Lecture 8

Computation

History of Data Science, Spring 2022 @ UC San Diego

Suraj Rampure

Announcements

- Homework 8 is released, and is due **Sunday, May 22nd at 11:59PM.**
- Our final class is next week – there will be no class in Week 10 (Memorial Day). → Homework 9
- Come to the DSC town hall tomorrow from 3-5PM in the auditorium of SDSC!

Agenda

- Babbages’ engines and Ada Lovelace’s first program.
- The ENIAC.
- Turing.
- Modern advances.

Origins

Symbols

binary = base 2

8 bits (1 byte)

$\rightarrow 11111111_2 = 255$

- We've looked at how base 10 numbers can be stored in binary.

$2^0+2^1+2^2+\dots+2^7=2^8-1$

- But base 10 numbers are not the only thing our computers need to store and work with.
  - What about negative numbers? Decimals?
  - Strings? ASCII
  - Colors?
  - All of these can be stored in binary as well.

ord("A") → 65

ord("a") → 97

first digit is sign

10101 → −5

00101 → +5

⇒ 10000 → −0

00000 → +0

"Two's complement"

(152, 203, 24)

0-255 0-255 0-255

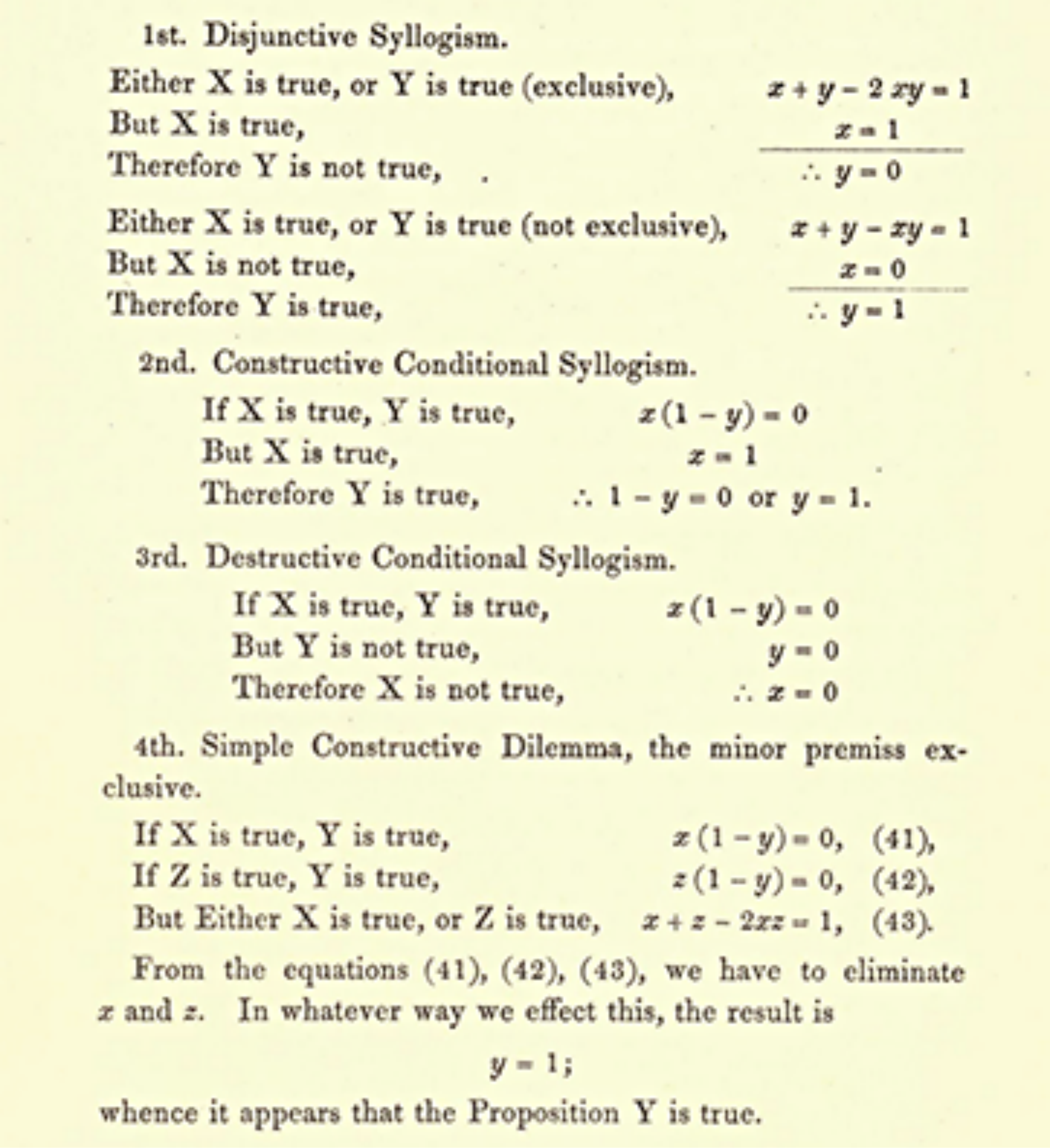
floating point

1001001001100...

≠ 254.32175 (e.g.)

Boolean algebra

- George Boole (1815-1864) was an English mathematician.
- In 1854, he published *An investigation into the Laws of Thought, on Which are founded the Mathematical Theories of **Logic** and Probabilities*, in which he laid the foundations of **Boolean algebra**.
  - In Boolean algebra, there are two values – 1 (True) and 0 (False), and three operators – AND, OR, and NOT.
- Fun fact: at the age of 19, Boole created his own elementary school!



1st. Disjunctive Syllogism.

Either X is true, or Y is true (exclusive), $x + y - 2xy = 1$
But X is true, $x = 1$
Therefore Y is not true, $\therefore y = 0$

Either X is true, or Y is true (not exclusive), $x + y - xy = 1$
But X is not true, $x = 0$
Therefore Y is true, $\therefore y = 1$

2nd. Constructive Conditional Syllogism.

If X is true, Y is true, $x(1 - y) = 0$
But X is true, $x = 1$
Therefore Y is true, $\therefore 1 - y = 0$ or $y = 1$.

3rd. Destructive Conditional Syllogism.

If X is true, Y is true, $x(1 - y) = 0$
But Y is not true, $y = 0$
Therefore X is not true, $\therefore x = 0$

4th. Simple Constructive Dilemma, the minor premiss exclusive.

If X is true, Y is true, $x(1-y) = 0$, (41),
If Z is true, Y is true, $z(1-y) = 0$, (42),
But Either X is true, or Z is true, $x + z - 2xz = 1$, (43).

