Reflection and its use

from science to meditation

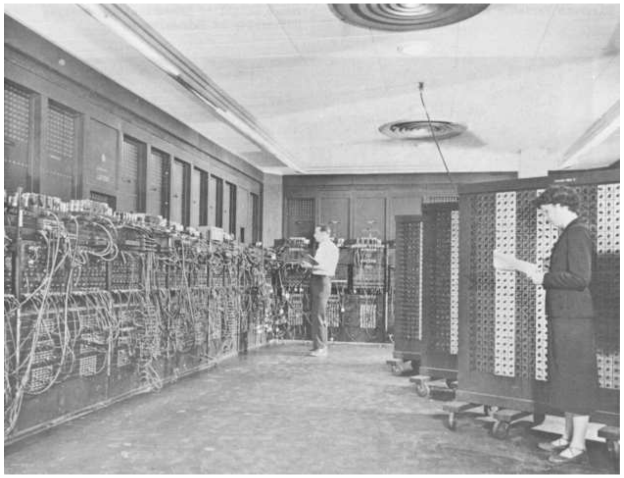
Computing

The first digital electronic computer: ENIAC

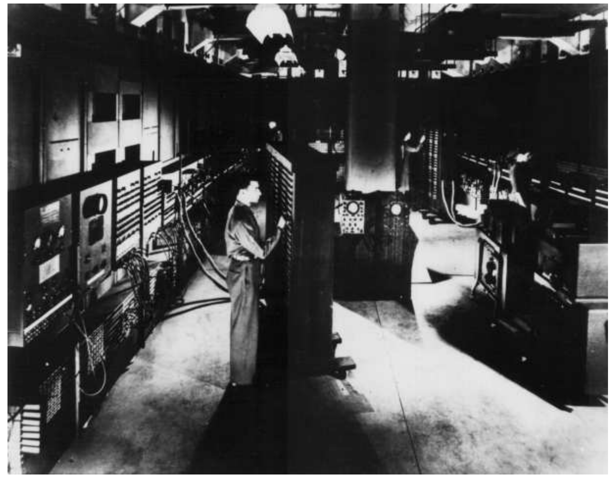
ENIAC (1945)	Spec	Portégé (2003)
6K	Memory	30G
5kHz	Speed	1.2GHz
200KW	Power Consumption	25W
19.000	Tubes	0 (millions of transistors)
30.000Kg	Weight	1.200g
$467.000	Price	$2.500

Computing missile trajectory by hand: 20 hours.
By analogue machine: 15 minutes.
By ENIAC: 30 seconds

