MISA.Z ROKID https://developer.rokid.com/

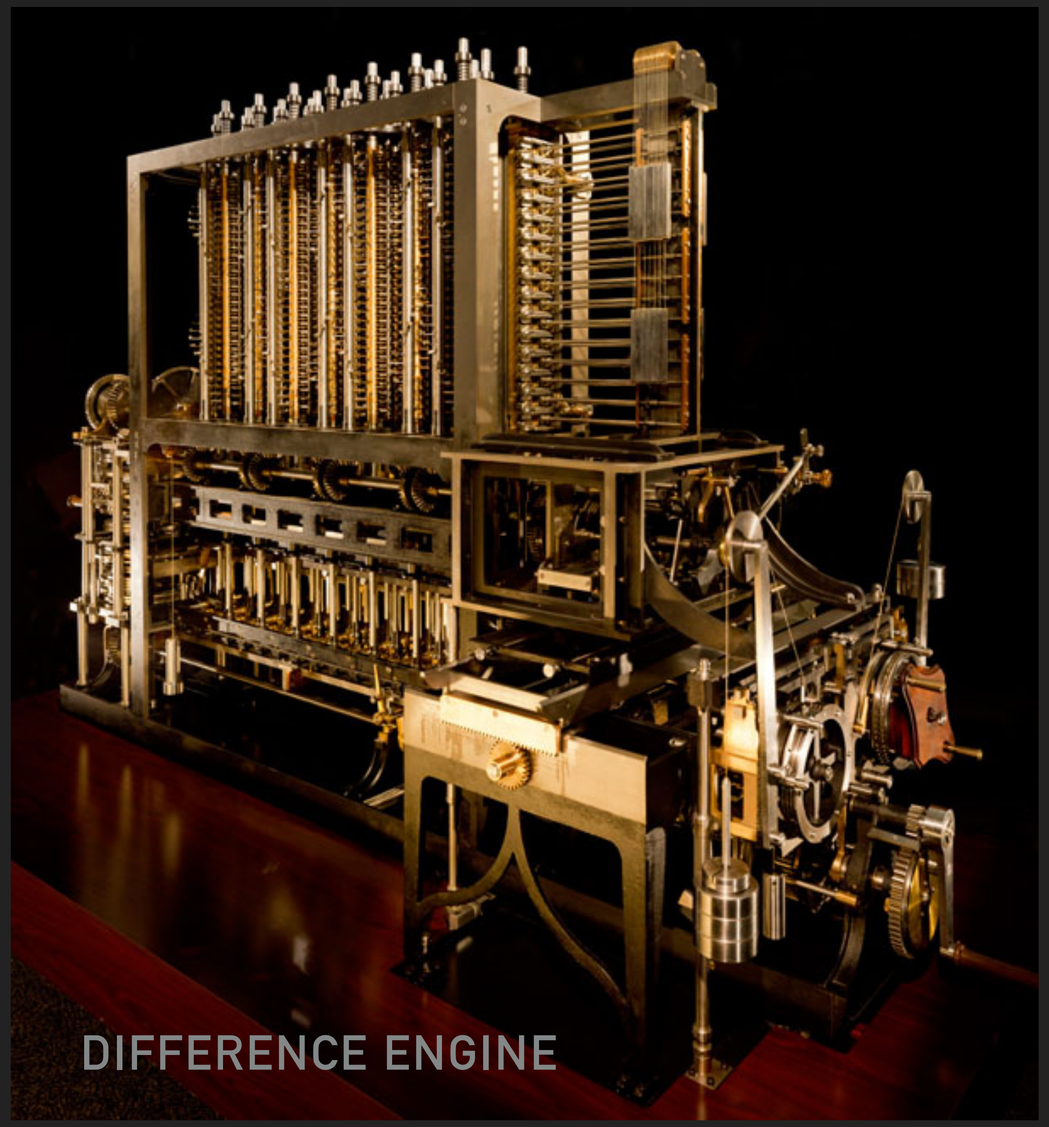
OPERATING SYSTEM

MATH GAME :)

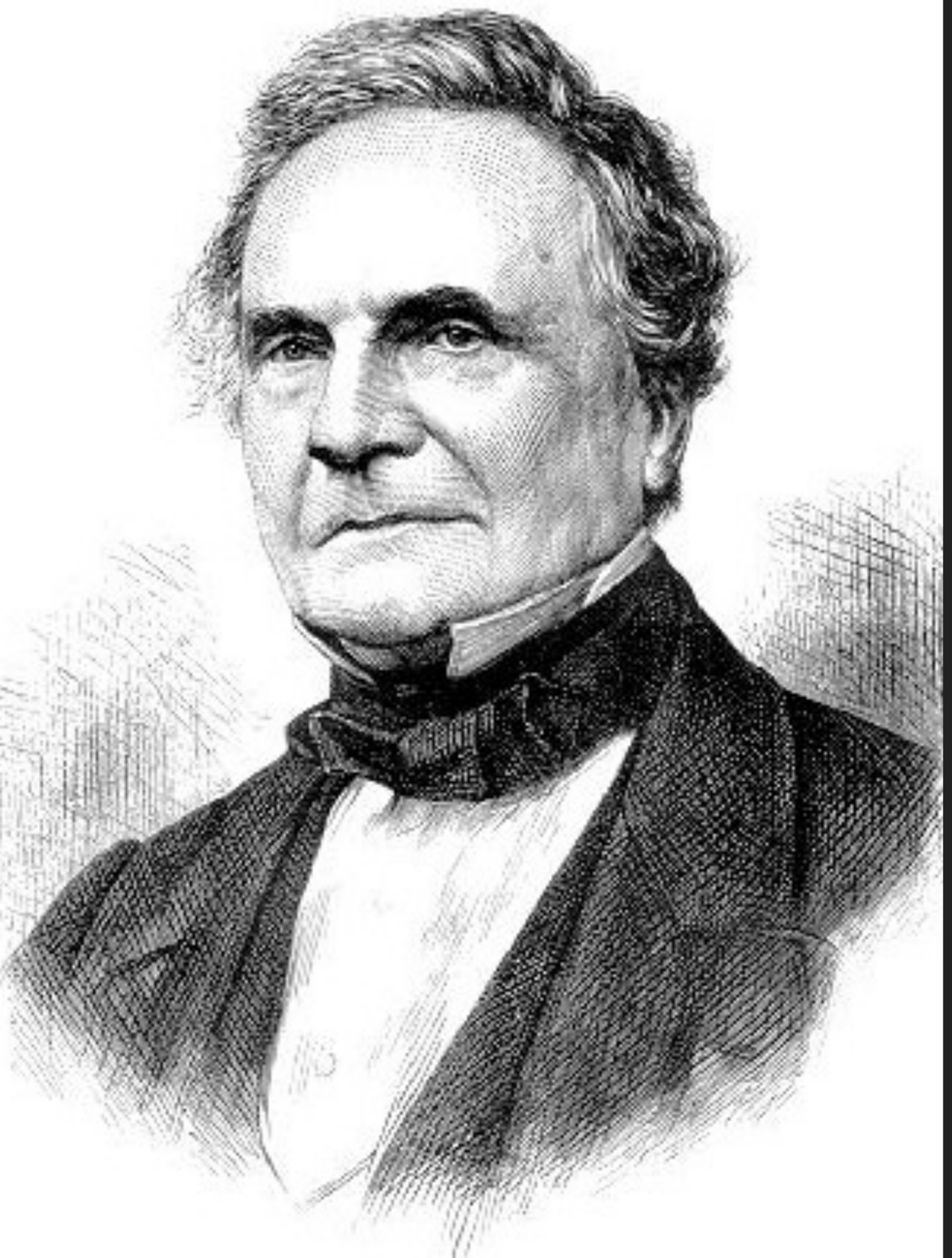
- To f(x) = x^A + B, e.g. f(x) = x^2 + 6
- f(1) = 1^2 + 6 = 7
- f(2) = 2^2 + 6 = 10 d(2) = f(2) - f(1) = 10 - 7 = 3
- f(3) = 3^2 + 6 = 15 d(3) = f(3) - f(2) = 15 - 10 = 5 d(3) - d(2) = 5 - 3 = 2
- f(4) = 4^2 + 6 = 22 d(4) = f(4) - f(3) = 22 - 15 = 7 d(4) - d(3) = 7 - 5 = 2 DIFF = 2
- f(5) = 5^2 + 6 = 31
- f(5) = d(5) + f(4) = d(4) + 2 + 22 = 31
- f(x) = d(x) + f(x-1) = d(x-1) + DIFF + f(x-1)

MECHANICAL COMPUTING

- Specific task for each machine
- Input & Output
- Non-software
- Non-programable