From the equations (41), (42), (43), we have to eliminate x and z. In whatever way we effect this, the result is

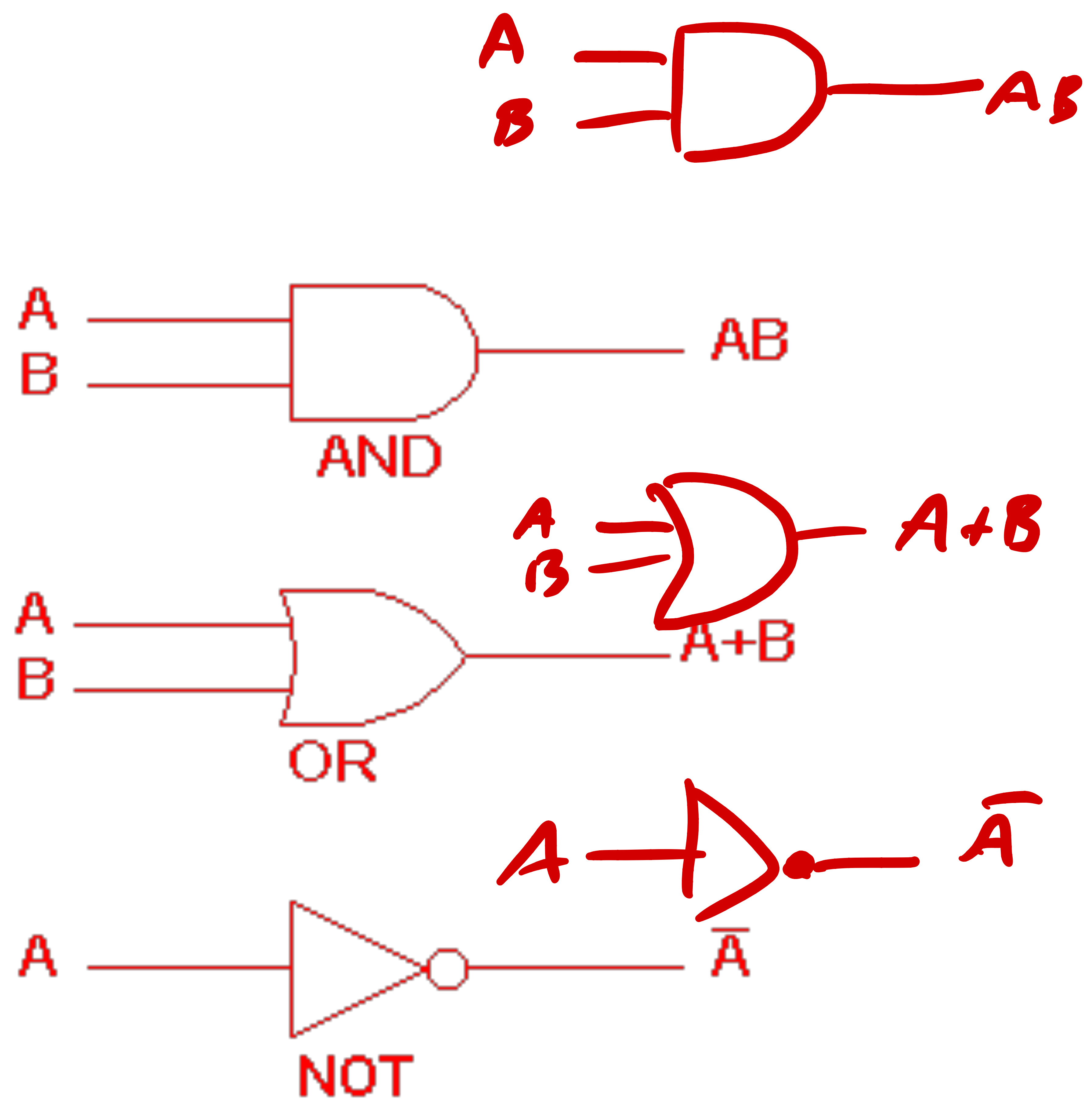
$$y = 1;$$

whence it appears that the Proposition Y is true.

a or (b and not c)