Pictures 1



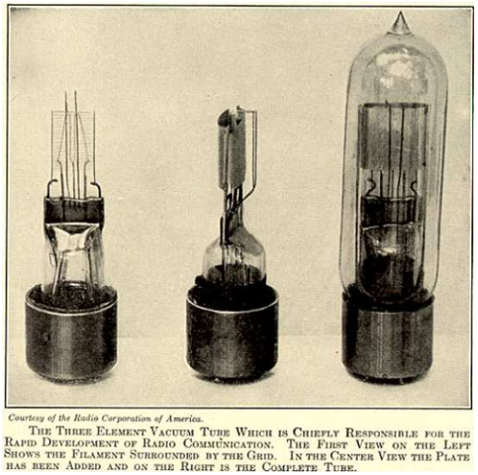
ENIAC
the colossos



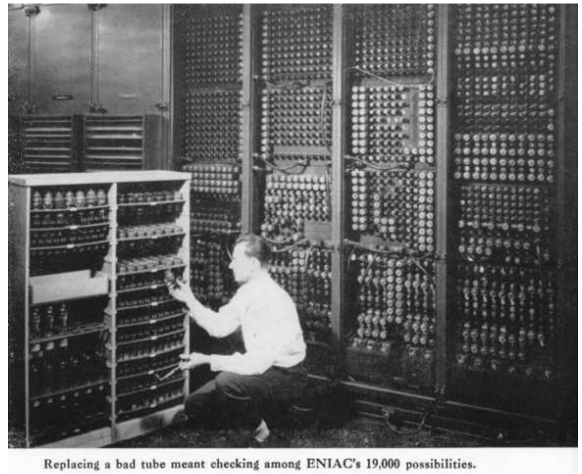
ENIAC
by night

Oldfashioned parts

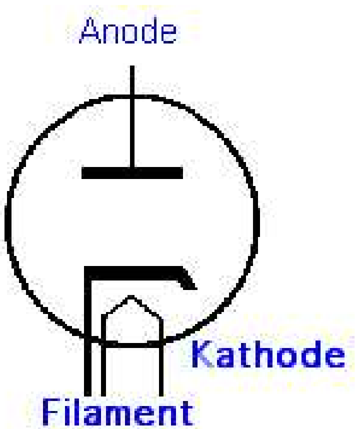
Pictures 2



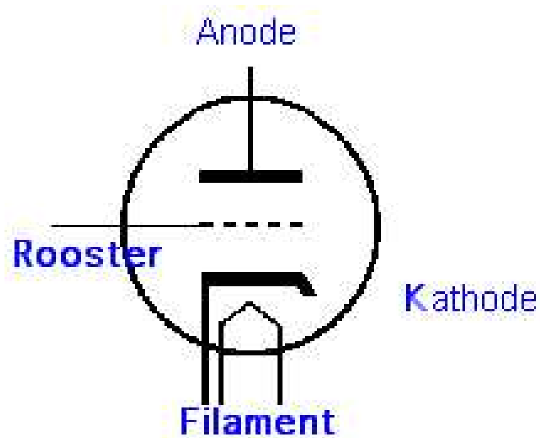
Radiotube



Replacing tubes



Diode



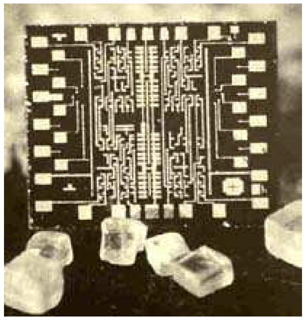
Triode

Tubes vs transistors

Transistor, a US invention

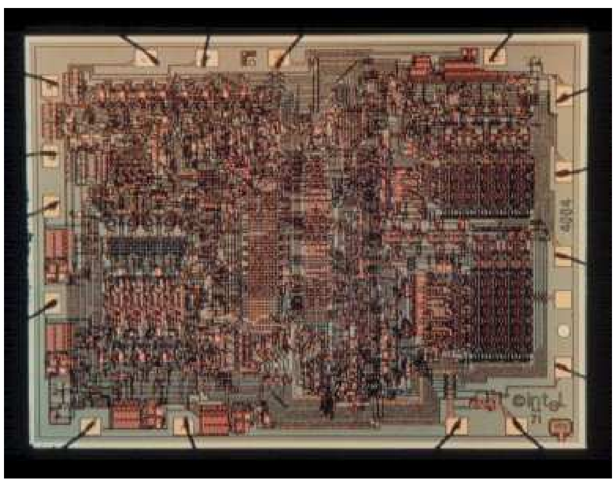
Around 1960 the Japanese transistor industry had distroyed the US radiotube industry.

Miniaturization

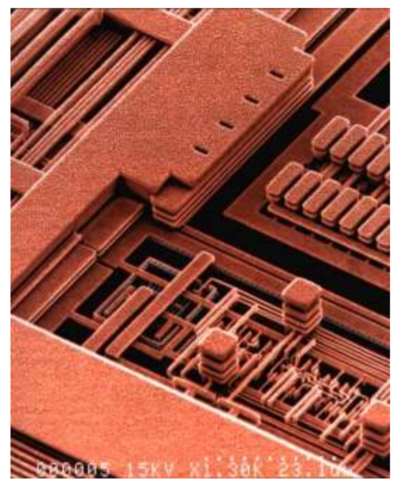


Integrated Cicuit
(IBM 1967)

Apollo 4
(1967)

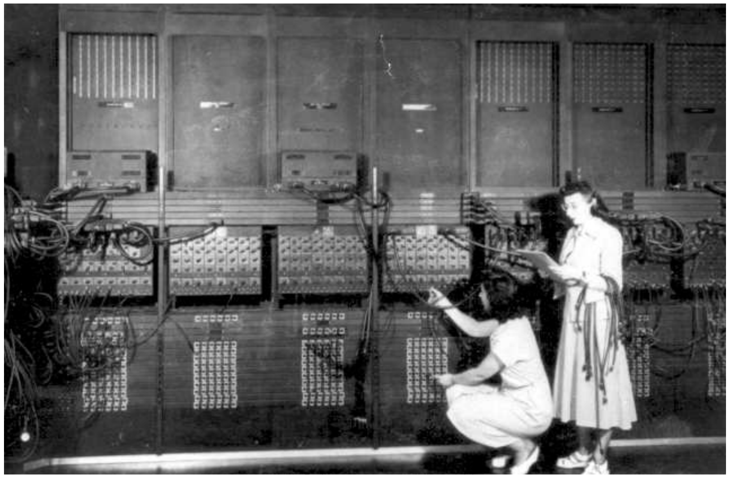


First micro-CPU
$24mm^2$,
2400 transistors
(Intel 1970)



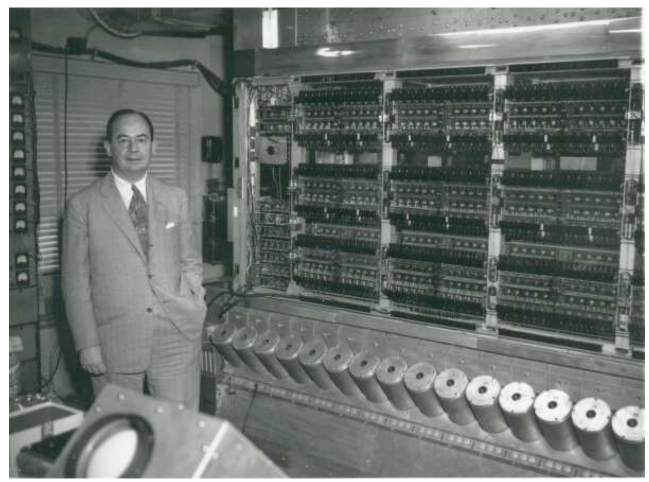
Submicron Technology
(FHDarmstadt, 2001)