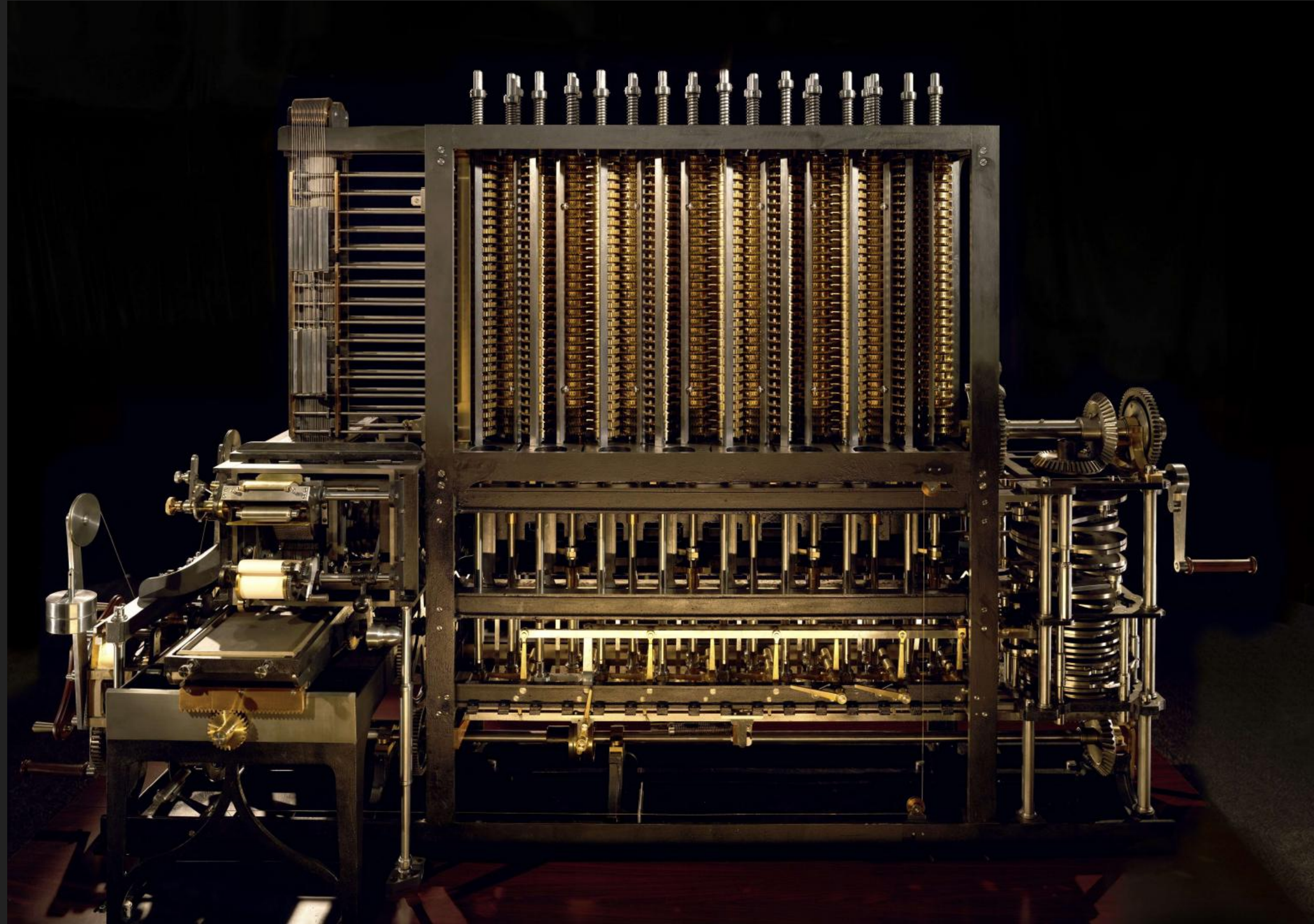
DIFFERENCE ENGINE



MACHINE COMPUTING

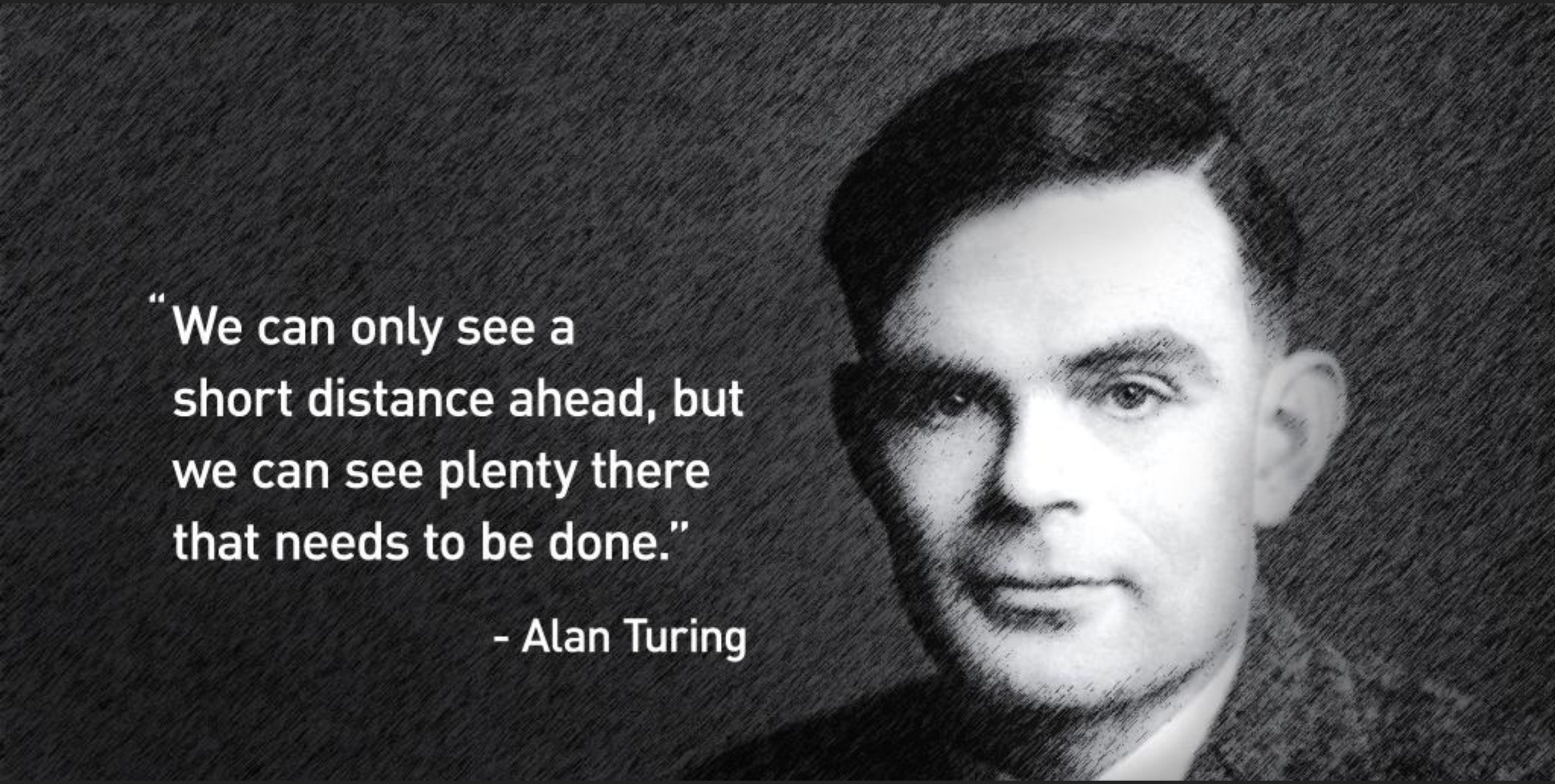
Charles Babbage
1791–1871

ANALYTICAL ENGINE



MACHINE COMPUTING

Ada Lovelace
1815–1852


"We can only see a
short distance ahead, but
we can see plenty there
that needs to be done."
- Alan Turing

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

Circular and circle-free machines.

If a computing machine never writes down more than a finite number of symbols of the first kind, it will be called *circular*. Otherwise it is said to be *circle-free*.

A machine will be circular if it reaches a configuration from which there is no possible move, or if it goes on moving, and possibly printing symbols of the second kind, but cannot print any more symbols of the first kind. The significance of the term "circular" will be explained in § 8.

Computable sequences and numbers.

A sequence is said to be computable if it can be computed by a circle-free machine. A number is computable if it differs by an integer from the number computed by a circle-free machine.

We shall avoid confusion by speaking more often of computable sequences than of computable numbers.

3. *Examples of computing machines.*

I. A machine can be constructed to compute the sequence 010101 .... The machine is to have the four m-configurations "$\mathfrak{b}$", "$\mathfrak{c}$", "$\mathfrak{f}$", "$\mathfrak{e}$" and is capable of printing "0" and "1". The behaviour of the machine is described in the following table in which "R" means "the machine moves so that it scans the square immediately on the right of the one it was scanning previously". Similarly for "L". "E" means "the scanned symbol is erased" and "P" stands for "prints". This table (and all succeeding tables of the same kind) is to be understood to mean that for a configuration described in the first two columns the operations in the third column are carried out successively, and the machine then goes over into the m-configuration described in the last column. When the second column is left blank, it is understood that the behaviour of the third and fourth columns applies for any symbol and for no symbol. The machine starts in the m-configuration $\mathfrak{b}$ with a blank tape.