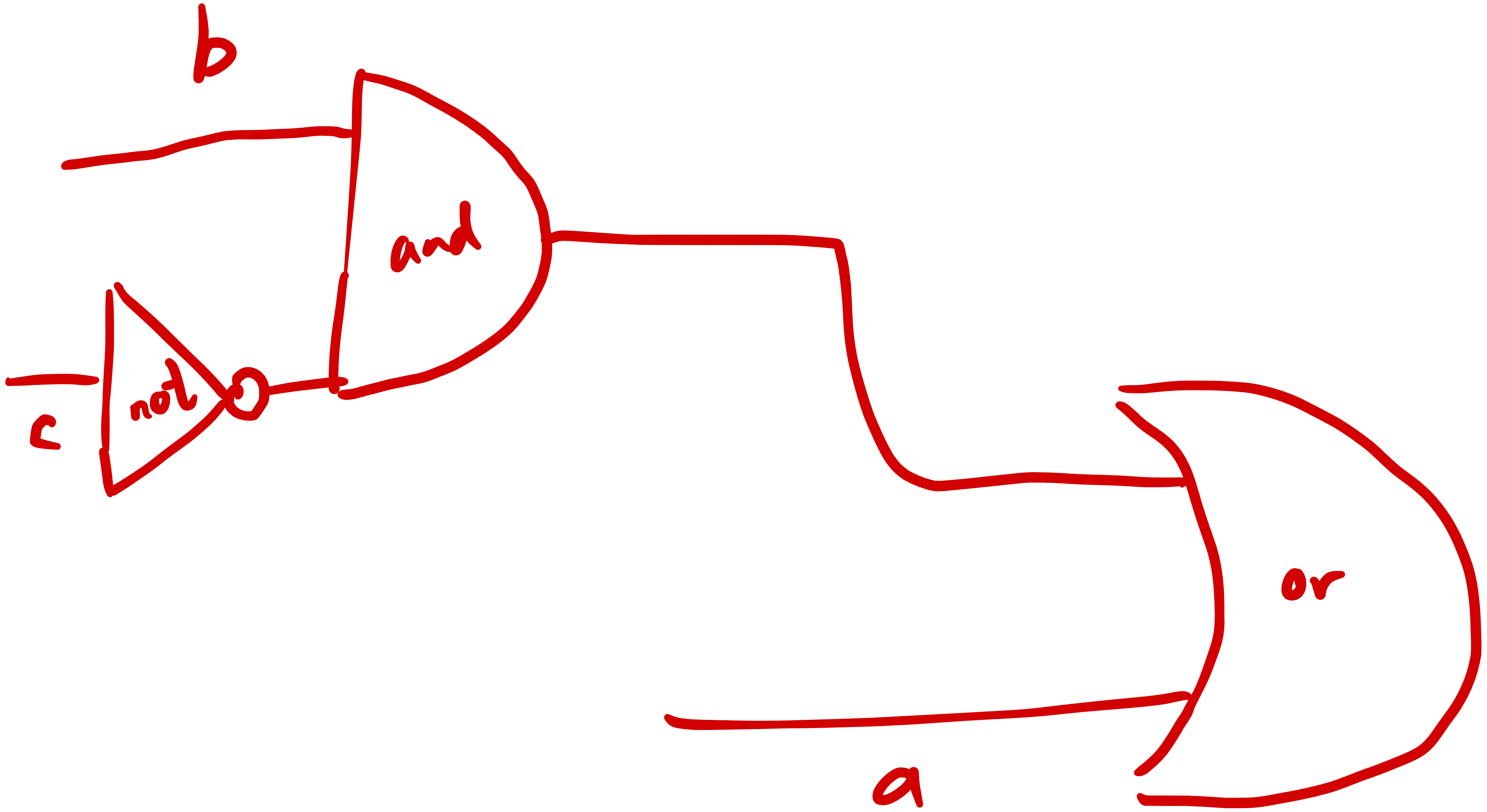
Boolean algebra

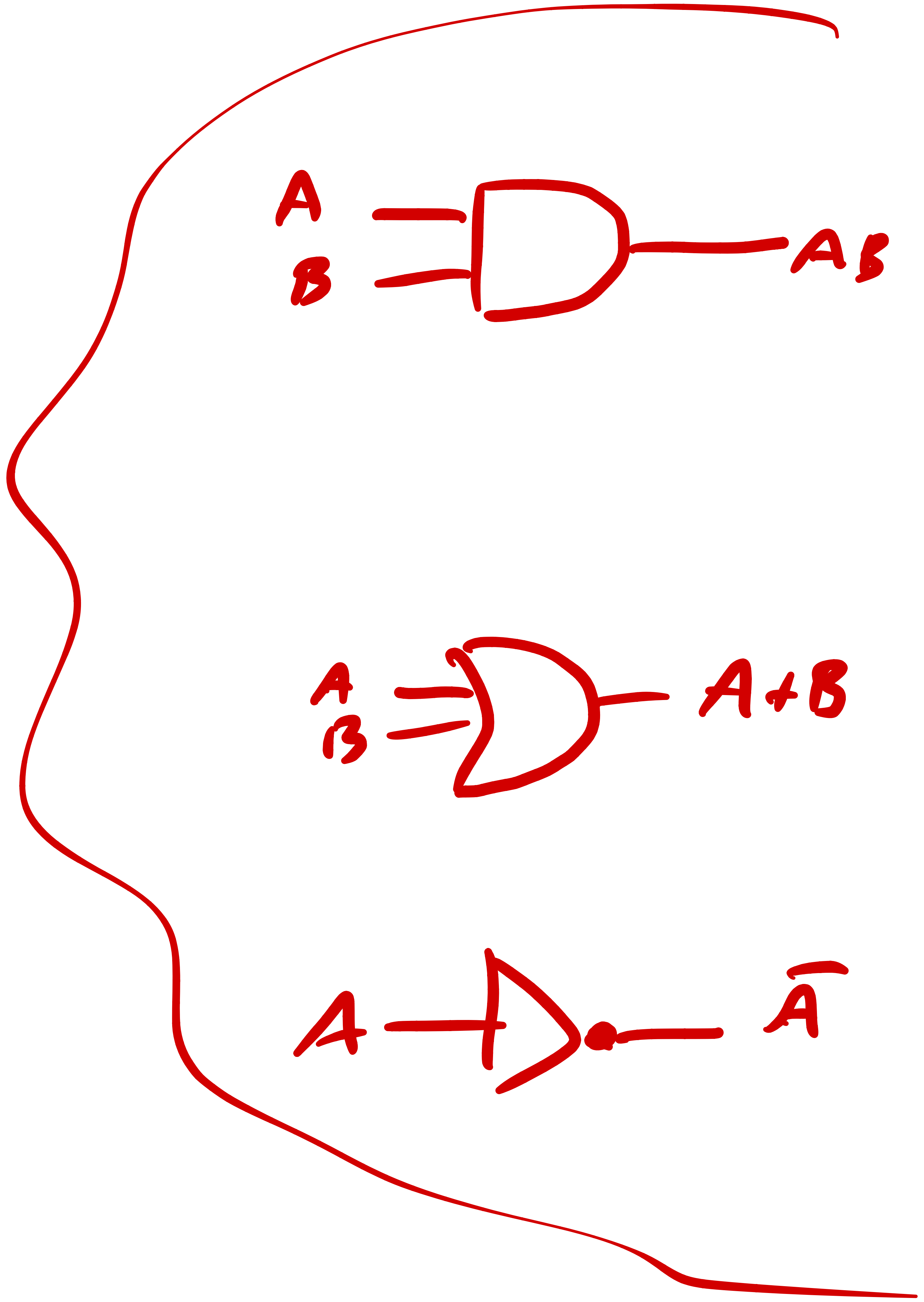
- In Boolean algebra, there are two values – 1 (True) and 0 (False), and three operators – AND, OR, and NOT.
- All other operations can be constructed using a combination of these three operators.
- Circuits use Boolean algebra to control the flow of current.



a or (b and not c)

