

CSCI 120 Introduction to Computation

History of computing (draft)

Saad Mneimneh
Visiting Professor
Hunter College of CUNY

1 Early computing devices

Counting is the most basic form of computing. In order to compute, we must count! Of course now that we have numbers, counting is implicit. But there was a time when numbers did not exist. So what is the earliest computing device? The answer: the fingers! In the old days, before numbers were invented, people used their fingers to count. As larger quantities started to emerge, i.e. larger than the ten fingers could represent, small objects like pebbles were used to help counting. In fact, that's why we still use 10 as the base for our natural number system. We have 10 digits only 0,1,2,3,4,5,6,7,8, and 9. Whenever we reach 10 fingers, that's a pebble. Therefore, 10 is a pebble and no fingers, 23 is two pebbles and 3 fingers, etc... People needed not only to count, but also to compute cost of goods, for instance in trading. Therefore, counting devices were invented to make everyday calculations. The abacus is one of the many counting devices invented to count large numbers.

The history of the abacus has been traced as far back as the ancient Greek and Roman civilization, 500 B.C. The abacus as we know it today appeared around 1200 A.D. in china. This is the modern abacus. It is a device usually of wood (plastic in recent times) having a frame that holds rods with freely moving beads mounted on them. The following figure shows the modern abacus.



Figure 1: Abacus

The position of the beads represent numbers. This “machine” relies on a human operator to perform useful operations. The human must operate it

by moving the beads. The abacus alone is merely a data storage device (or a representation device to represent numbers). The abacus is still in use today in some shops in Asia. For more information about the abacus and for a tutorial on how to use it for addition and other arithmetic operations, see <http://www.ee.ryerson.ca/~elf/abacus/> . Here's a basic description of how we represent numbers on an abacus. Instead of using fingers, pebbles, rocks, and mountains, etc... we need not have a physical entity associated with each category. We simply abstract or generalize and think about the units, the tens, the hundreds, the thousands, etc... Each of these categories correspond to one of the rods on the abacus. But each rod mimics a human being. Therefore, for each rod, the two beads on the upper part represent the two hands, and the five beads on the lower part represent the five fingers on each hand. Beads are moved towards the center to contribute their values. A bead from the lower part contributes a 1. A bead from the upper part contributes a 5. For instance, the following is a representation for the number 9876543210.

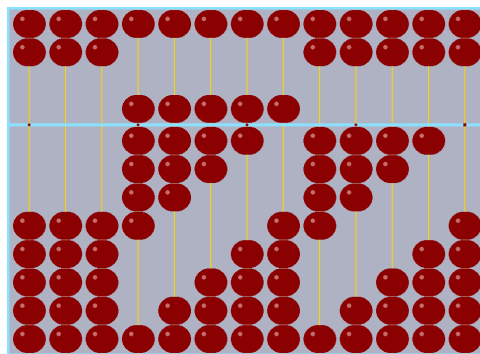


Figure 2: Representation of 9876543210

Although we can perform addition, subtraction, multiplication, division, square root, and cubic root using an abacus, the abacus is mainly used for addition and subtraction (the other operations are more complicated). For multiplication for instance, it is easier to use a slide rule. The slide rule is much recent than the abacus and performs multiplication, division, square root, and cubic roots much easier (while addition and subtraction are not supported by a slide rule).

In order to perform multiplication easily, the slide rule relies on the mathematical concept of logarithm. Logarithms were invented by the Scottish mathematician John Napier in 1614 (so the slide rule had to wait until then). So what are logarithms? First, fix a base, say base 10 (although this could be any number). The logarithm of the number 1 is denoted $\log 1$ and is always equal to 0. Moreover, $\log 10 = 1$ (generally, $\log b = 1$, where b is the base). Continuing, $\log 100 = 2$, $\log 1000 = 3$. And so on. Basically, if $\log x = y$, then $10 * 10 * \dots * 10$ (y times) is equal to x . Since $10 * 10 = 100$, then $\log 100 = 2$. Generally $\log b^x = x$, where b is the base. With 10 being our base, what happens to numbers that fall between 1 and 10? Well, any number between 1 and 10 is going to have a log value between 0 and 1. Similarly, any number between 10 and 100 is going to have a log value between 1 and 2. And so on. For example, $\log 3 = 0.4771$, $\log 30 = 1.4771$, $\log 300 = 2.4771$, and $\log 3000 = 3.4771$. As you can see, all these numbers fall exactly in the same place between the base 10 units. Mathematically speaking:

$$\log(a \times b) = \log a + \log b$$

$$\log(a/b) = \log a - \log b$$

For instance, to explain the above observation, $\log 300 = \log(3 \times 100) = \log 3 + \log 100 = \log 3 + \log(10 \times 10) = \log 3 + \log 10 + \log 10 = \log 3 + 1 + 1 = \log 3 + 2$. Every multiplication by 10 adds 1 to the log.

Therefore, working with logarithms would be a great time saver. Given two numbers a and b , we can look up $\log a$ and $\log b$, add them together, and then look for the number whose log is the sum.

Algorithm to multiply a and b

- Let $x = \log a$
- Let $y = \log b$
- Let $z = x + y$
- Let c be such that $\log c = z$
- c must be $a \times b$

However, there is still quite a lot of work required. We have to compute the log and, more importantly, find a number whose log is given. Soon after 1614, Edmund Gunter reduced this effort by drawing a number line in which the positions of numbers were proportional to their logs.

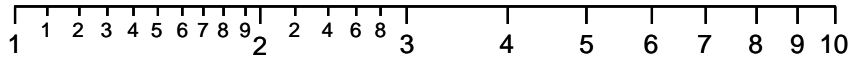


Figure 3: Log line

But what if the numbers are not in the range from 1 to 10? We can simply multiply or divide by 10 repeatedly until we get a number between 1 and 10. For instance, let's say we want to multiply 20 and 300. It is enough to multiply 2 and 3, and then multiply the result by 1000. Similarly, if we want to multiply 0.4 and 3, it is enough to multiply 3 and 4, and then divide the result by 10. Moreover, even if the result is more than 10, we can still obtain the correct answer. Here's an example for multiplying 4 and 3.

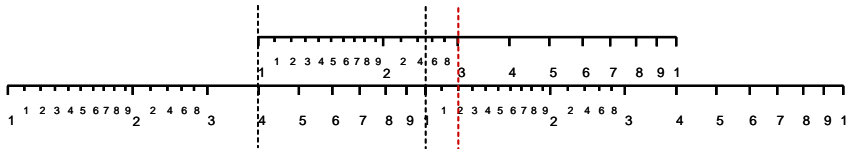


Figure 4: Multiplying 4 and 3 to get 12

Since we only have to sum two logarithms, it would be enough to just use two Gunter lines and walk them side by side. We need to use two dividers to keep track of the positioning. Mathematically, $\log 12 = \log(10 \times 1.2) = 1 + \log 1.2$.

Afterwards, William Oughtred simplified things further by taking two Gunger lines and sliding them relative to each other. The slide rule concept was born.

As a nice remark concerning the abacus and the slide rule, one of them can be categorized as *digital*, while the other as *analog*. Which is which? Well, theoretically speaking, the slide rule has an infinite number of states (the sliding part can be at any relative position from the fixed part). So that's analog (continuous). While the abacus has only a finite number of states determined by the possible positioning of the beads (each bead is either up or down). So that's digital (discrete). In general, digital devices can handle error better than analog devices (hold an abacus and a loose slide rule and imagine an earthquake).

2 The mechanical age

In more recent years, the design of computing machines was based on the technology of gears. Among those machines were the Pascaline in 1642 by Pascal (France); Leibniz's calculating machine in 1671 (Germany); the Difference Engine in 1786 by Mueller (Germany) which was forgotten and rediscovered in 1822 by Babbage (England); and the Analytical Engine in 1837 also by Babbage (using an idea from Joseph Jacquard in 1801 based on punched cards).

The Pascaline was built to perform only additions. But it can be used to perform subtractions too. Multiplication and division are not possible. The following figure shows the conceptual design of the Pascaline.

