Secrets of The Can-Do Machine

Paul S. Wang, Sofpower.com

December 19, 2020

At the center of the information revolution is the digital computer—a can-do machine like no other. At once it can be a TV, a phone, a thermometer, a compass, a music box, a language interpreter, and much more. In fact we know it can perform any task following a well-designed program—an "app" (application) as we call it these days.

It is exactly such flexibility that makes the modern computer a very different creature, namely a *general-purpose* or *universal machine*. But what features or characteristics make this possible? Are there things the modern computer can't do? What tasks are easy or hard for the computer?

In this computational thinking (CT) article, we will take a closer look and try to provide some answers. By stripping away the complications, we arrive at the nature and essence of the digital computer. This way, we can reveal the reasons why it is a universal machine. We can also acquire CT principles that are useful everyday.

You will find the reading both interesting and eye opening.

A Universal Machine

The unique revolutionary power of the digital computer comes from its *universality*. What makes it a universal machine vs. other specific-use machines is the fact that it is a *stored-program* machine. By storing, calling and executing any given program on the fly, a computer instantly becomes a different machine without rewiring or other physical modifications.

The modern computer, with its hardware, operating system and apps can be very complicated and hard to understand. Yet, at its core, it's quite simple—the CPU executes a sequence of *instructions*, one after another. A computer program, however complicated or powerful, is just a sequence of instructions and data. Each instruction has an *opcode*, representing the operation or action to be performed, and operands, values supplied for the operation. An operation may store a result in memory, perform I/O, designate which instruction to execute next, or otherwise change the *internal state* of the computer. The key is all the instructions, data, opcode, operand values are all represented by 1's and 0's in memory with the famous *binary code*.

These characteristics of the digital computer are derived directly from the nature of its hardware.

Digital Hardware

1's and 0's are just a way to describe two electrical states—the presence or absence of an electric current or voltage. Such a signal is the smallest unit of data inside a computer and is known as a *bit*. A bit can represent one of two opposing states: on/off, yes/no, yin/yang, up/down, left/right, true/false, and, of course, 1/0. There are many other possibilities but we conventionally refer to these two states as 1 and 0, the two binary digits. A group of 8 bits is called a *byte* and a group of bytes, usually 4 or 8 bytes depending on the computer, is called a *word*.

Be sure to disassociate 0 and 1 as bit values from their day-to-day meaning as numbers. Why not think of 1 as "presence" and 0 as "absence" if you wish.

The *central processing unit* (CPU) is the brain of a computer where information gets processed with electronic logic circuits. A modern CPU, on a fingernail sized silicon chip, may contain billions of transistors—fast, tiny (about 70 silicon atoms, certainly invisible to the naked eye), and cheap devices to store and process electronic signals.

Typically, the CPU performs operations by retrieving and storing data in the main memory, a place to readily access information to be processed. Such is the nature of digital computer hardware and it dictates that, inside the computer, information be represented and processed exclusively in the form of 1's and 0's.

Digital Data

All data inside the computer are digital, represented with 1's and 0's. It is not hard to understand if we first look at our own world. In English, we represent information using words and numbers composed of a set of alphabets (upper and lower case A-Z) and digits (0 through 9). Other languages may use different alphabets.

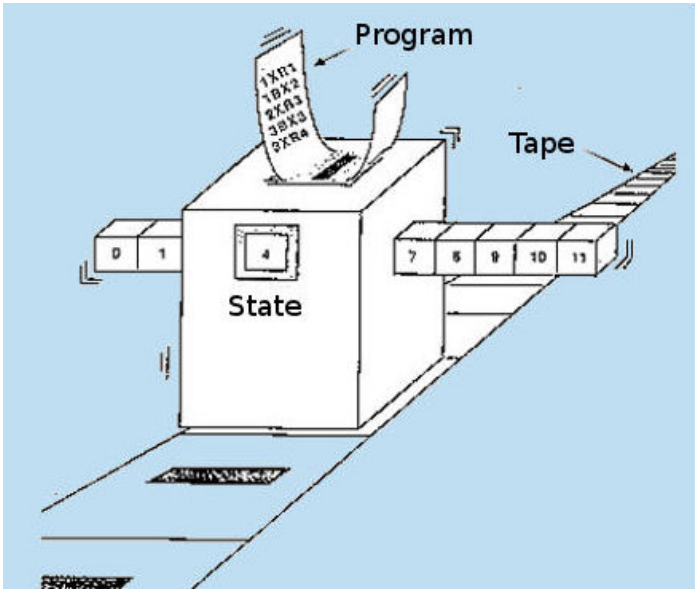
Inside the computer the alphabet has just two symbols, namely 1 and 0. In this strange world, everything must be stated in 1's and 0's.