$$a \lor (b \land \neg c)$$



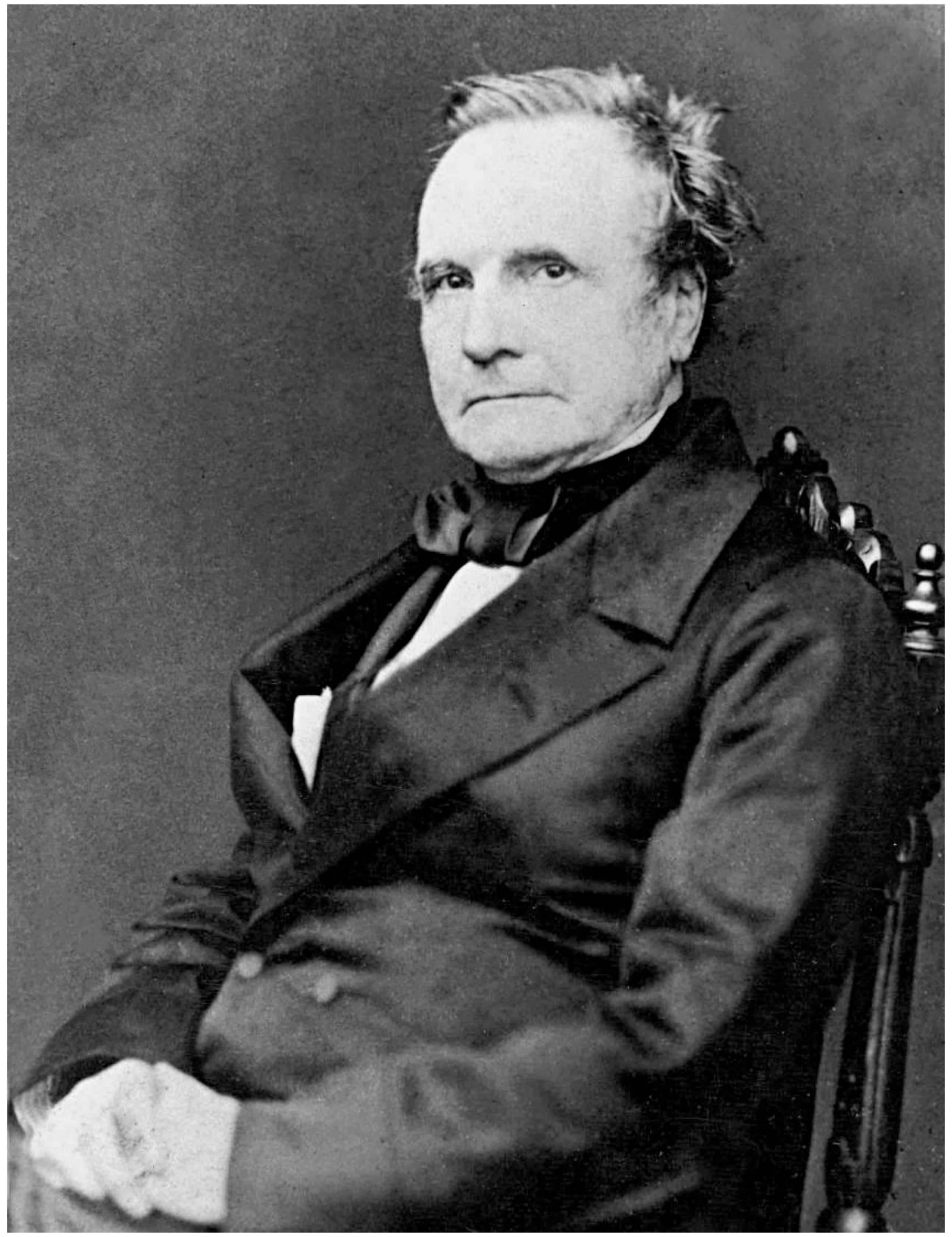


similar to union, intersection, and complement

Babbage and Lovelace

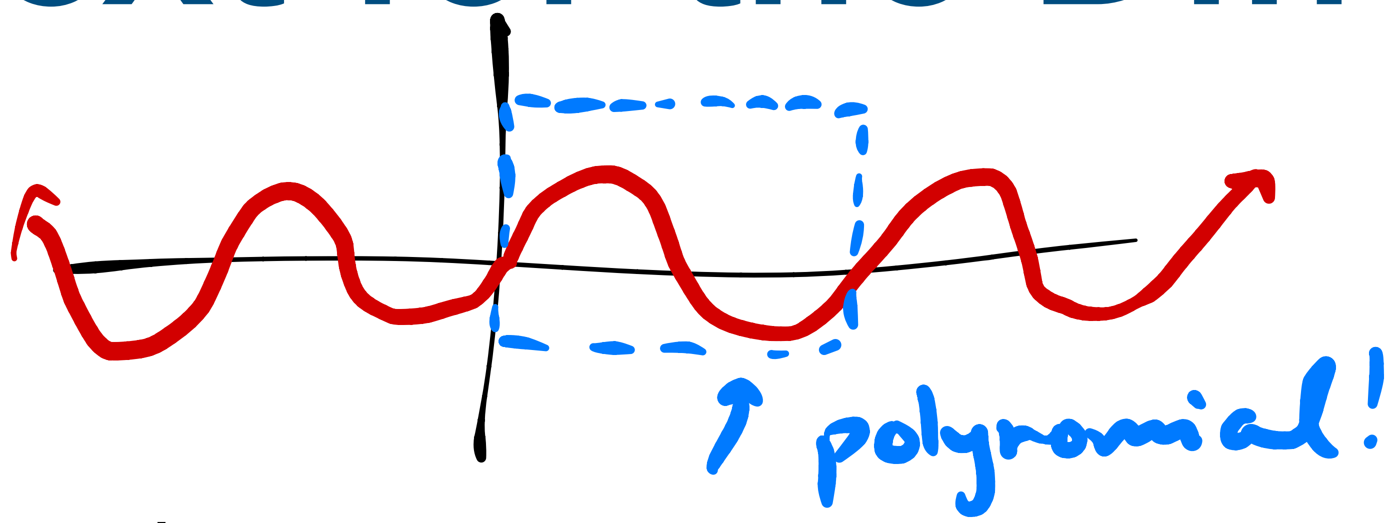
Babbages’ engines

- Charles Babbage (1791-1871) was an English mathematician and engineer, and is credited with designing the first mechanical computer.
- Babbage designed two types of “engines”:
    - A **Difference Engine**, which used the method of finite differences to compute tables of polynomials, and
    - An **Analytical Engine**, which could carry out more sophisticated operations involving arithmetic, looping, and conditional statements.
- Neither engine was fully built during Babbage’s lifetime, though he built prototypes.



Charles Babbage

Context for the Difference Engine



$f(x) = x^4 - 3x^3 + 13x^2 - 4x - 7$

x	$f(x)$
0	
1	
2	
3	
⋮	⋮

- Until the late 1900s, it was common to look at printed tables for logarithms and trig functions.
- The accuracy of these tables was crucial for geodesists, astronomers, and navigators, however these tables often contained errors.
- Babbage's goal was to create accurate tables that were generated using a machine.
- **Key Idea:** polynomials can be used to approximate any function. Taylor series

The method of finite differences

$f(x) = x^2 - 5x + 10$
$f'(x) = 2x - 5$
$f''(x) = 2$

Key Idea: in an nth degree polynomial, the nth differences are constant.

constant

Example: $f(x) = x^2 - 5x + 10$

x	$f(x)$	Δ_1	Δ_2
1	6		
2	4	−2	
3	4	0	2
4	6	2	2
5	10	4	2
6	16	6	2

Example: $f(x) = x^2 - 5x + 10$

required $f(1), f(2), f(3)$

x	$f(x)$	Δ_1	Δ_2
1	6	-2	2
2	4	0	2
3	4	2	2
4	6	4	2
5	10	6	2
6	16	8	2

key: only used addition!!!

Example: $f(x) = x^3 - 3x^2 + x - 1$

x	$f(x)$	Δ_1	Δ_2	Δ_3
0	-1	-1	0	6
1	-2	-1	6	6
2	-3	5	12	6
3	2	17	18	6
4	19	35	24	6
5	54	59	30	6
6	113	89	36	6
7	202	125	42	6