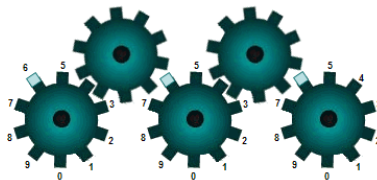


Figure 5: The Pascaline (concept, not actual design)

One complete rotation of a gear corresponds to 1/10 of a rotation for the gear to its left. Therefore, when completing a full rotation, the gear on the left moves one pin. This is the mechanism used to propagate a carry. Before starting a calculation, the gears are reset to zero. This is done manually. To add two numbers, the first number is entered, digit by digit, by rotating the appropriate gears. Then, the second number is entered similarly by appropriate rotations of the corresponding gears. This will cause the gears to display the sum of the two numbers. This can now be continued to add a third number, and so on. For instance, if we want to add 7 and 7, the first gear is set to position 7. Then, when rotated 7 pins again, the first gear will cause a single pin movement in the second gear (the tens) making it display 1. At the same time, the first gear will indicate 4.

Unfortunately, the gears do not turn backward to perform subtraction. But it is possible to perform subtractions using 9's complement. Here's how. Assume we have 3 gears/digits (but the algorithm can be extended to any number of gears/digits). Let's say we want to compute $146 - 65$. This is $999 - 999 + 146 - 65 = 999 - (999 - 146 + 65) = 999 - ((999 - 146) + 65) = 999 - (853 + 65) = 999 - 918 = 081$. Therefore, if we can complement 146 to obtain 853, add that to 65 using the Pascaline to obtain 918, then complement

the result to obtain 081. Why 9's complement? We could have added and subtracted any number other than 999. First, it is simple to find the 9's complement on the fly. Second, using 999 guarantees that we do not get negative numbers when we subtract (because 999 is the largest number that can be represented by the machine).

Algorithm to compute $a - b$ using a Pascaline

- Let $x = 9\dots9 - a$ (on the fly)
- Let $y = x + b$ using Pascaline
- Let $z = 9\dots9 - y$ (on the fly)
- z must be $a - b$

Similar to the Pascaline, Leibniz's machine had its algorithms firmly embedded in its mechanical architecture, but it offered a variety of arithmetic operations from which a human operator could select in advance. Leibniz also described the binary number system (i.e. base 2) which is now used in all digital devices and modern computers (more about this binary system later). However, up to the 1940s, many machines were based on the harder to implement decimal system (i.e. base 10).

Here's a comparison among the three devices discussed so far:

	Abacus	Slide Rule	Pascaline
Representation	placement of beads	amount of sliding	positions of gears
Algorithm	none	multiplication	addition
Program	none	encoded into log scale	encoded by rotation of gears
User	provide input and perform algorithm	only provide input	only provide input
States	finite (how many?)	infinite	finite

3 More mechanics

The Difference Engine (of which only a demonstration model was constructed) is also a mechanical device that was designed to tabulate polynomial functions. Logarithms and other mathematical functions can be approximated by polynomials; therefore, this machine is more general than it appears at first. So what is a polynomial? A polynomial is a sum of different exponents of a single variable x . For example, $p(x) = 2x^2 - 3x + 2$ is a polynomial in x of degree 2 (the highest exponent). If $x = 0$, $p(0) = 2(0^2) - 3(0) + 2 = 2$. If $x = 1$, then $p(1) = 2(1^2) - 3(1) + 2 = 2 - 3 + 2 = 1$. If $x = 2$, then $p(2) = 2(2^2) - 3(2) + 2 = 4$. The principle behind the difference engine is Newton's (the second smartest person in history) method of differences. This can be illustrated by an example using our polynomial $p(x)$.

