0X BARE
C8 METAL

WE ARE TECHNICAL

We aim to achieve competence and excellence in technical knowledge, its applications, and effects.

WE ARE INCLUSIVE

We provide an inclusive community environment which invites and celebrates diversity of experience, thought, and belief.

RESPONSIBLE
ETHICAL
INCLUSIVE
TECHNICAL

WE ARE ETHICAL

We make decisions rooted in the principles of equity and justice, focusing on the greater good of our communities.

WE ARE RESPONSIBLE

We honor commitments and take responsibility for our actions and outcomes.

ALLEGHENY COLLEGE

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

OUR SHARED VALUES

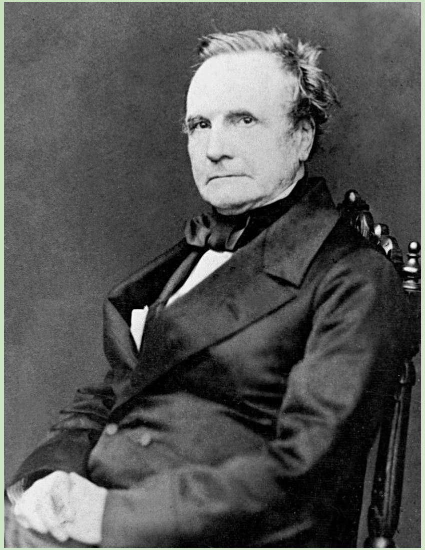
This course is the product of several historical accidents.

0x C8 BARE METAL

First computer builder. →

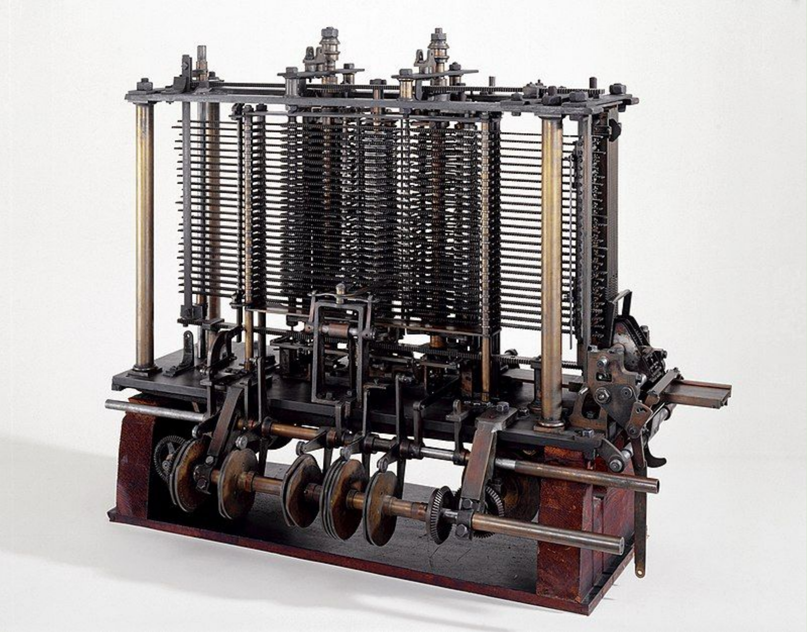
This is Charles Babbage and Ada Lovelace.

First computer programmer. →



0x C8 BARE METAL

This is part of their first computer.



(The full thing was never built.)

0x C8 BARE METAL

This is Alan Turing.

In 1935, Turing outlined a computer that could write to, read from, and erase from a paper tape.