Example: $f(x) = x^3 - 3x^2 + x - 1$

x	$f(x)$	Δ_1	Δ_2	Δ_3
0	-1			
		-1		
1	-2			
		-1	0	
2	-3			
		5	6	6
3	2			

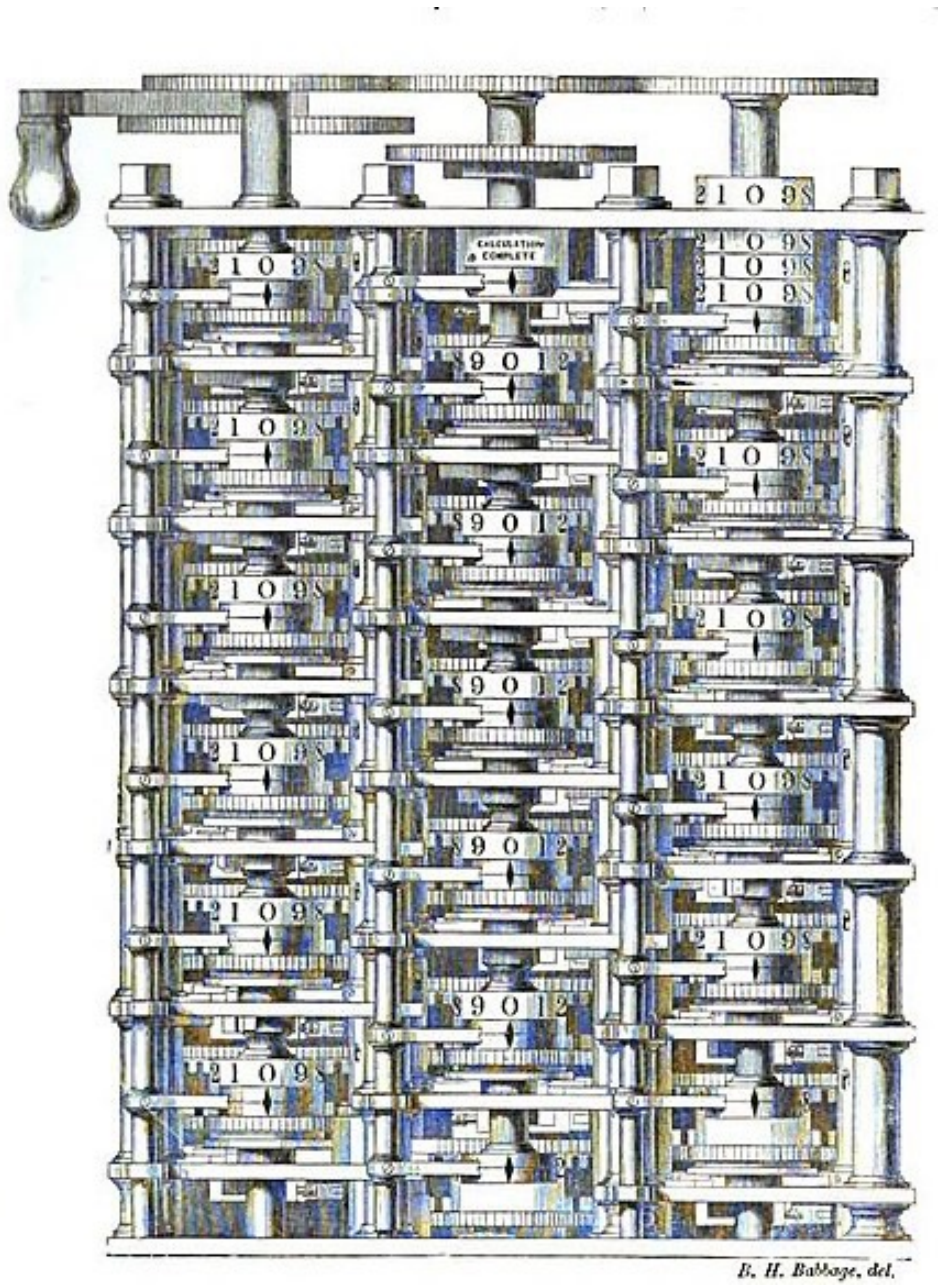
$f'(x) = 3x^2 + \ldots$

$f''(x) = 6x + \ldots$

$f'''(x) = 6$

Difference Engine

- Babbage's Difference Engines used the method of finite differences to evaluate polynomials, using only addition!
- He designed at least two difference engines – Difference Engine No. 1 and Difference Engine No. 2. The latter was designed to evaluate polynomials of degree 7 and used a fraction of the parts of No. 1, which could only evaluate polynomials of degree 6.
- Due to funding constraints, he was never able to fully construct these engines.



A sketch of part of a difference engine

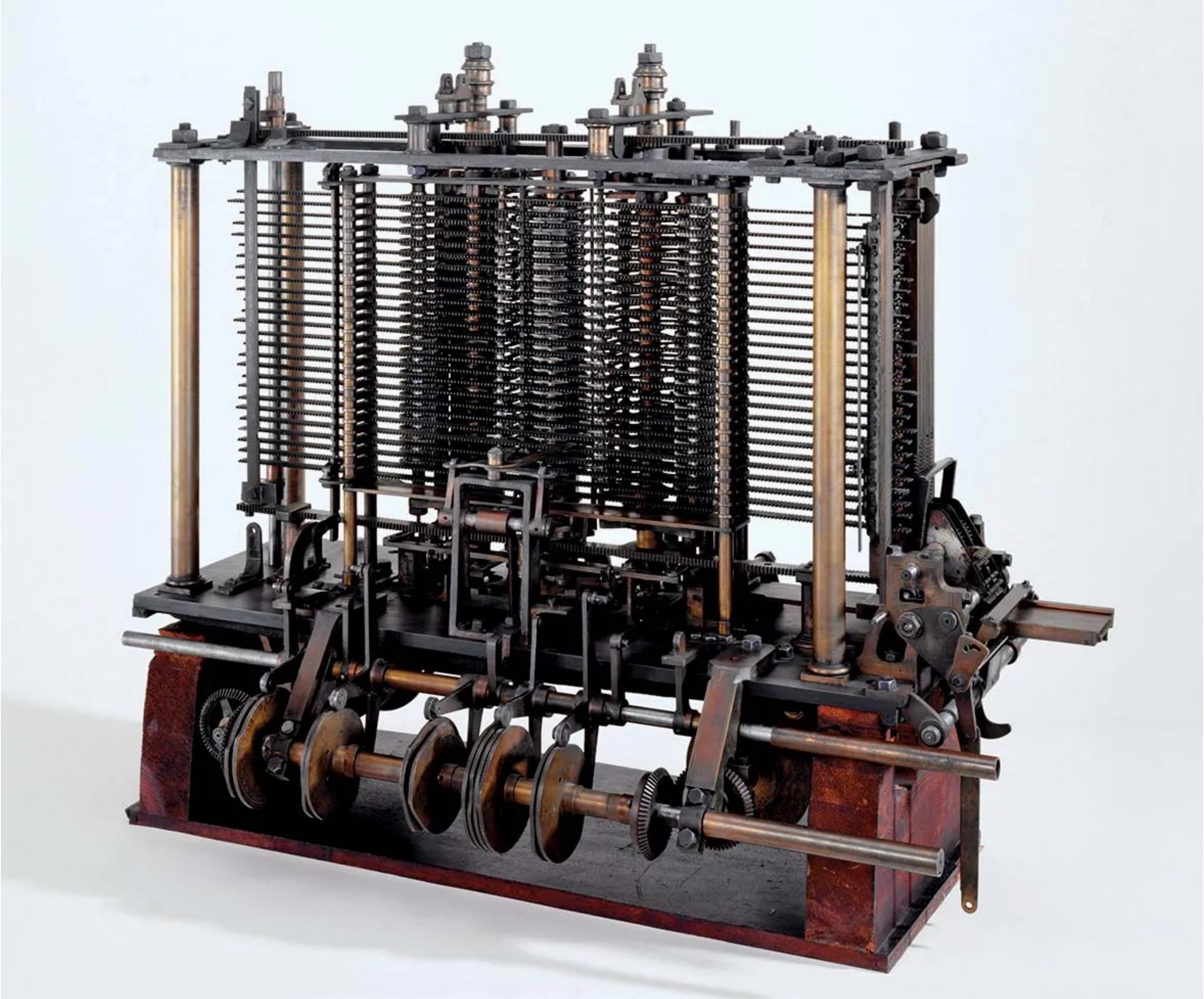
Analytical Engine

- After he devised Difference Engine No. 1, but before he devised Difference Engine No. 2, Babbage designed the more general **Analytical Engine**.
- The Analytical Engine could be programmed using **punch cards**.
  - Babbage borrowed the punch card idea from Joseph Marie Jacquard, who created a loom for weaving textiles that read instructions from a punched card.