	1 st difference	2 nd difference
$p(0) = 2$		
	$1 - 2 = -1$	
$p(1) = 1$		$3 - (-1) = 4$
	$4 - 1 = 3$	
$p(2) = 4$		$7 - 3 = 4$
	$11 - 4 = 7$	
$p(3) = 11$		$11 - 7 = 4$
	$22 - 11 = 11$	
$p(4) = 22$		
\vdots		

Note that the value in the last column is always a constant (in this case 4). This is no coincidence! In fact, this is true for any polynomial of degree n , the n^{th} difference is always constant (hence the $(n + 1)^{\text{th}}$ difference is always zero). Those who have taken calculus may observe that this is based on the fact that the n^{th} derivative of a function of degree n is a constant. This fact means that we can now compute the value of $p(x)$ for any x . For instance, let's compute $p(5)$ without actually using the polynomial, i.e. without performing any multiplication. We work backward. We start with the second difference being 4 for sure. This means the first difference will be $11 + 4 = 15$. Therefore, $p(5) = 22 + 15 = 37$. Let's verify that: $p(5) = 2(5^2) - 3(5) + 2 = 2(25) - 3(5) + 2 = 50 - 15 + 2 = 37$.

Babbage's difference engine could hold 7 numbers of 31 decimal digits each and could thus tabulate 7th degree polynomials to that precision.

4 Programming

So far, all these mechanical machines are non-programmable (each does only one specific thing). The Analytical Engine of Babbage was programmable. The program is fed to the machine via holes in paper. This idea by itself was not originated by Babbage. He got it from Jacquard, who, in 1801, had developed a weaving loom in which the steps to be performed during the weaving process were determined by a pattern of holes in cards.



Figure 6: Punched card (similar to bits in memory)

The Analytical Engine consists of a number of piles, each containing a number of disks, say n . Each disk is numbered around the circumference from 0 to 9 (as in the Pascaline). Therefore, each pile can represent a number with n digits. The machine can perform addition, subtraction, multiplication, and division. It is possible to specify three piles and an operation, then the machine will perform that operation using the two numbers stored on the first two piles as operands, and store the result in the third pile. The machine is designed in a way that a sequence of such operations can be specified in advance via punched cards. For instance, conceptually speaking, a program can look like Figure 7.

Columns on which are inscribed the primitive data	Cards of the operations			Variable cards			Statement of results
	Number of the operations	No. of the Operation-cards	Nature of each operation	Columns acted on by each operation	Columns that receive the result of each operation	Indication of change of value on any column	
${}^1V_0 = m$	1	1	\times	${}^1V_0 \times {}^1V_4 = {}^1V_6 \dots\dots$	${}^1V_6 \dots\dots$	$\left\{ \begin{array}{l} {}^1V_0 = {}^1V_0 \\ {}^1V_4 = {}^1V_4 \end{array} \right\}$	${}^1V_6 = mn'$
${}^1V_1 = n$	2	"	\times	${}^1V_3 \times {}^1V_1 = {}^1V_7 \dots\dots$	${}^1V_7 \dots\dots$	$\left\{ \begin{array}{l} {}^1V_3 = {}^1V_3 \\ {}^1V_1 = {}^1V_1 \end{array} \right\}$	${}^1V_7 = m'n$
${}^1V_2 = d$	3	"	\times	${}^1V_2 \times {}^1V_4 = {}^1V_8 \dots\dots$	${}^1V_8 \dots\dots$	$\left\{ \begin{array}{l} {}^1V_2 = {}^1V_2 \\ {}^1V_4 = {}^0V_4 \end{array} \right\}$	${}^1V_8 = dn'$
${}^1V_3 = m'$	4	"	\times	${}^1V_5 \times {}^1V_1 = {}^1V_9 \dots\dots$	${}^1V_9 \dots\dots$	$\left\{ \begin{array}{l} {}^1V_5 = {}^1V_5 \\ {}^1V_1 = {}^0V_1 \end{array} \right\}$	${}^1V_9 = d'n$
${}^1V_4 = n'$	5	"	\times	${}^1V_0 \times {}^1V_5 = {}^1V_{10} \dots\dots$	${}^1V_{10} \dots\dots$	$\left\{ \begin{array}{l} {}^1V_0 = {}^0V_0 \\ {}^1V_5 = {}^0V_5 \end{array} \right\}$	${}^1V_{10} = d'm$
${}^1V_5 = d'$	6	"	\times	${}^1V_2 \times {}^1V_3 = {}^1V_{11} \dots\dots$	${}^1V_{11} \dots\dots$	$\left\{ \begin{array}{l} {}^1V_2 = {}^0V_2 \\ {}^1V_3 = {}^0V_3 \end{array} \right\}$	${}^1V_{11} = dm'$
	7	2	$-$	${}^1V_6 - {}^1V_7 = {}^1V_{12} \dots\dots$	${}^1V_{12} \dots\dots$	$\left\{ \begin{array}{l} {}^1V_6 = {}^0V_6 \\ {}^1V_7 = {}^0V_7 \end{array} \right\}$	${}^1V_{12} = mn' - m'n$
	8	"	$-$	${}^1V_8 - {}^1V_9 = {}^1V_{13} \dots\dots$	${}^1V_{13} \dots\dots$	$\left\{ \begin{array}{l} {}^1V_8 = {}^0V_8 \\ {}^1V_9 = {}^0V_9 \end{array} \right\}$	${}^1V_{13} = dn' - d'n$
	9	"	$-$	${}^1V_{10} - {}^1V_{11} = {}^1V_{14} \dots\dots$	${}^1V_{14} \dots\dots$	$\left\{ \begin{array}{l} {}^1V_{10} = {}^0V_{10} \\ {}^1V_{11} = {}^0V_{11} \end{array} \right\}$	${}^1V_{14} = d'm - dm'$
	10	3	\div	${}^1V_{13} \div {}^1V_{12} = {}^1V_{15} \dots\dots$	${}^1V_{15} \dots\dots$	$\left\{ \begin{array}{l} {}^1V_{13} = {}^0V_{13} \\ {}^1V_{12} = {}^1V_{12} \end{array} \right\}$	${}^1V_{15} = \frac{dn' - d'n}{mn' - m'n} = x$
	11	"	\div	${}^1V_{14} \div {}^1V_{12} = {}^1V_{16} \dots\dots$	${}^1V_{16} \dots\dots$	$\left\{ \begin{array}{l} {}^1V_{14} = {}^0V_{14} \\ {}^1V_{12} = {}^0V_{12} \end{array} \right\}$	${}^1V_{16} = \frac{d'm - dm'}{mn' - m'n} = y$
1	2	3	4	5	6	7	8