Configuration		*Behaviour*	
m-config.	*symbol*	*operations*	*final m-config.*
$\mathfrak{b}$	None	$P0, R$	$\mathfrak{c}$
$\mathfrak{c}$	None	R	$\mathfrak{c}$
$\mathfrak{e}$	None	$P1, R$	$\mathfrak{f}$
$\mathfrak{f}$	None	R	$\mathfrak{b}$

If (contrary to the description in § 1) we allow the letters L, R to appear more than once in the operations column we can simplify the table considerably.

m-config.	*symbol*	*operations*	*final m-config.*
$\mathfrak{b}$	None	$P0$	$\mathfrak{b}$
	0	$R, R, P1$	$\mathfrak{b}$
	1	$R, R, P0$	$\mathfrak{b}$

II. As a slightly more difficult example we can construct a machine to compute the sequence 00101101101111011111 .... The machine is to be capable of five m-configurations, viz. "$\mathfrak{o}$", "$\mathfrak{q}$", "$\mathfrak{p}$", "$\mathfrak{f}$", "$\mathfrak{b}$" and of printing "$\mathfrak{d}$", "x", "0", "1". The first three symbols on the tape will be "$\mathfrak{d}\mathfrak{d}0$"; the other figures follow on alternate squares. On the intermediate squares we never print anything but "x". These letters serve to "keep the place" for us and are erased when we have finished with them. We also arrange that in the sequence of figures on alternate squares there shall be no blanks.

Configuration		*Behaviour*	
m-config.	*symbol*	*operations*	*final m-config.*
$\mathfrak{b}$		$P\mathfrak{d}, R, P\mathfrak{d}, R, P0, R, R, P0, L, L$	$\mathfrak{o}$
$\mathfrak{o}$	1	R, Px, L, L, L	$\mathfrak{o}$
	0		$\mathfrak{q}$
$\mathfrak{q}$	Any (0 or 1)	R, R	$\mathfrak{q}$
	None	$P1, L$	$\mathfrak{p}$
$\mathfrak{p}$	x	E, R	$\mathfrak{q}$
	$\mathfrak{d}$	R	$\mathfrak{f}$
	None	L, L	$\mathfrak{p}$
$\mathfrak{f}$	Any	R, R	$\mathfrak{f}$
	None	$P0, L, L$	$\mathfrak{o}$

To illustrate the working of this machine a table is given below of the first few complete configurations. These complete configurations are described by writing down the sequence of symbols which are on the tape,

https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf

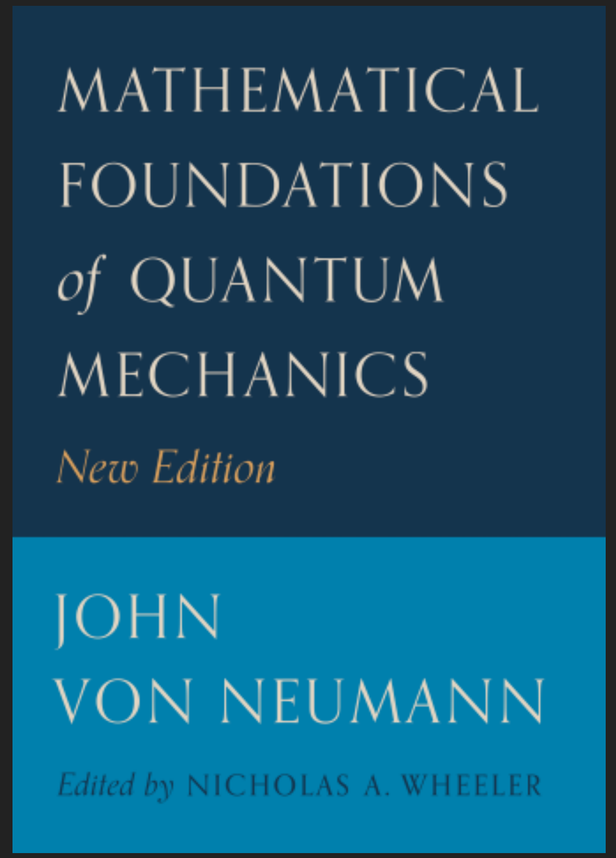
If people do not believe that mathematics is simple, it is only because they do not realize how complicated life is.