Punched cards used in a Jacquard loom

Analytical Engine

- The Analytical Engine is often called the "first computer" due to its design – it consisted of:
    - a **mill** for computation, CPU
    - a **store** to hold values, RAM
    - a **reader** to accept instructions, and keyboard/mouse
    - a **printer** to display results display



A partially complete Analytical Engine, consisting of a mill and a printer. (source)

Ada Lovelace

English poet

- Ada Lovelace (1815-1852) was the daughter of Lord Byron and a friend of Charles Babbage.
- She worked closely with Babbage on the design of the Analytical Engine.
- She proposed a **series of steps** that the analytical engine could use to produce the sequence of Bernoulli numbers.
  - This is thought to be the first **computer program**, and hence Lovelace is the first **computer programmer**.
  - However, since the Analytical Engine was never built in their lifetimes, the program was never tested.

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

typo somewhere

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data. $^1V_1$ 0 0 0 1 $\boxed{1}$	$^1V_2$ 0 0 0 2 $\boxed{2}$	$^1V_3$ 0 0 0 4 $\boxed{n}$	Working Variables. $^0V_4$ 0 0 0 0	$^0V_5$ 0 0 0 0	$^0V_6$ 0 0 0 0	$^0V_7$ 0 0 0 0	$^0V_8$ 0 0 0 0	$^0V_9$ 0 0 0 0	$^0V_{10}$ 0 0 0 0	$^0V_{11}$ 0 0 0 0	$^0V_{12}$ 0 0 0	$^0V_{13}$ 0 0 0	Result Variables. $^1V_{21}$ B_1 in a decimal fraction. $\boxed{B_1}$	$^1V_{22}$ B_3 in a decimal fraction. $\boxed{B_3}$	$^1V_{23}$ B_5 in a decimal fraction. $\boxed{B_5}$	$^0V_{24}$... 0 0 0 0 $\boxed{B_7}$
1	×	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$\left\{\begin{matrix} ^1V_2 = {}^1V_2 \\ ^1V_3 = {}^1V_3 \end{matrix}\right\}$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	−	$^1V_4 - {}^1V_1$	$^2V_4$	$\left\{\begin{matrix} ^1V_4 = {}^2V_4 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n-1$	1	...	...	$2n-1$													
3	+	$^1V_5 + {}^1V_1$	$^2V_5$	$\left\{\begin{matrix} ^1V_5 = {}^2V_5 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n+1$	1	...	...	...	$2n+1$												
4	÷	$^2V_5 \div {}^2V_4$	$^1V_{11}$	$\left\{\begin{matrix} ^2V_5 = {}^0V_5 \\ ^2V_4 = {}^0V_4 \end{matrix}\right\}$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$						
5	÷	$^1V_{11} \div {}^1V_2$	$^2V_{11}$	$\left\{\begin{matrix} ^1V_{11} = {}^2V_{11} \\ ^1V_2 = {}^1V_2 \end{matrix}\right\}$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\frac{1}{2} \cdot \frac{2n-1}{2n+1}$						
6	−	$^0V_{13} - {}^2V_{11}$	$^1V_{13}$	$\left\{\begin{matrix} ^2V_{11} = {}^0V_{11} \\ ^0V_{13} = {}^1V_{13} \end{matrix}\right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	....	$-\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - {}^1V_1$	$^1V_{10}$	$\left\{\begin{matrix} ^1V_3 = {}^1V_3 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n - 1\ (= 3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	+	$^1V_2 + {}^0V_7$	$^1V_7$	$\left\{\begin{matrix} ^1V_2 = {}^1V_2 \\ ^0V_7 = {}^1V_7 \end{matrix}\right\}$	$= 2 + 0 = 2$	...	2	...	...	...	...	2										
9	÷	$^1V_6 \div {}^1V_7$	$^3V_{11}$	$\left\{\begin{matrix} ^1V_6 = {}^1V_6 \\ ^0V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2} = A_1$						
10	×	$^1V_{21} \times {}^3V_{11}$	$^1V_{12}$	$\left\{\begin{matrix} ^1V_{21} = {}^1V_{21} \\ ^3V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	....	B_1			
11	+	$^1V_{12} + {}^1V_{13}$	$^2V_{13}$	$\left\{\begin{matrix} ^1V_{12} = {}^0V_{12} \\ ^1V_{13} = {}^2V_{13} \end{matrix}\right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$ ...	...	...	...	...	...	...	...	...	...	...	....	0	$\left\{-\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}\right\}$				
12	−	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$\left\{\begin{matrix} ^1V_{10} = {}^2V_{10} \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n - 2\ (= 2)$	1	...	...	...	...	...	...	...	...	$n-2$							
13	−	$^1V_6 - {}^1V_1$	$^2V_6$	$\left\{\begin{matrix} ^1V_6 = {}^2V_6 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n - 1$	1	...	...	...	...	$2n-1$											
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$\left\{\begin{matrix} ^1V_1 = {}^1V_1 \\ ^1V_7 = {}^2V_7 \end{matrix}\right\}$	$= 2 + 1 = 3$	1	...	...	...	...	...	3										
15	÷	$^2V_6 \div {}^2V_7$	$^1V_8$	$\left\{\begin{matrix} ^2V_6 = {}^2V_6 \\ ^2V_7 = {}^2V_7 \end{matrix}\right\}$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$									
16	×	$^1V_8 \times {}^3V_{11}$	$^4V_{11}$	$\left\{\begin{matrix} ^1V_8 = {}^0V_8 \\ ^3V_{11} = {}^4V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2} \cdot \frac{2n-1}{3}$						
17	−	$^2V_6 - {}^1V_1$	$^3V_6$	$\left\{\begin{matrix} ^2V_6 = {}^3V_6 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n - 2$	1	...	...	...	...	$2n-2$											
18	+	$^1V_1 + {}^2V_7$	$^3V_7$	$\left\{\begin{matrix} ^2V_7 = {}^3V_7 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 3 + 1 = 4$	1	...	...	...	...	...	4										
19	÷	$^3V_6 \div {}^3V_7$	$^1V_9$	$\left\{\begin{matrix} ^3V_6 = {}^3V_6 \\ ^3V_7 = {}^3V_7 \end{matrix}\right\}$	$= \frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{\frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = A_3\right\}$						
20	×	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$\left\{\begin{matrix} ^1V_9 = {}^0V_9 \\ ^4V_{11} = {}^5V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0								
21	×	$^1V_{22} \times {}^5V_{11}$	$^0V_{12}$	$\left\{\begin{matrix} ^1V_{22} = {}^1V_{22} \\ ^0V_{12} = {}^2V_{12} \end{matrix}\right\}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...	0	$B_3 A_3$	....	......	B_3		
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$\left\{\begin{matrix} ^2V_{12} = {}^0V_{12} \\ ^2V_{13} = {}^3V_{13} \end{matrix}\right\}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	...	...	...	...	...	...	...	...	....	0	$\left\{A_3 + B_1 A_1 + B_3 A_3\right\}$				
23	−	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$\left\{\begin{matrix} ^2V_{10} = {}^3V_{10} \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n - 3\ (= 1)$	1	...	...	...	...	...	...	...	...	$n-3$							
					Here follows a repetition of Operations thirteen to twenty-three.																	
24	+	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$\left\{\begin{matrix} ^4V_{13} = {}^0V_{13} \\ ^0V_{24} = {}^1V_{24} \end{matrix}\right\}$	$= B_7$	...	...	...	...	...	...	...	...	...	...	....	....	....	......	......	......	B_7
25	+	$^1V_1 + {}^1V_3$	$^1V_3$	$\left\{\begin{matrix} ^1V_1 = {}^1V_1 \\ ^1V_3 = {}^1V_3 \\ ^5V_6 = {}^0V_6 \\ ^5V_7 = {}^0V_7 \end{matrix}\right\}$	$= n + 1 = 4 + 1 = 5$ .... by a Variable-card. by a Variable card.	1	...	$n+1$	...	...	0	0										