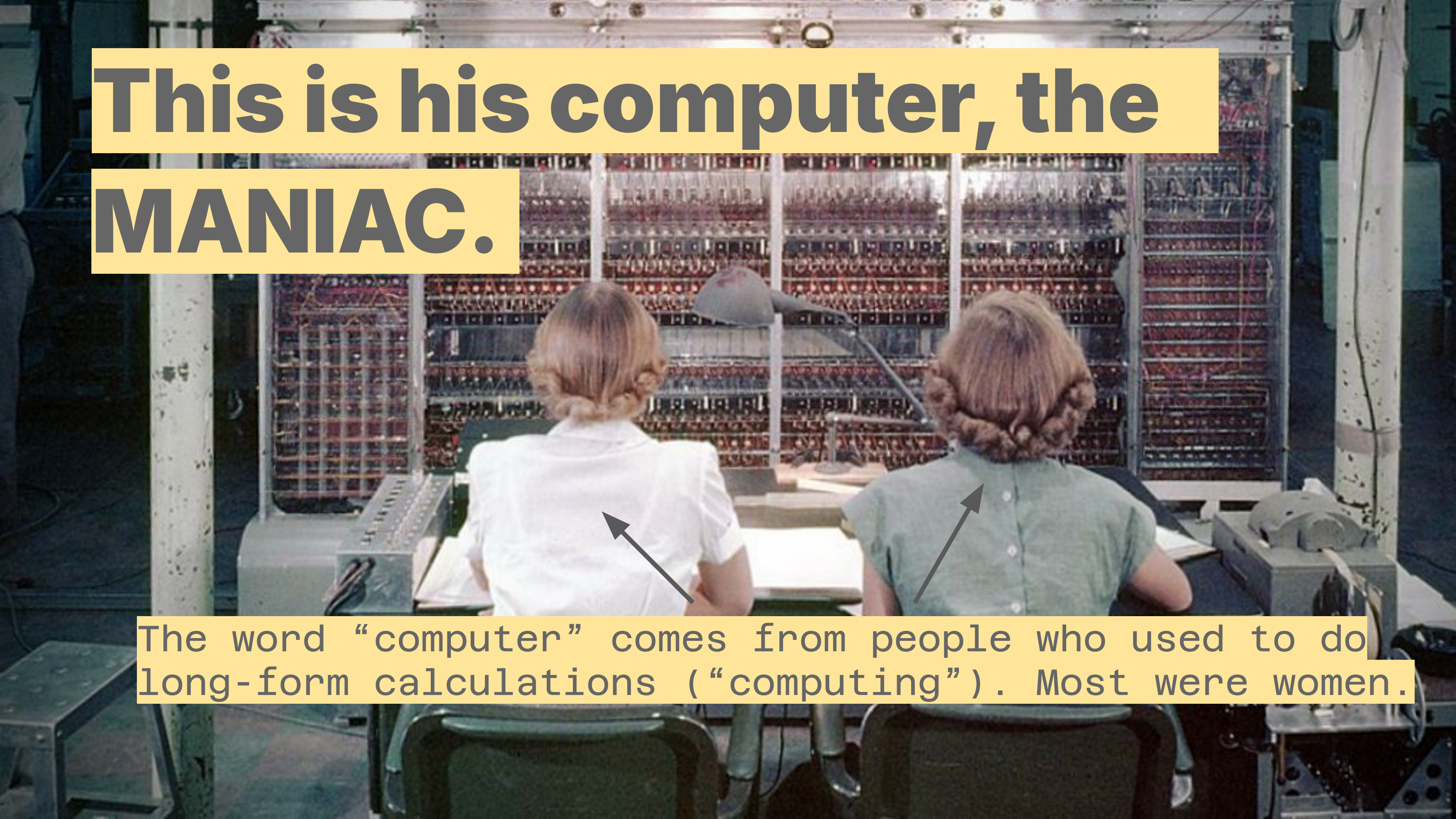
> "
>
> ...given sufficient time, sufficient tape and a precise description, [it] could emulate the behavior of any other computing machine

George Dyson, *Turing's Cathedral.*

This is John von Neumann. Call him "Johnny."

(Every one else does.)

This is his computer, the MANIAC.



The word “computer” comes from people who used to do long-form calculations (“computing”). Most were women.

The thing is, though, that Johnny liked bombs. A lot.

So, the computations that the MANIAC did simulated atomic bomb explosions at Los Alamos.

(Sometimes, folks used the MANIAC to play chess.)