Figure 7: Example program

The Analytical Engine may be considered the first modern computer in concept because it exposes a lot of computer architecture aspects that were rediscovered in the 20th century. For instance, the pile of disks is like the register in the Central Processing Unit (CPU) of a modern computer. Registers store data in the form of bits (later on this). Moreover, most arithmetic and logic operations in the CPU take two registers as operands and produce the result in a third register.

In 1842, an Italian mathematician, Louis Menebrea, published a memoir in French on the subject of the Analytical Engine. Babbage enlisted Augusta Ada Byron (countess of Lovelace, daughter of the romantic poet Lord Byrons) as translator for the memoir, and during a nine-month period in 1842-1843, she worked on the article and a set of Notes she appended to it. These are the source of her enduring fame because they demonstrated how the Analytical Engine could be programmed to perform various computations. She is often identified today as the world's first programmer. In the 1970s, the US department of defense named a programming language in her honor, Ada. For more information on the Analytical Engine and this subject, see "A Sketch of the Analytical Engine" at <http://www.fourmilab.ch/babbage/>.

Jacquard's idea on punched cards was also used by Herman Hollerith (1860 - 1929), who applied the concept of representing information as holes in paper cards to speed up the tabulation process in the 1890 US census. It was this work that led to the creation of IBM (originally named Tabulating Machine Corporation). Punched cards were still in use until the 1990s, but in very limited set of applications, e.g. toll tickets. Another use of punched cards that we see today is in coffee places (buy 10 coffees get 1 free).

Note: Saying that a machine is programmable does not mean literally that the machine does many things. It is the program that keeps on changing. But the machine has the ability to take a program as an input and act upon it. In fact, that's the only thing that the machine can do. For instance, from the perspective of the Analytical Engine, the machine is only doing one thing: looking at those punched cards and mechanically acting based on the presence/absence of the holes. Therefore, the machine is only an interpreter for the program. However, such a combination of program/interpreter creates a variety in what the machine seems to be doing.