Lovelace's note on how to use the Analytical Engine to compute the sequence of **Bernoulli numbers**.

Bernoulli numbers

- As you explore discover in this week's homework, the Bernoulli numbers are a sequence of rational numbers that arise when computing sums of the form

$$1^m + 2^m + 3^m + \ldots + n^m$$

  - Example sums:
    - $1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$ $= \frac{1}{2}n^2 + \frac{1}{2}n$
    - $1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$ $= \frac{1}{3}n^3 + \ldots$
    - $1^3 + 2^3 + 3^3 + \ldots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$ $+ \frac{1}{4}n^4 + \ldots$
- The first 8 Bernoulli numbers:

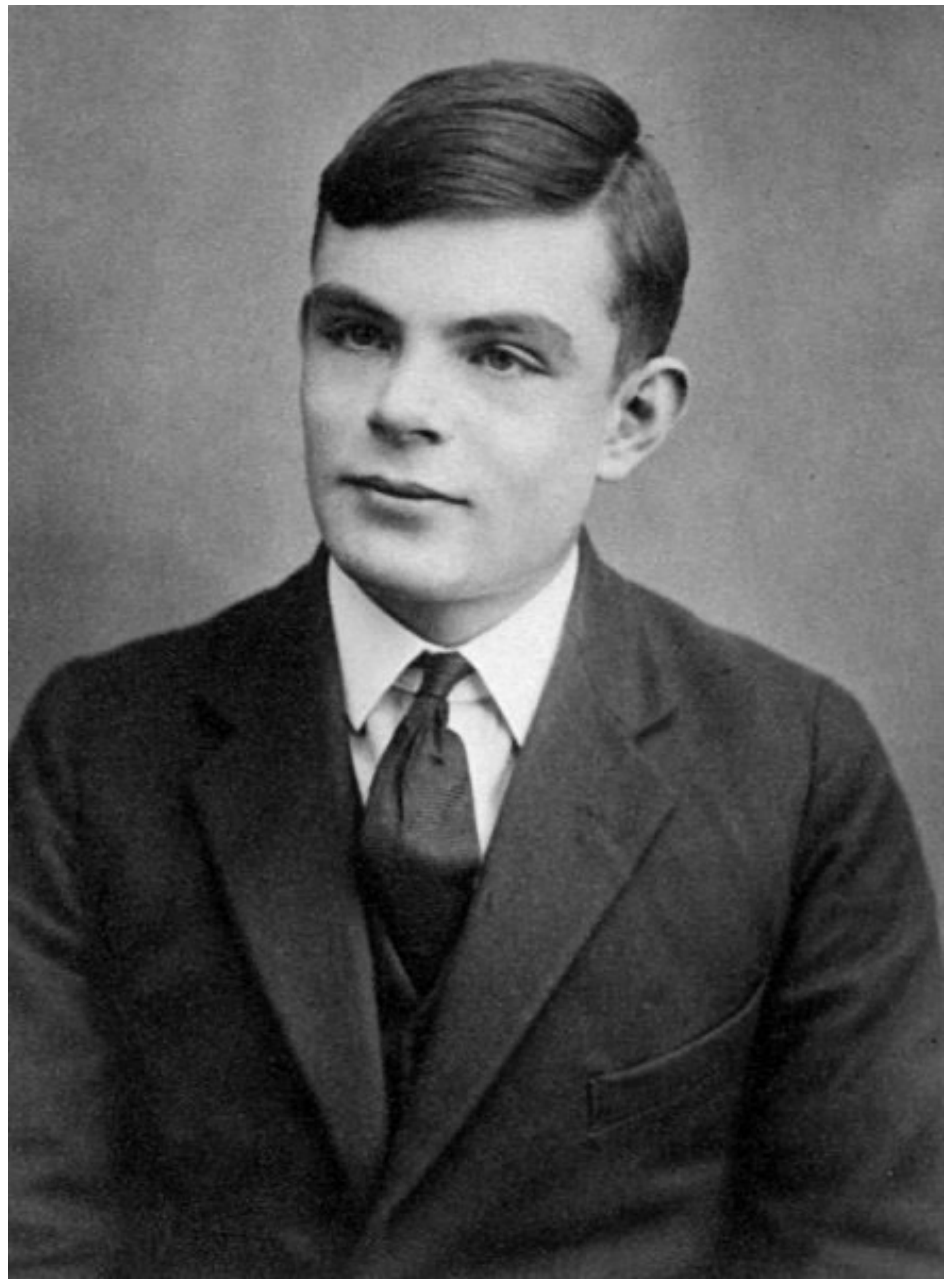
$$B_0 = 1,\ B_1 = -\frac{1}{2},\ B_2 = \frac{1}{6},\ B_3 = 0,\ B_4 - \frac{1}{30}, B_5 = 0, B_6 = \frac{1}{42},\ B_7 = 0, \ldots$$