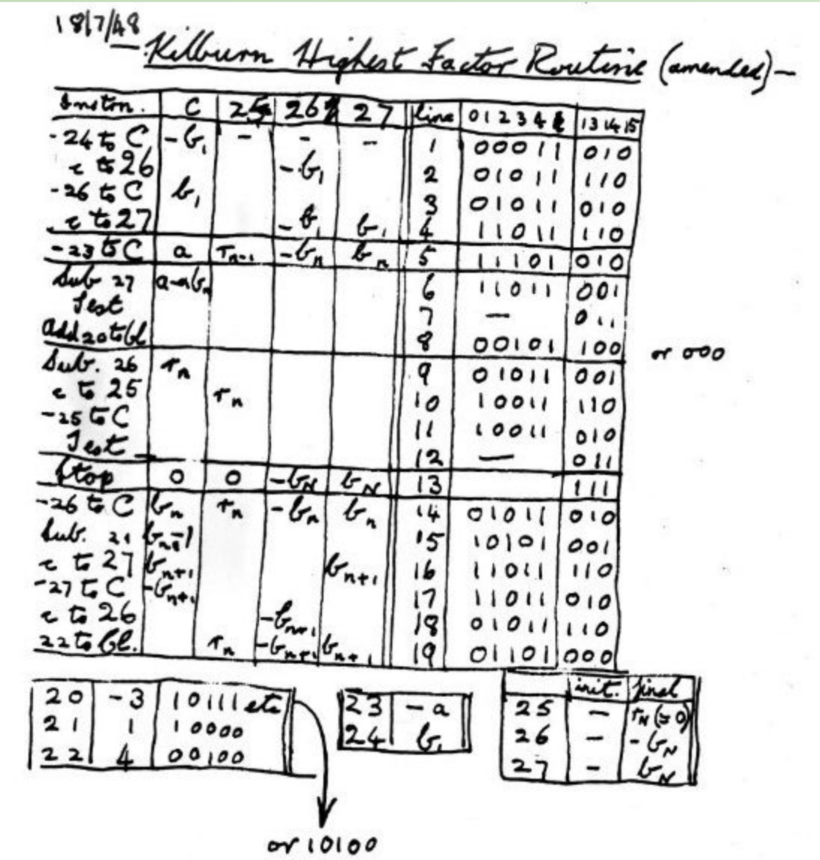
But really, the early computers simulated bombs.

(Johnny was really good at convincing the U.S. Navy to give him money.)

In 1953, there were 53 KB of RAM in the entire world.

The MANIAC had 32.

This is what a a program looked like.



18/7/48 — Kilburn Highest Factor Routine (amended) —

Instn.	C	25	26	27	line	0 1 2 3 4	13 14 15	
-24 to C	$-b_1$	–	–	–	1	00011	010	
c to 26			$-b_1$		2	01011	110	
-26 to C	b_1				3	01011	010	
c to 27			$-b_1$	b_1	4	11011	110	
-23 to C	a	r_{n-1}	$-b_n$	b_n	5	11101	010	
Sub 27	$a-nb_n$				6	11011	001	
Test					7	—	011	
Add 20 to Cl					8	00101	100	or 000
Sub. 26	r_n				9	01011	001	
c to 25		r_n			10	10011	110	
-25 to C					11	10011	010	
Test					12	—	011	
Stop	0	0	$-b_N$	b_N	13		111	
-26 to C	b_n	r_n	$-b_n$	b_n	14	01011	010	
Sub. 21	b_n-1				15	10101	001	
c to 27	b_{n+1}			b_{n+1}	16	11011	110	
-27 to C	$-b_{n+1}$				17	11011	010	
c to 26			$-b_{n+1}$		18	01011	110	
22 to Cl.		r_n	$-b_{n+1}$	b_{n+1}	19	01101	000	

20	-3	10111 etc
21	1	10000
22	4	00100

or 10100

23	$-a$
24	b_1

	init.	final
25	—	r_N (= 0)
26	—	$-b_N$
27	—	b_N

Klára Dán von Neumann

And this is probably one of the main programmers.

This is Gordon Moore.

(He, like, co-founded Intel.)

Moore's Law

"

> The complexity for minimum component costs has increased at a rate of roughly a factor of two per year....there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer.

Gordon Moore, "Cramming more components onto integrated circuits." (1965)

