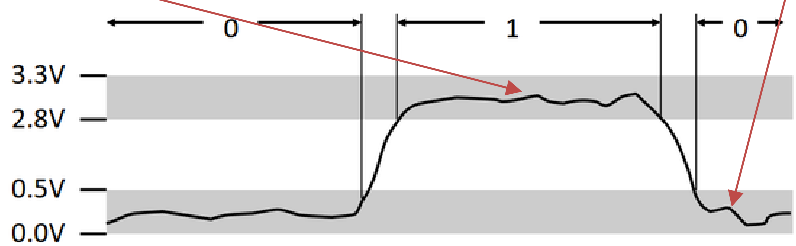
**Note:** This is not exactly how images are converted to binary, but the example above is more for demonstration purposes.

Let's look at another example. Sound clips are points on a sound wave:



Just like with the image of Mario, we can find an appropriate binary representation for every point on the sound wave above. The first point on the sound wave has an (x, y) coordinate of **(1, 3.2).** We can then construct a binary representation of these numbers for this and every other point on the sound wave.

Why was binary code chosen to represent information? Because it is simple! A binary digit can be only one of two things: zero **'0'** or one **'1'**. At the computer hardware/circuitry level, it is easy to represent '0' and '1'. This is because electric currents can be somewhat controlled. For example, we can turn circuits **'on'** or **'off'** where 'on' can represent a **'1'** and 'off' can represent a **'0'**. Modern computers have a constant flow of electricity, so '0' and '1' can be represented by **low voltages** and **high voltages** respectively:




Along with electrical current, there are many ways to represent/store 0s and 1s. For example, the billions of microscopic on/off transistors in a USB key or the billions/trillions of points on a magnetic surface (i.e., hard drives, CDs, DVDs, etc.) (more on this in the next lesson). You could even write all the 0s and 1s on a piece of paper but that would require lots of paper!

Also consider how information is transferred across the internet using cables or wireless radio waves. Cables can use the concept of low/high voltages to represent the 0s and 1s, whereas wireless radio waves can use low/high radio frequencies to represent 0s and 1s. The possibilities to represent 0s and 1s are virtually endless!