Programming



ENIAC: ‘programming’ ladies

Alan Turing (1912-1954)



John von Neumann (1903-1957)

Turing Machines

States	$\{q_0, \ldots, q_n, q_h\}$
Symbols	$1, 0$ (blank)
Tape	Infinite on one side!
(Reading)head	depending on state and input does certain things

There are intructions of the form

$q\ s\ s'\ q'$	In state q reading s replace it by s' and go to state q'
$q\ s\ R\ q'$	In state q reading s move head right and go to state q'
$q\ s\ L\ q'$	In state q reading s move head left and go to state q'

In the beginning the machine is in state q_0 and the head is at the beginning of the tape. (Then it cannot be moved left; if this is attempted, the machine halts.)

Example of a Turing machine. States$= \{q_0, q_1, q_h\}$.

$q_0\ 1\ 0\ q_1$
$q_0\ 0\ 0\ q_h$
$q_1\ 0\ R\ q_0$

This machine moves tape with e.g. 1111 and the rest 0's one place to the right, leaving a 0 at the beginning.

Exercises

1. Make a Turing machine that adds one 1 to an input on the tape consisting of k consecutive 1's.

2. Make a Turing machine that erases all consecutive 1's on the tape and halts at the place of the last erased 1.

3. Make a Turing machine that erases all consecutive 1's on the tape and halts two places to the right of the place of the last erased 1.

4. Make a Turing machine that never halts.

5. Make your own Turing machine. Give specification first.

Turing machines as arithmetic partial functions

An instruction set for a Turing machine M makes it a function of one or several number inputs.

$$M(n), M^2(n_1, n_2),\ M^3(n_1, n_2, n_3), \ldots$$

Moreover, the function is partial: sometimes the machine never halts.

This is achieved as follows. The input (1,3,2) is represented as

$$101110110000\ldots$$

The input (1,0,2) is represented as

$$100110000\ldots$$

The machine starts acting on such a tape in state q_0 with its head at the leftmost 1 on the tape. After the machine halts (if it halts at all) the number of 1's are counted and this is the output. Thereby the 1's do not need to be consecutive.

Exercises

1. Make a Turing Machine M such that $M^2(n, m) = n + m$.
2. Make a Turing machine M such that

$$
\begin{aligned}
M(n) &= \frac{n}{2}, && \text{if } n \text{ is even;} \\
&= \frac{n+1}{2}, && \text{else.}
\end{aligned}
$$

Leibniz's Ideal

Leibniz wanted to construct a machine that could solve all problems.

Turing shows that one cannot solve by a Turing machine the problem whether a given (another) Turing machine stops on a given input.

CHURCH-TURING THESIS. *All computation that can be done by a human, can be done by some Turing Machine.*

COROLLARY. Also humans cannot determine whether a machine with a given input will halt.

Programmable computers

THEOREM [TURING 1936]. There exists a *Universal Turing Machine* U such that for all Turing machines M there is a code p (the program; just a number) such that for all n one has

$$U^2(p, n) = M(n).$$

More generally, one even has that for all M there exists a p such that for all k and all $\vec{n} = n_1, \ldots, n_k$ one has

$$U^{k+1}(p, \vec{n}) = M^k(\vec{n}).$$

An unsolvable problem

Unsolvability of the Halting Problem. It is undecidable by a Turing Machine whether $M^k(\vec{n})$ halts (by looking at M and $\vec{n}$ as 'input').

Proof. Suppose we could check whether $U^2(p,n)$ halts. Define the function f by

$$\begin{array}{rcll} f(n) & = & U^2(n,n)+1, & \text{if } U^2(n,n) \text{ halts;} \\ & = & 0, & \text{if } U^2(n,n) \text{ does not halt.} \end{array}$$

Then f can be defined by some Turing Machine M: we have for all n

$$f(n) = M(n).$$

By the universality of U there exists a program p for M: for all n

$$f(n) = M(n) = U^2(p,n).$$

Suppose $U^2(p,p)$ halts. Then

$$U^2(p,p) = f(p) = U^2(p,p)+1,$$

this is impossible. Hence $U^2(p,p)$ does not halt. But then

$$U^2(p,p) = f(p) = 0,$$

which is again impossible. ■

Moral

It is quite remarkable that Turing,

while solving a philosophical problem of Leibniz,

initiated a multi-trillion $ industry.

Reflection is an essential part of it.

It is used by compilers that translate 'higher languages'

into machine instructions.