— John von Neumann —

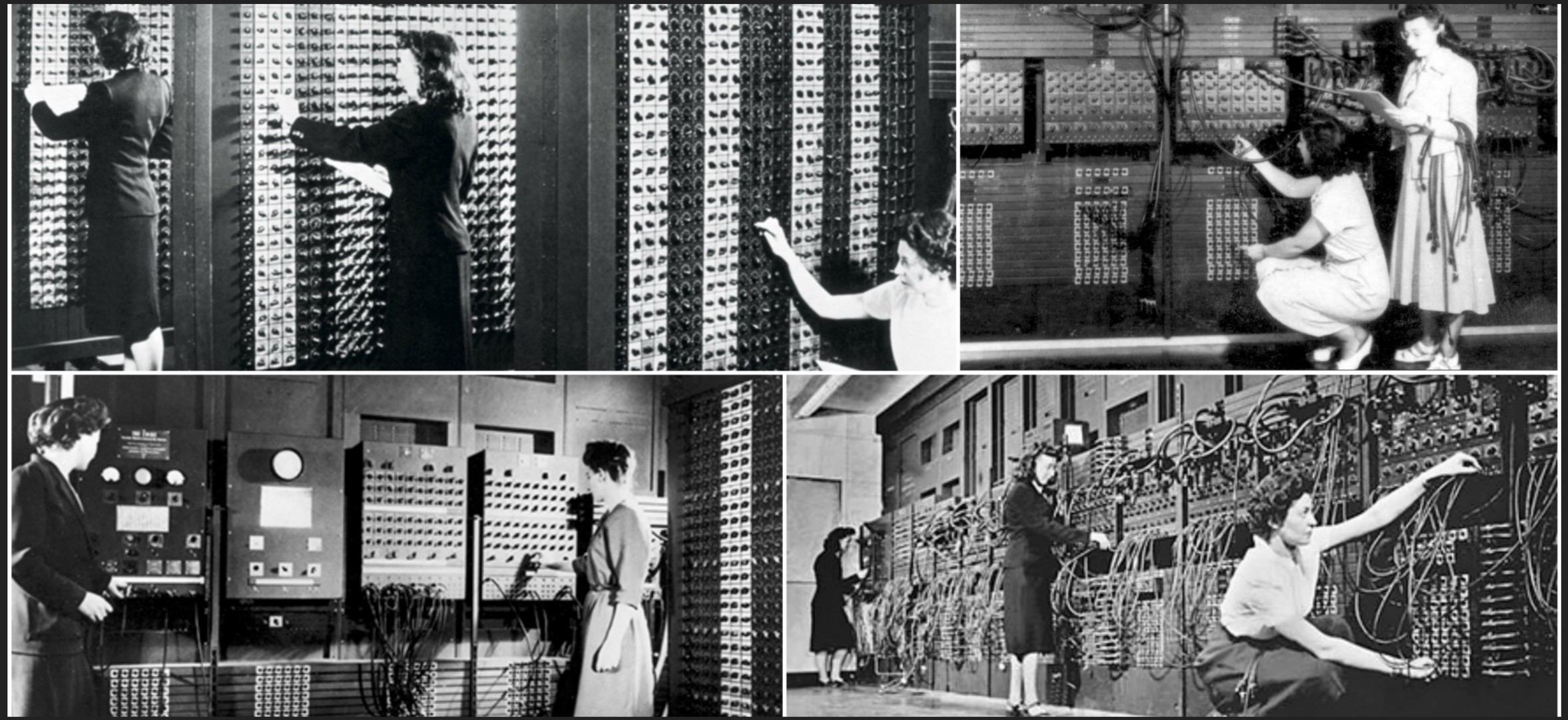
AZ QUOTES

PROGRAMABLE COMPUTER

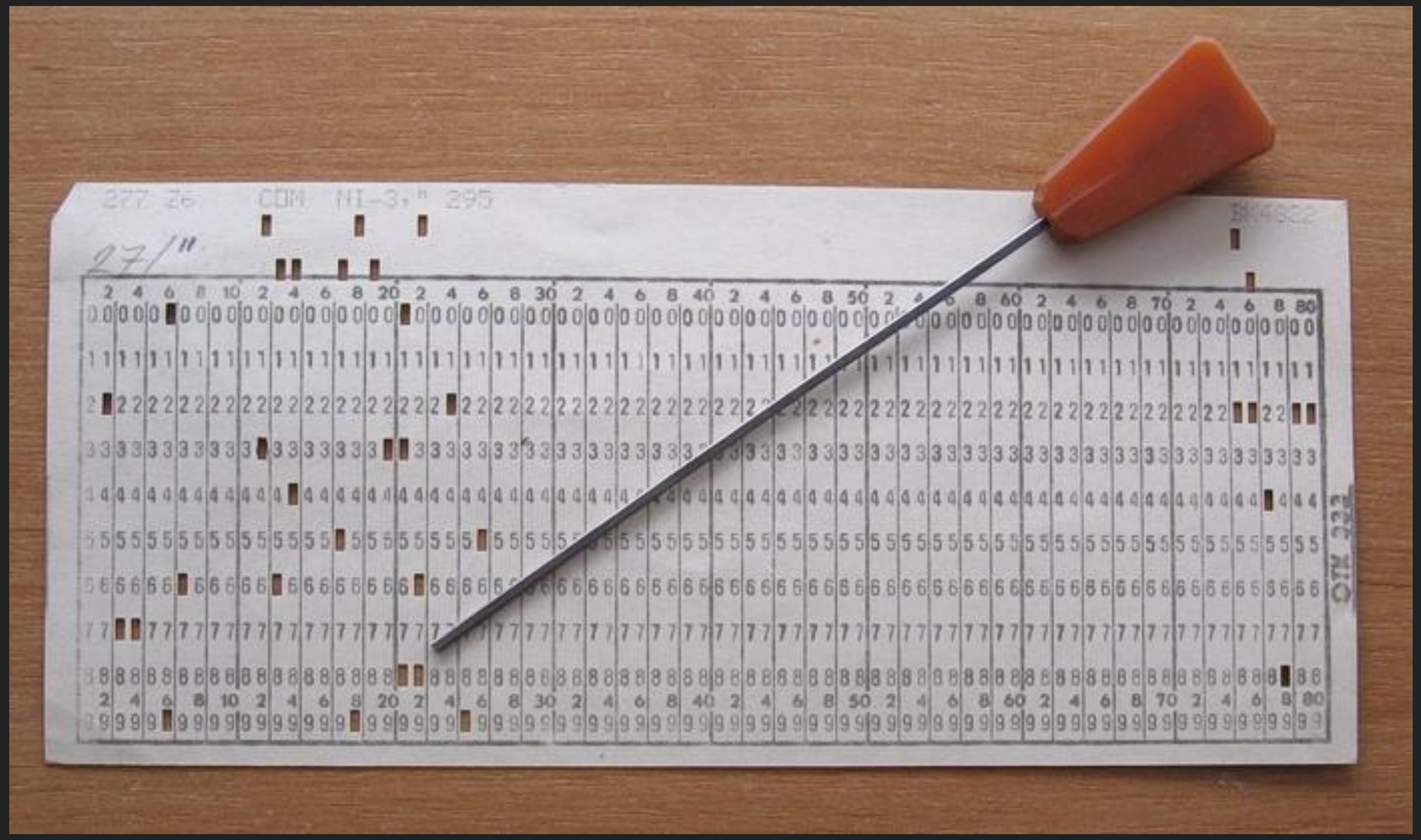
In 1943, the ENIAC project was created for the US Army during World War II. Unlike any other previous computer project, the first all-electronic, programmable computer was successfully programmed by six intelligent young women. Kathleen “Kay” McNulty Mauchly Antonelli, “Betty” Jean Jennings Bartik, Frances Elizabeth “Betty” Holberton, Marlyn Wescoff Meltzer, Frances Bilas Spence, and Ruth Lichterman Teitelbaum.



PROGRAMABLE COMPUTER



PROGRAMABLE COMPUTER

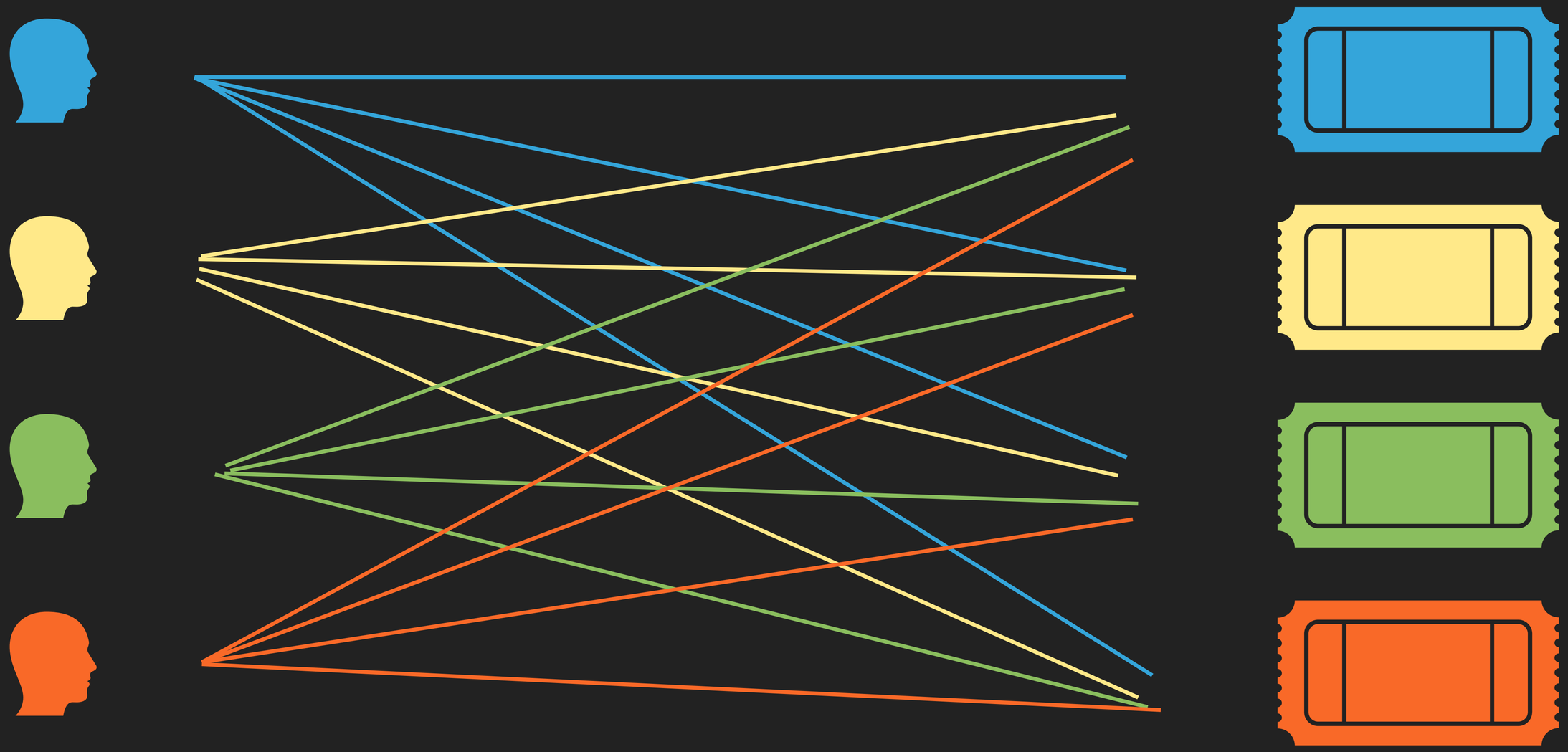


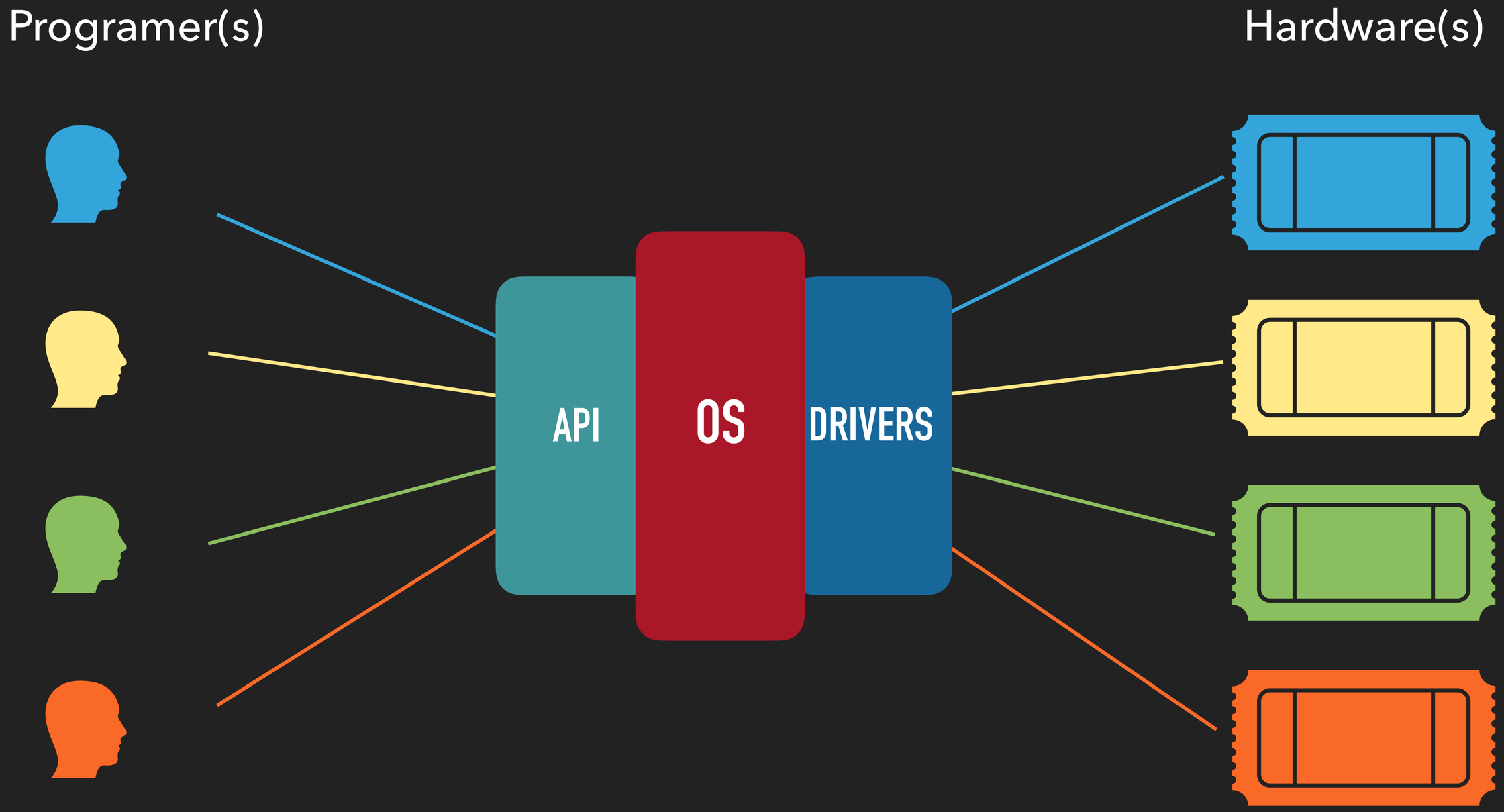
WHY DO WE NEED OPERATING SYSTEMS?