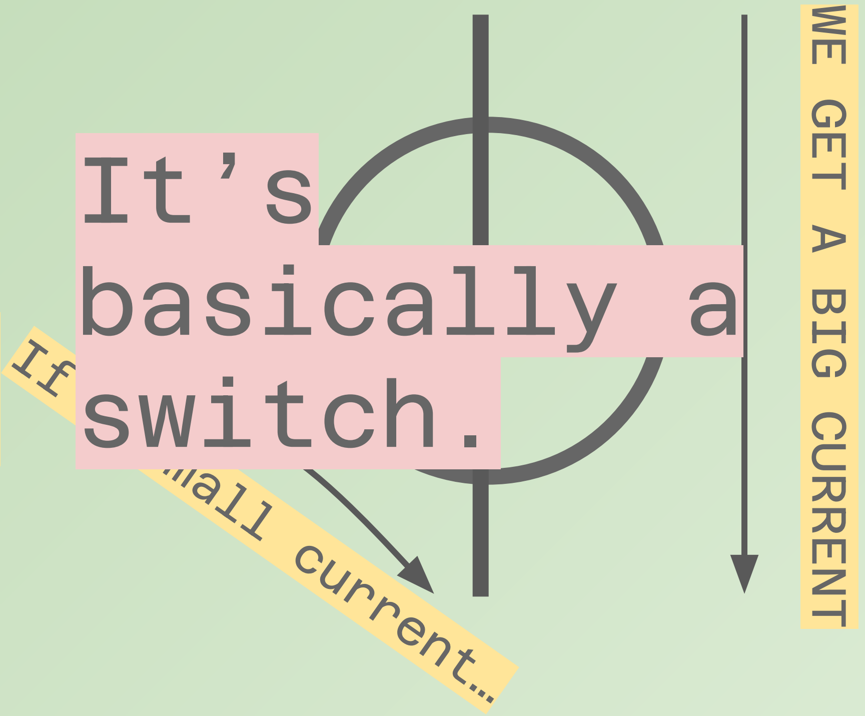
How’s that going?

(2022 edition)

> "
>
> **Intel claim[s] that by 2030, there [will be] circuits with transistor counts of a trillion, roughly ten times the number of transistors currently available on modern CPUs.**

Jorge Jimenez, “Intel says there will be one trillion transistors on chips by 2030”

WTF is a transistor?



It's basically a switch.

If
mall current…

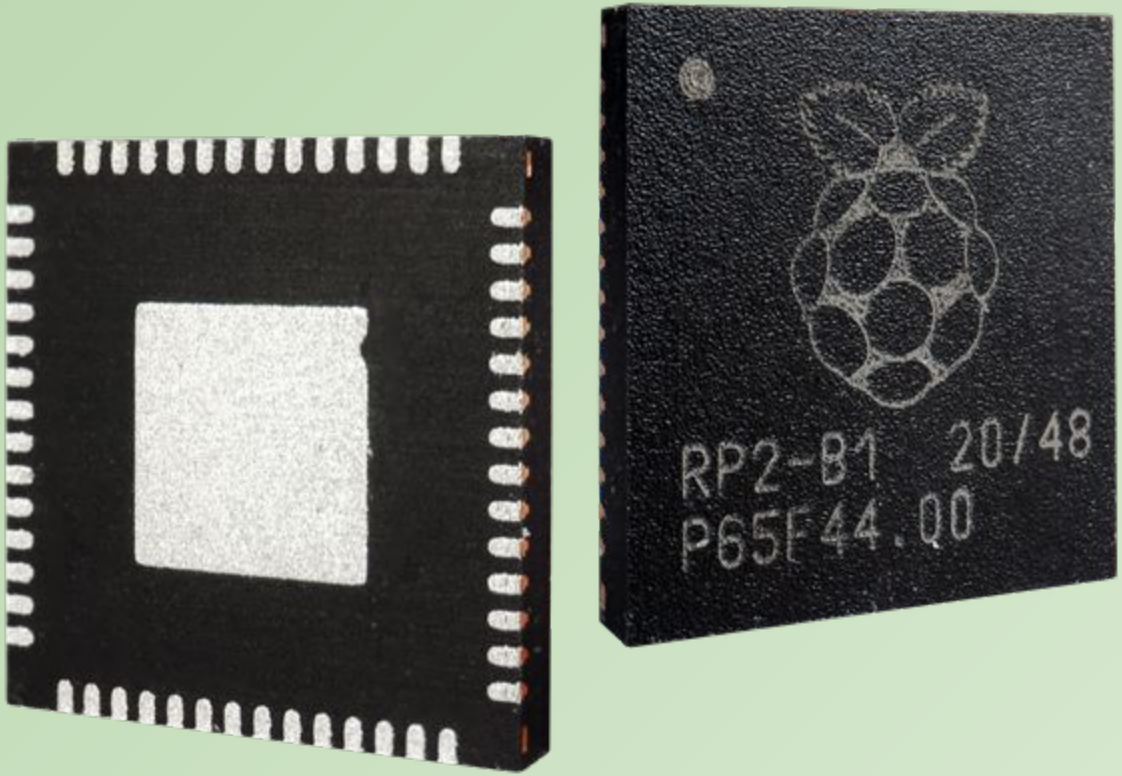
WE GET A BIG CURRENT

And a switch is basically a single

bit

Short for “binary digit,” coined by John W. Tukey in 1945.