Now we can go a little deeper and consider the computer's computational model, giving us a crystal-clear idea of the behavior of the computer.

The Turing Machine

Alan Turing, the undisputed father of computer science, was born on June 23, 1912, in England. His contributions are so important that Computer Science's Nobel Prize is named the *Turing Award*. At the tender age of 24, Turning published a paper introducing a computing model, now known as the *Turing*

machine. A Turing machine is a theoretic device that models how a computer works.



A Turing machine can be described as follows.

1. The machine has a finite number of internal states. It is a *finite-state machine*.
2. Input to and output of the machine are given as symbols written on a tape that is not limited in length.
3. The collection of symbols, the machine's *alphabet*, is a finite set.
4. The symbol it is currently reading, together with its current state, determines the machine's action. An action may involve any and all of these: writing a new symbol on the tape, moving the read/write head forward or backward (or keeping it in place), entering a new state, and halting.
5. The Turing machine's finite-length program specifies its different states, and exact actions for different input at each state, including state transitions.

The Turing machine pictured starts working after loading a program. It has 12 states (0 through 11) and is currently in state 4.

Modern computers are shown to be equivalent to the Turning machine. The alphabet is the set {0, 1}, data and program stored in the main memory define its states and actions, and it performs I/O. Specifically:

1. The cpu and main memory are finite.
2. The tape models I/O. There is no length restriction on I/O.
3. The set of symbols {1,0} is finite.
4. The input values interpreted by the current program leads to specific actions.
5. Program actions certainly include making changes to memory, performing I/O, and choosing the next step.

Tasks that can be performed by a Turing machine are known as *computable* or *decidable*. Those that cannot be performed by a Turing machine are *uncomputable* or *undecidable*. This simply means that a task is computable if it is possible to devise an algorithm or write a computer program for it. Of course, creating effective and efficient algorithms is central to computer science. Another important topic of theoretic computer science is *computational complexity* that studies the amount of time required to solve certain classes of problems. Some computable problems can take a long time, longer than the Sun's life span, to finish.

Data Representation

Computers do everything using only 1's and 0's. It is exactly this simplicity that makes the modern computer so fast, precise, powerful, and flexible.

It turns out that bit patterns, a sequence of bits, can be used to represent numbers. Then we can, by convention (Unicode for example), assign a different number to each character, so they can be represented. Colors, sounds, and so on can be treated the same way. These then form the basis to represent documents, pictures, audio and video.

For example, in the RGB color system, a group of three bytes is used to represent red, green, and blue components of a color, Thus, a total of 256^3 different colors can be represented. See my previous CT blog *A World of 1's and 0's* for more information.

Red		(255,0,0)	Cyan		(0,255,255)
Green		(0,255,0)	Yellow		(255,255,0)
Blue		(0,0,255)	Magenta		(255,0,255)

A computer processes such data with programs. Furthermore, because programs themselves are also represented by 1's and 0's, they can be processed just like data.

Programs As Data

High-level programs are represented by characters. Such programs are transformed (by other computer programs known as compilers or an interpreters), and translated into machine language before being executed. A machine language programs, written in CPU instructions, can be run directly.

Thus, programs are also data in a computer. This means, a computer can transform and manipulate programs. A computer can even change and modify its own programming.

As we have mentioned, modern computers are *universal machines* because they can load and store programs. When you run a different program, the computer literally becomes a different machine, without modification or intervention at the hardware level. The single most important reason making this possible: the binary logical circuits using 1's and 0's.

Power of Abstraction

The Turing machine is a prime example of a central idea in CT—*abstraction*.

> *Strip away unimportant, unrelated, or irrelevant details from concrete instances, and focus on the essential core characteristics of the subject matter.*

Through abstraction, we often can make a complicated situation much clearer by peeling away unessential aspects that often can obscure the issue. The ability of abstraction allows us to distill the essence of an idea or concept and, therefore, gain a more profound understanding. For example, we can talk about adults, teenagers, children, liberals, conservatives, men, women, even numbers, odd numbers, and so on. But, not until we abstract to the concept of "set" and set membership, can we gain a clear concept and a language to discuss collections of objects in a general and rigorous manner.

Finally

The nature of digital hardware gives rise to a wondrous world of bits, bytes and words—a world that has only two letters in its alphabet. It being a stored-program machine makes it possible to download and launch apps at will.

The can-do machine can drive your car, pilot your plane, and bring spacecrafts to the Moon and back to the Earth. Sure there are things it can't do, such as falling in love or becoming a politician. In practice, it can do anything provided that a program can be developed for it.

We the humans, may want to borrow the computer's can-do spirit. That way we will be more flexible in the tasks we can, or are willing to, perform. This can make us more objective, brave, and willing to try new things—breaking any self-made cage and realizing our full potential.