In earlier day’s user had to design the application according to the internal structure of the hardware. Operating System was needed to enable the user to design the application without concerning the details of the computer’s internal structure. In general the boundary between the hardware & software is transparent to the user.

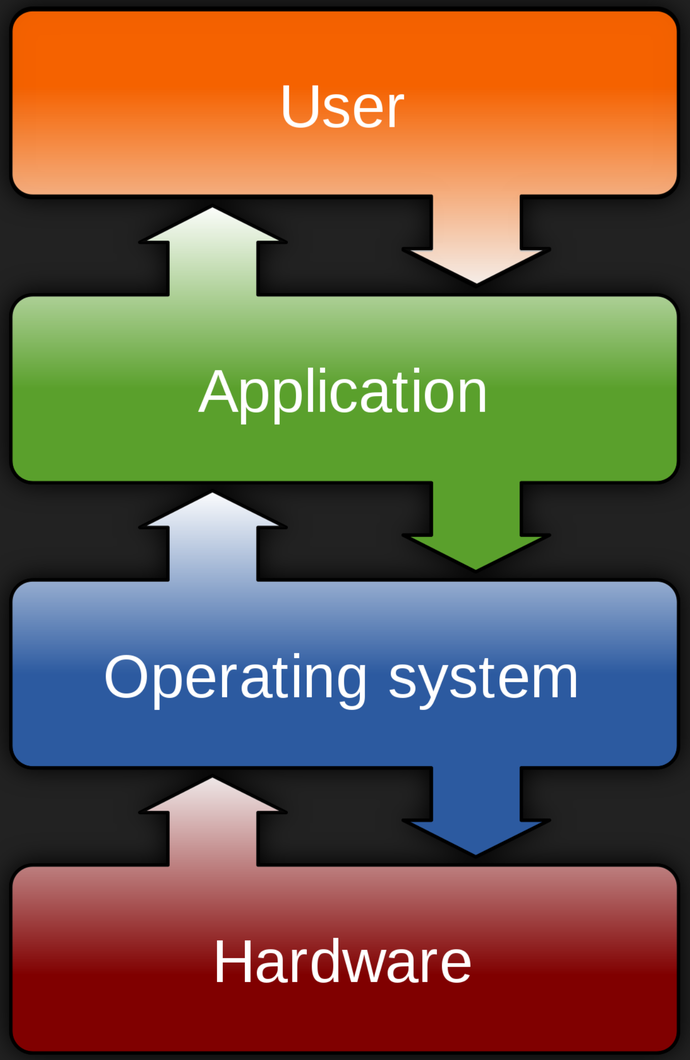
Programer(s)

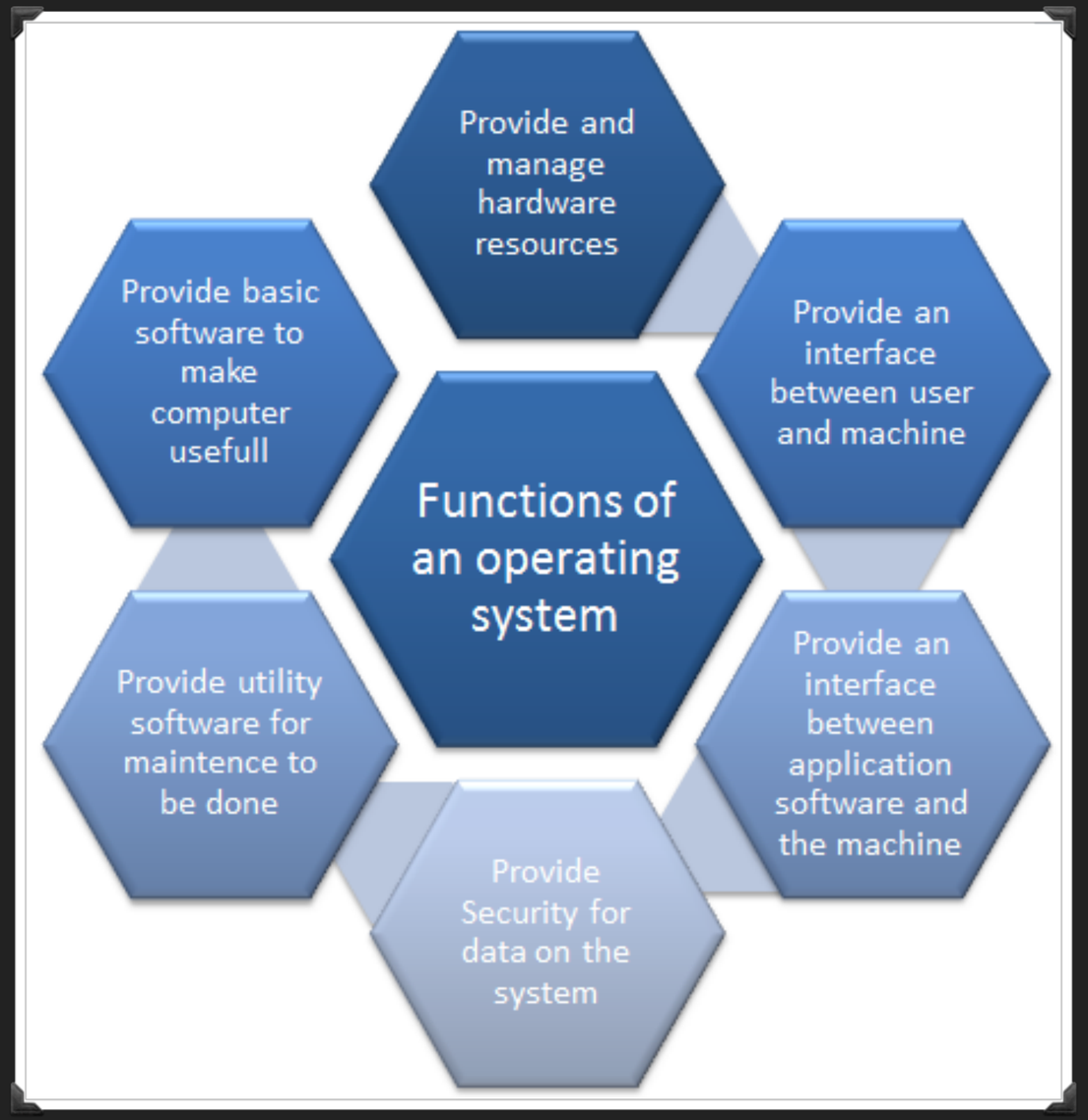
Hardware(s)




Programer(s)
Hardware(s)
API
OS
DRIVERS

OPERATING SYSTEM





THE FIRST OPERATING SYSTEM

(60th-70th years)

Each manufacturer had such proprietary operating system, which differed from each other completely, and the operator had always relearn everything from scratch.