Early-to-mid 1900s

Turing

42 years

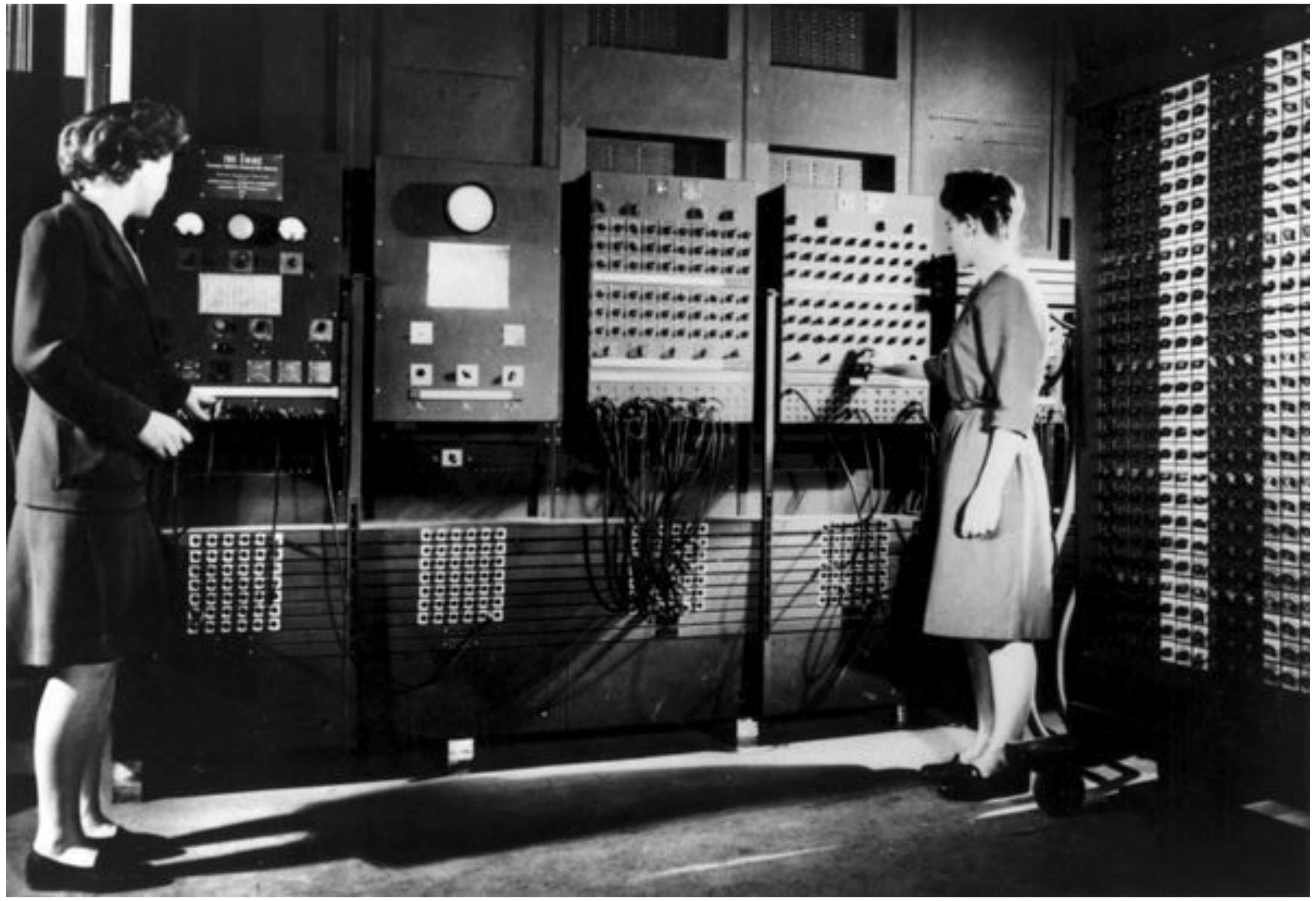
- Alan Turing (1912-1954) was an English mathematician, and is known as the founder of **theoretical computer science**.
    - The "Nobel Prize in CS" is called the Turing Award.
- He ideated the **Turing machine**, a mathematical model of a computer forms the basis of modern computation.
    - Big idea: there are computer programs that "don't exist" (e.g. the Halting problem). "halting problem"
- He also helped crack the **Enigma**, a machine the Germans used during WWII to encrypt their morse code messages.
    - This helped shorten WWII by an estimated two years (saving 10s of millions of lives).



Bletchley Park, the headquarters of the Allies’ codebreaking efforts during WWII.

ENIAC

- In 1945, ENIAC (Electronic Numerical Integrator and Calculator) was designed and built at the University of Pennsylvania by John Mauchly and John Presper Eckert.
  - It is the first "programmable, general-purpose electronic digital computer." It could perform 5,000 additions per second.
  - It was massive – it took up 1800 square feet of space and weighed 30 tons.
  - The US Military funded the ENIAC project as it would help them quickly calculate artillery firing ranges (though the ENIAC was only completed just after WWII).



Jean Bartik and Frances Spence programming the ENIAC.

Hopper

- Grace Hopper (1906-1992) was an American computer scientist and Navy officer.
  - She earned her PhD in math at Yale, and was a programmer for the Harvard Mark I and II, an early electromechanical computer developed by IBM.
  - Afterwards, she joined the Eckert-Mauchly Computer Corporation, and while there she developed the idea of a programming language that looked like English, as the current method of programming was very symbol-heavy. The result was COBOL, which is still in use today ‼️

- Fun fact: she discovered the first "bug", which was a moth that snuck into the Harvard Mark II.

↳ punchcard bugs ?

yellkey.com/special
word

COBOL (Common Business-Oriented Language)

"It's much easier for most people to write an English statement than it is to use symbols. So I decided data processors ought to be able to write their programs in English, and the computers would translate them into machine code." – Hopper (source)

Many banks and government agencies **still** rely on code written in COBOL.

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. ADD_NUMBERS.
       DATA DIVISION.
       FILE SECTION.
       WORKING-STORAGE SECTION.
       01   FIRST-NUMBER     PICTURE IS 99.
       01   SECOND-NUMBER    PICTURE IS 99.
       01   RESULT           PICTURE IS 9999.
       PROCEDURE DIVISION.

       MAIN-PROCEDURE.
           DISPLAY "Here is the first Number "
           MOVE 8 TO FIRST-NUMBER
           DISPLAY FIRST-NUMBER

           DISPLAY "Let's add 20 to that number."
           ADD 20 TO FIRST-NUMBER
           DISPLAY FIRST-NUMBER

           DISPLAY "Create a second variable"
           MOVE 30 TO SECOND-NUMBER
           DISPLAY SECOND-NUMBER

           *>COMMENT: COMPUTE THE TWO NUMBER AND PLACE INTO RESULT*
           COMPUTE RESULT = FIRST-NUMBER +  SECOND-NUMBER.

           DISPLAY "The result is:".
           DISPLAY RESULT.
        STOP RUN.
END PROGRAM ADD_NUMBERS.
```

That’s all!