And if Turing's right: we assemble enough *bits* (i.e. "tape")…


RP2-B1 20/48
P65F44.00

Our job is to manipulate switches. Always has been.

We do so through an instruction set architecture (ISA).

(And every chip has one.)

Enter a sweet new chip: the LPC

Learning Paper Computer

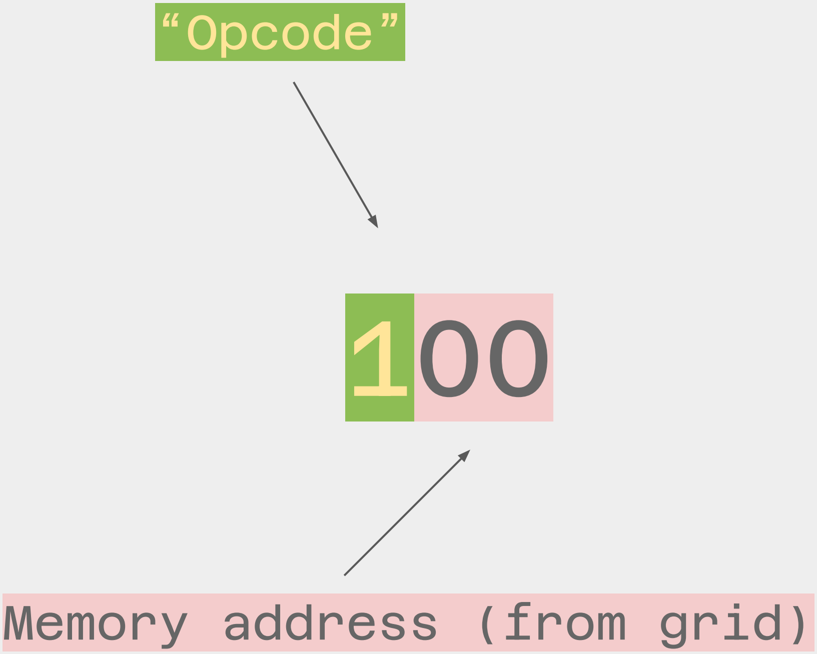
Instruction Set Architecture (ISA)

1XX	ADD	Add
2XX	SUB	Subtract
3XX	STA	Store
4LR	SFT	Shift
5XX	LDA	Load
6XX	BRA	Branch Always
7XX	BRZ	Branch if zero
8XX	BRP	Branch if positive
901	INP	Input
902	OUT	Output
000	HLT	Halt

0x C8 BARE METAL

Learning Paper Computer

Anatomy of a command



“Opcode”

100

Memory address (from grid)

Learning Paper Computer

Anatomy of a command

901 and 902 are special, hard-wired commands.

(So is 4LR, but we'll talk about that later.)

Learning Paper Computer

Functional rules

Memory can’t be written to except by 3XX (STA)

You’re limited to the amount of memory cells (99) provided

Cells can only hold 3-digit numbers.

Programs must be 0-terminated (i.e. 000 (HLT) is the last command)

Learning Paper Computer

Functional rules

Start at cell 00 by setting the PROGRAM COUNTER to 00.

Advance one cell at a time, incrementing the PROGRAM COUNTER.

Every 6XX, 7XX, or 8XX sets cell 99 as well (and rewrites the PROGRAM COUNTER to whatever value it jumps to (+ 1).

Learning Paper Computer

Simple input-output program

1	901
2	902
3	000

Learning Paper Computer

Simple input-output program (3 inputs)

```
1   901
2   902
3   901
4   902
5   901
6   902
7   000
```

Learning Paper Computer

Let's cause an error!

1	901
2	902
3	901
4	902
5	901
6	902
8	000

Learning Paper Computer

Simple program to add one to any input

1	901
2	100
3	902
4	000

Learning Paper Computer

Optimized input-output program (3 inputs)

```
1    901      @ Take an input
2    705      @ Branch if 0
3    902      @ Output
4    601      @ Branch 1
5    000      @ Halt
```