UNIX

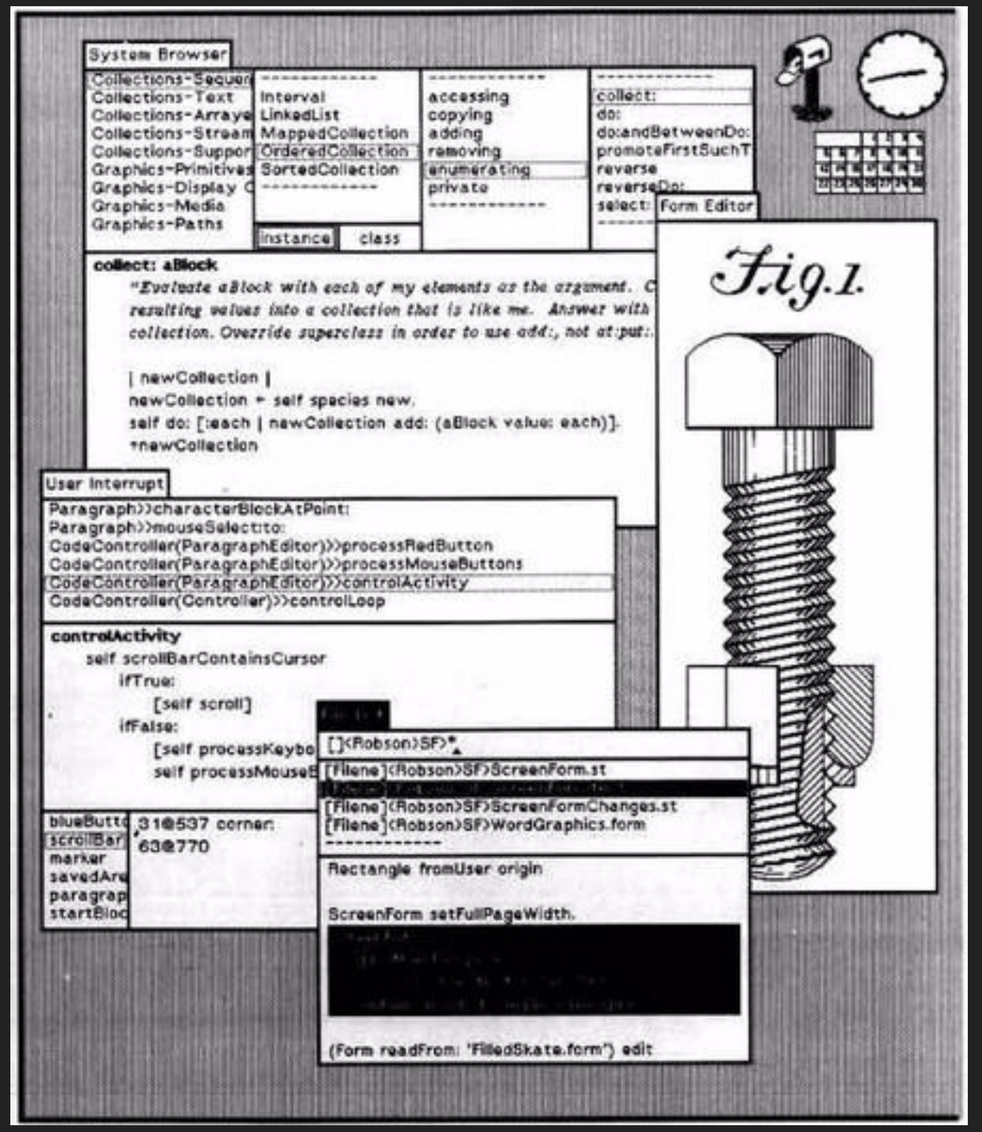
1969 – UNIX

He laid the foundations of today's operating systems (Linux, Mac OS X, NeXTSTEP, OpenBSD ..)

(Dennis Ritchie and Ken Thompson are developing UNIX)

XEROX ALTO

(1973) Xerox Alto was not a commercially very successful, his proposal, but affected the whole generation of computers and operating systems. It was the first computer using a mouse and a fully graphical interface.



APPLE II

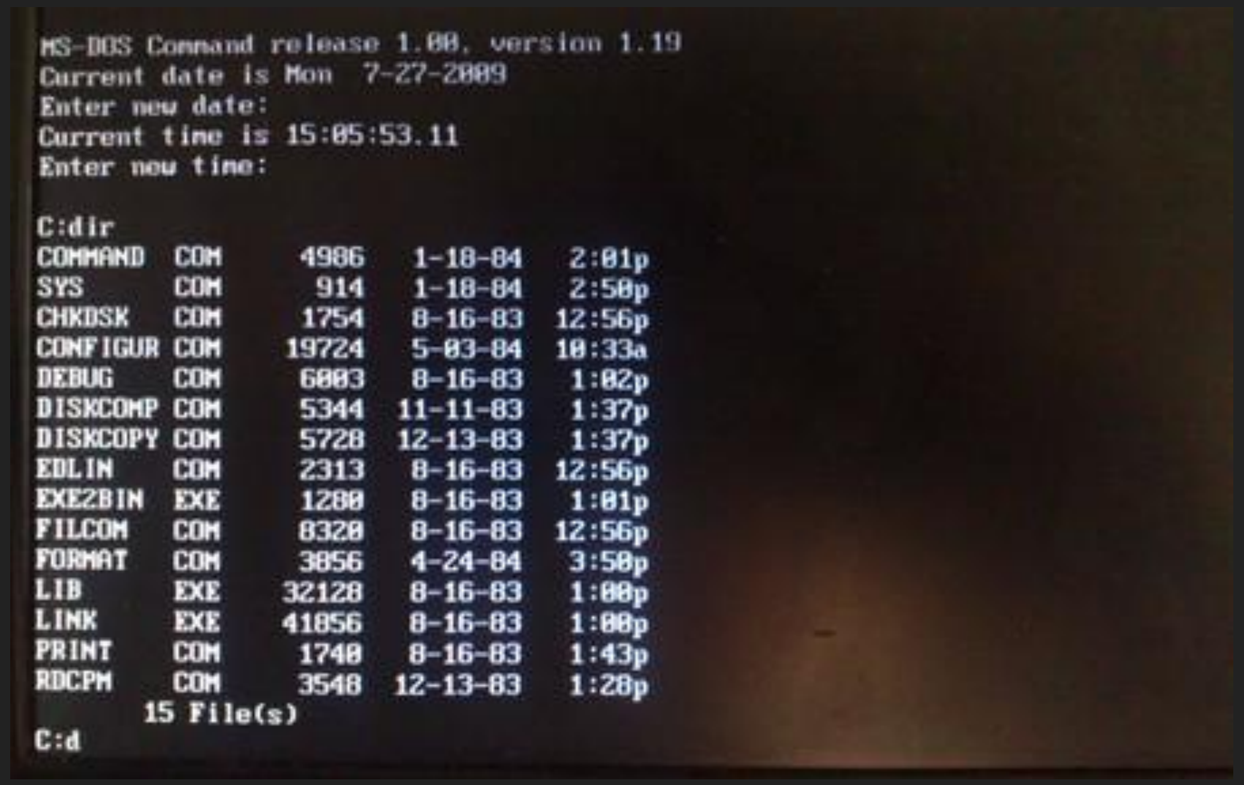
Steve Wozniak Apple II became one of the first massively expanded home computers.

Apple DOS (Disk Operating System) first appeared on Apple II computers a year after its launch, and strongly influenced by the way it evolved operating systems over the next decade.

MS DOS

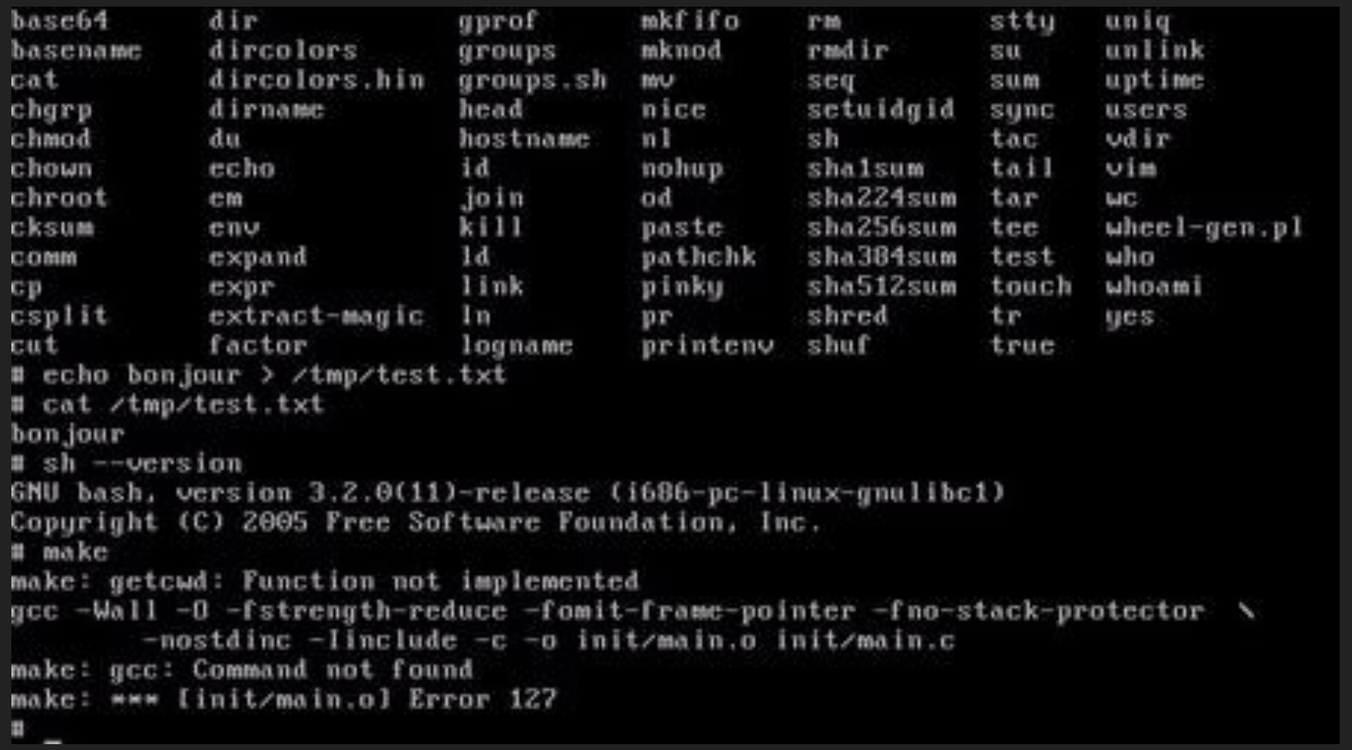
1981 - MS DOS

System Which of the then little-known service firms on behalf of Microsoft has done a multinational corporation. He first appeared on IBM PC .



