Riding the digital bandwagon

by Utpal Phukan - Thursday, January 16, 2014

http://gonitsora.com/riding-digital-bandwagon/

This series of articles are primarily intended at students and laymen who are from time to time harassed by all the digital acronyms floating around and baffled by tricks of the dotcom kids. The author will try to be as simple as possible while explaining things, or more precisely try to explain things in laymen's terms. In this series we will traverse from computers to network of computers, internet to Google, the technologies powering websites like Facebook, mathematics behind some of these technologies etc. As the series will progress we will try to be more technical in terms of contents and explanation.

Let's break the ice by starting with the most illustrious of them all, "the computer". There couldn't be a more precise name for a computer than a "computer". What can a computer do? Well, as the name suggest what it all can do is to compute (!). What does it compute? I can see your eyebrows raising when I tell you that all a computer can do is "addition", "subtraction" and some logical comparisons like less than, greater than stuff; in simple terms a computer is no better than a kindergarten kid in terms of mathematical abilities. Yeah, I can see your disbelief. When we write a document in a Word processor like MS Word or take a print outwhere on earth is the "addition", "subtraction" stuff? OK, we will discuss all these. But before that let's learn one important term called "abstraction". In the world of computers abstraction means to hide lower level details. But the term "lower level" has quite different meaning than its literal one. Computers can do its job with the help of two parts- one is hardware (i.e. the machine, the visible part) and the other is software (the invisible part). Let's take a simile a music player. The music player is the hardware but without a record (of course with songs in it) it is essentially useless. The songs engraved in the records can be thought of as the software of the music player. (The term lower level cited above means nearer to the hardware and higher level means nearer to we humans). In case of computers, software is actually set of instructions, i.e. *what-we-want-the-computer-to-do* stuff packaged in computer's language. Now, what is this term computer language? In simple terms computer languages are languages which can be understood by it. Let's be more specific. A computer is basically an electronic (digital) device. Imagine an electric switch used to light bulbs in our home. An electric switch has only two states, i.e. ON and OFF. When it is ON the electricity flows through the wires to the bulb and hopefully it glows (if not fused). In terms of mathematics this state of ON can be defined as 1 and OFF can be defined as 0. The computer consists of large numbers of such electronic switches which are capable of switching states (i.e. ON/OFF) very fast. The speed of a computer is roughly the speed (of flipping state) of these switches. A computer can only understand 1 and 0. Thus comes the term BINARY (means two). You remember in your kindergarten days you learned a number system called DECIMAL (deci means 10). The decimal number system consists of 10 numbers (from 0 to9) and 10 is called the base number of the system. All the other numbers can be shown as the power of 10. One example:

$$123=1times 10^2+2times 10^1+3times 10^0$$

Just like this the BINARY number system consists of only two numbers 1 and 0. One example:

$$101=1times 2^2+0times 2^1+1times 2^0=4+0+1=5$$

5 is the decimal equivalent of the binary number 101 (How do we convert a decimal number to binary. Very simple- just divide the decimal number by 2(i.e. the base number of binary) repeatedly until the remainder becomes less than 2. Then write down all the remainder from bottom to top. One example:

You might also remember that Multiplication is a shortcut method of doing Addition and Division is a shortcut for Subtraction. Every letter, characters seen in the keyboard has equivalent binary representation inside the computer. Every high level instruction that we give to the computer is actually an *abstraction* of some *addition/subtraction/logical comparison* of these binary numbers. (Next time when you click on the mouse just think about the number of calculations the computer performs). What we see in the monitor is the human understood view, but behind all these abstractions it is still binary for the computer (there are no 0 and 1 in the computer either- a computer processing something means actually flipping of its electronic switches and flow of electrons- period). We humans don't understand binary- so we introduced an *abstraction* called programming language. A programming language is a way of giving instruction to the computer with natural language like keywords rather than taking the right rope walk of giving them in binary (!)- The beauty of human mind. Just like natural language, a programming language also has its vocabulary and grammar. How will the computer understand these programming languages whereas we know that all it can understand is binary? Thus enters another digital term called *compiler*. A compiler is like a language interpreter- it converts the high level programming language instructions to machine language. Ultimately these machine language instructions are converted to binary which the computer obeys.

A computer is as good as the programmer. It is by no means smarter than humans. The only advantage of it is speed and reliability of calculations. In fact human mind is the smartest computer on earth. But we are lazy and that's why we invented computers. Invention is a doubtful term for me. Do we actually invent things? Can we create something out of the blue? The universe has everything in it. All we do is abstraction of solving a particular problem. We are like children trying to make something with a pile of Lego. What is mathematics after all? Isn't it a way of representing logical thoughts in numerical/symbolic terms? Let's take an example. Humans have always been fascinated by the birds and just like birds we also wanted to fly. So we *'invented'* airplane. This is our costly way of solving the *flying-like-a-bird* problem. We scoff at the mythological story of Gods flying without using any devices. How can we be so sure that there can't be a better way of flying? Remember the famous three famous laws of Newton? Newton discovered some patterns in the behavior of things which he summed up as the three laws. Someday you might discover some other patterns and can come up with a better abstraction. Abstraction gives comfort. We are lazy and so we love it. Keep your thoughts open.

[ad#ad-2]