LINUX

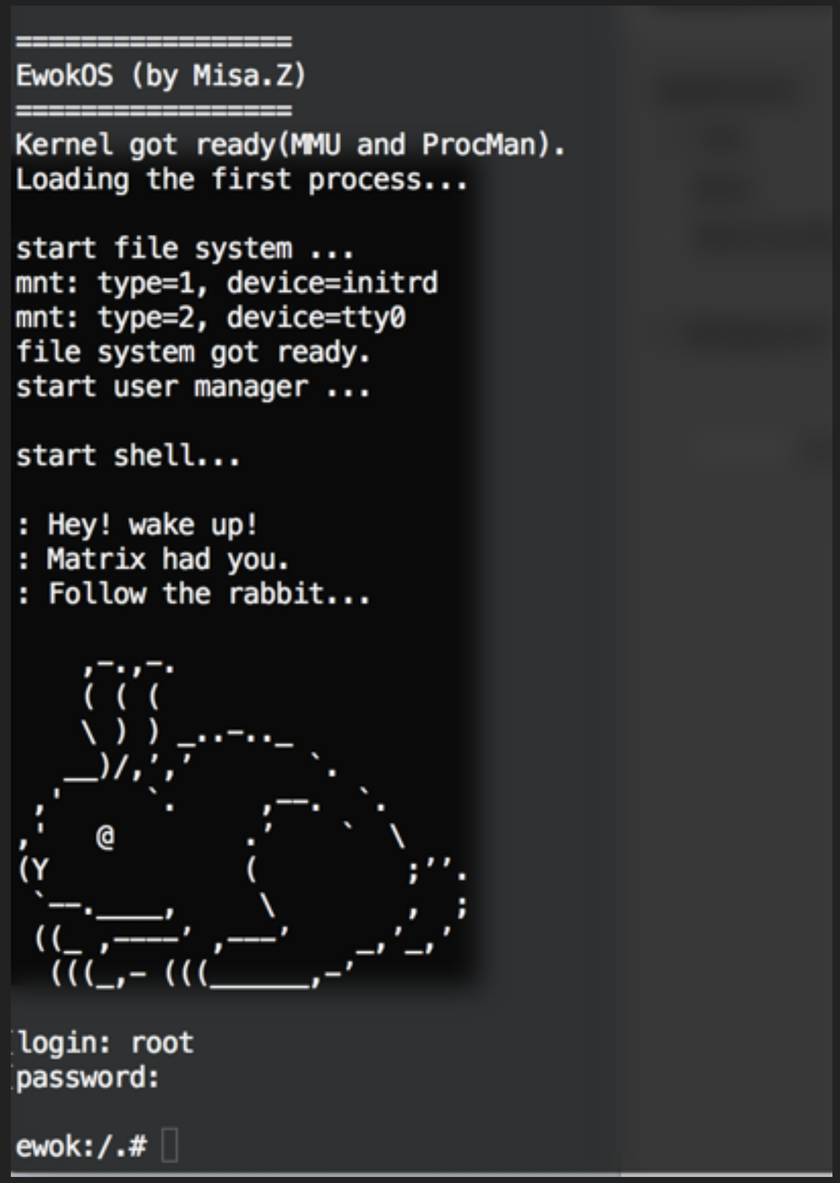
Began with the commencement of a personal project by Finnish student Linus Torvalds to create a new free operating system kernel. Since then, the resulting Linux kernel has been marked by constant growth throughout its history. Since the initial release of its source code in 1991.



CODE STUDY

https://github.com/MisaZhu/EwokOS

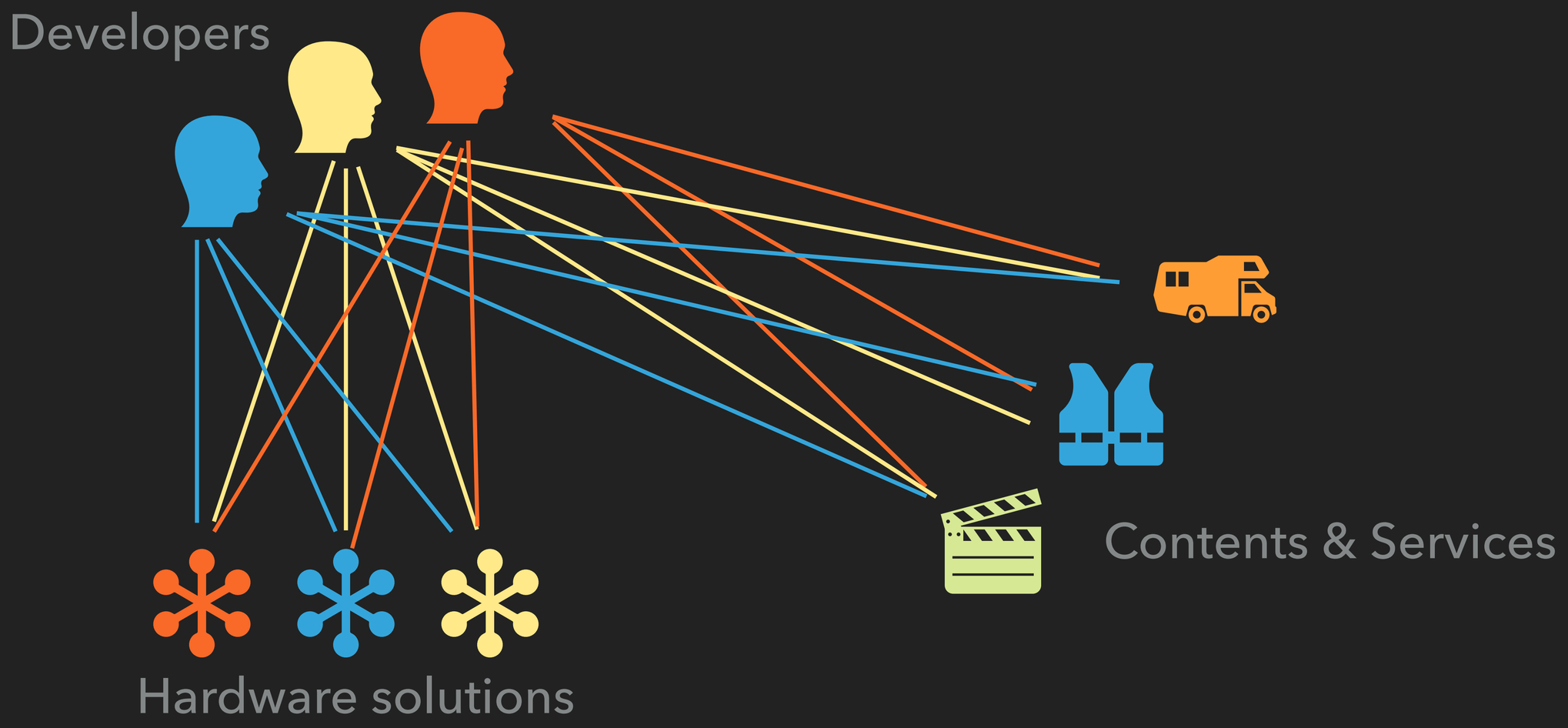
A multi-tasking micro-kernel operating system for ARM, with a simple filesystem and a tiny shell.



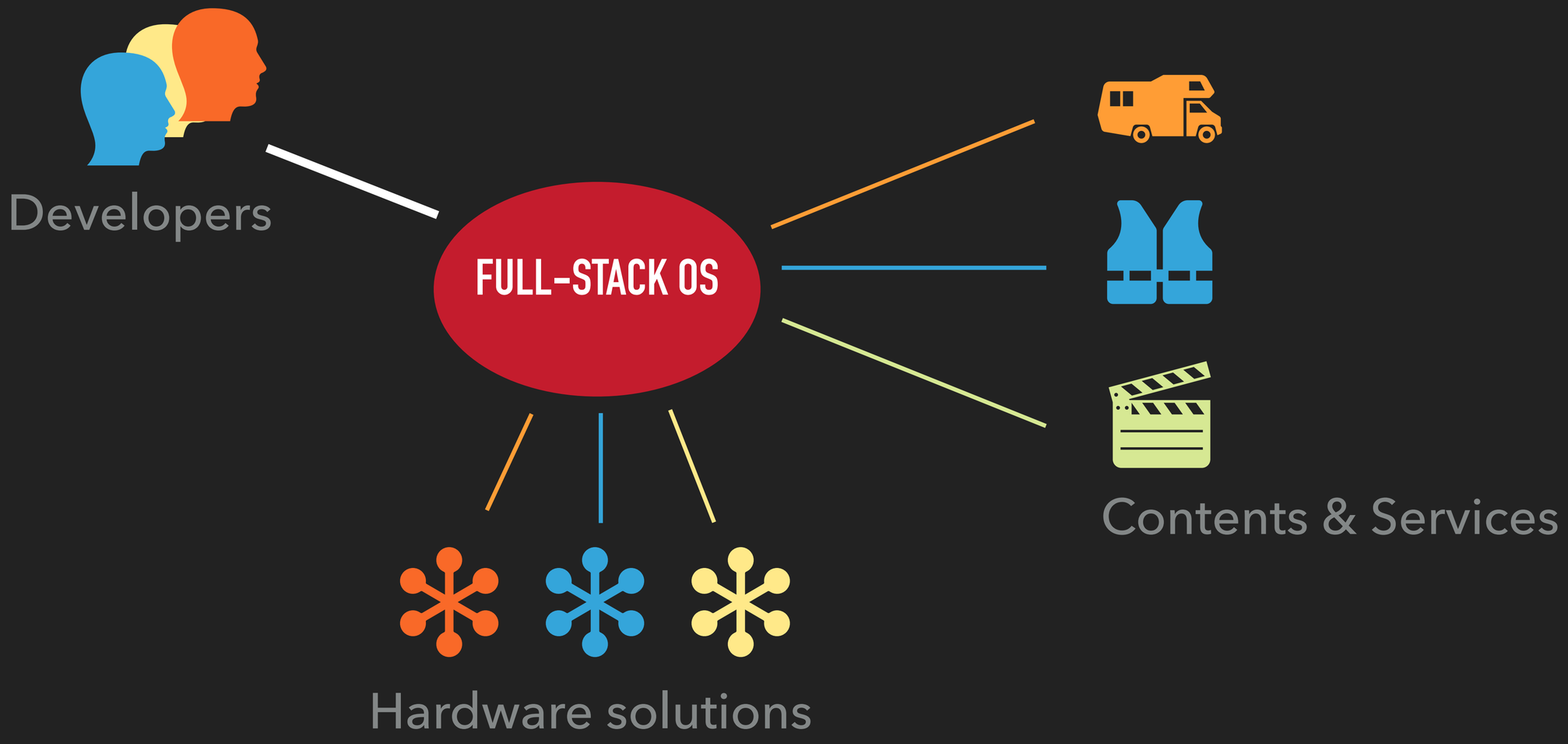
MODERN FULL-STACK OS

- Rich contents and services
- More user-interaction types
- Lots of hardware platforms
- Huge developer community
- Ecosystem

WHY DO WE NEED FULL-STACK OS?



WHY DO WE NEED FULL-STACK OS?



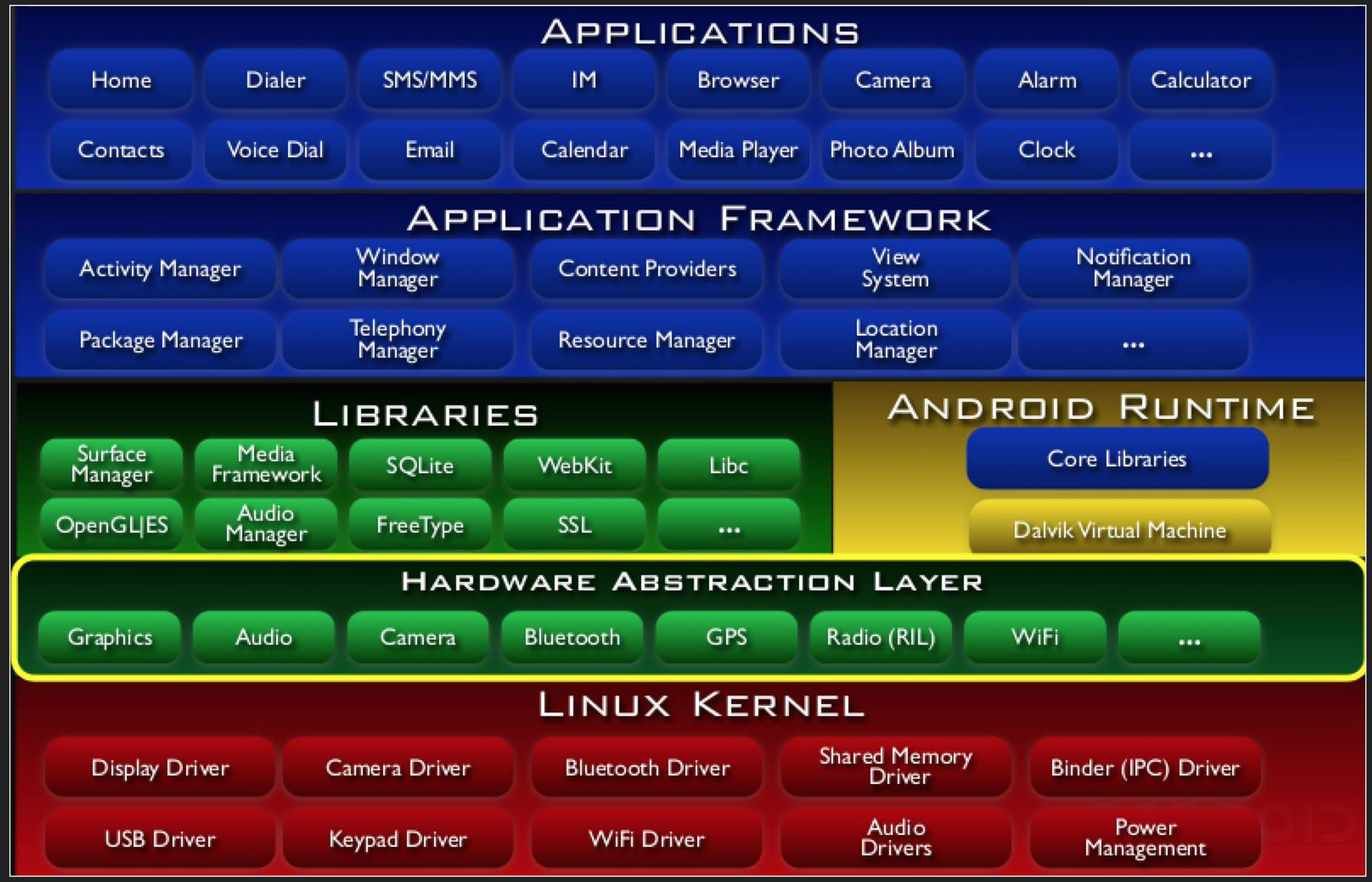
MODERN FULL-STACK OS

IOS

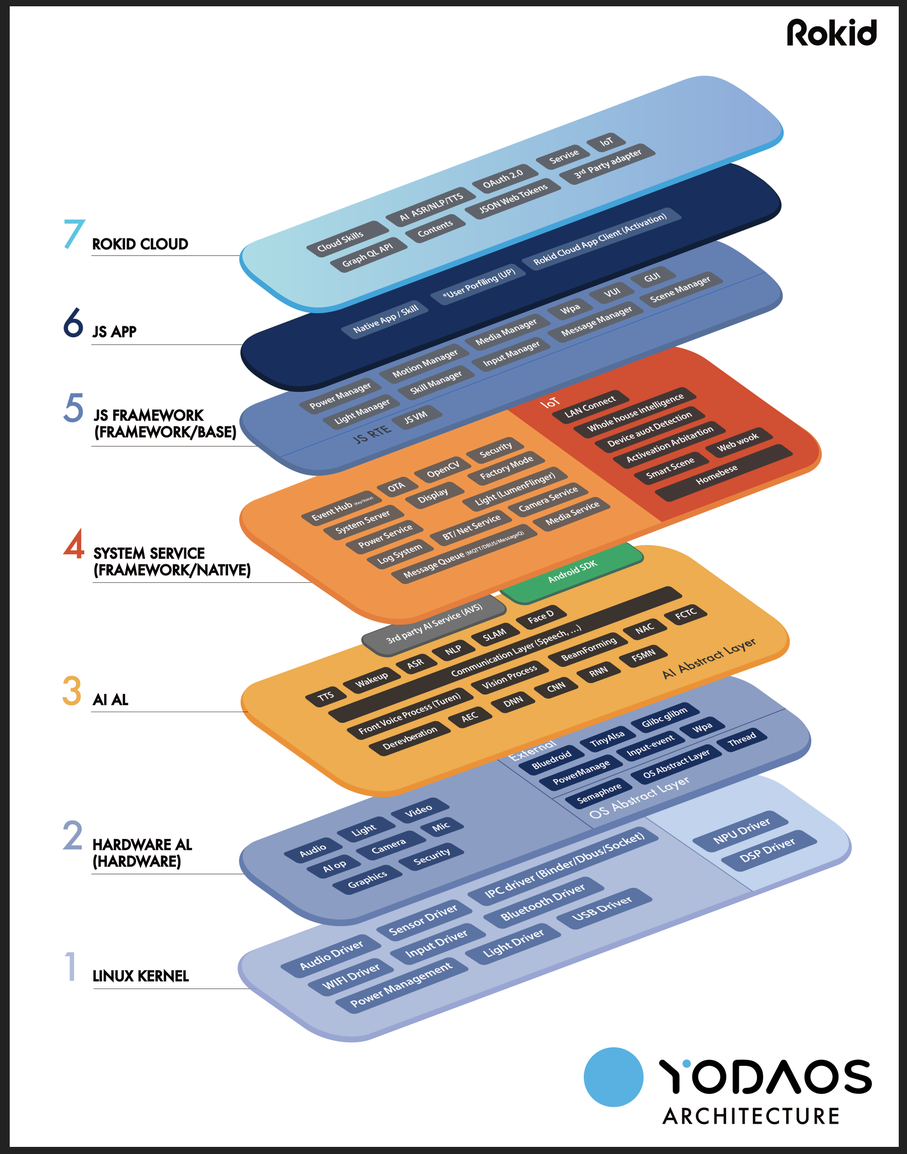
ANDROID

YODA OS

ANDROID FRAMEWORK



OPERATING SYSTEM



Developers

Design house
&Solution providers

Contents & Services

Chip & Hardware solutions

THE END.

MAY THE FORCE BE WITH YOU