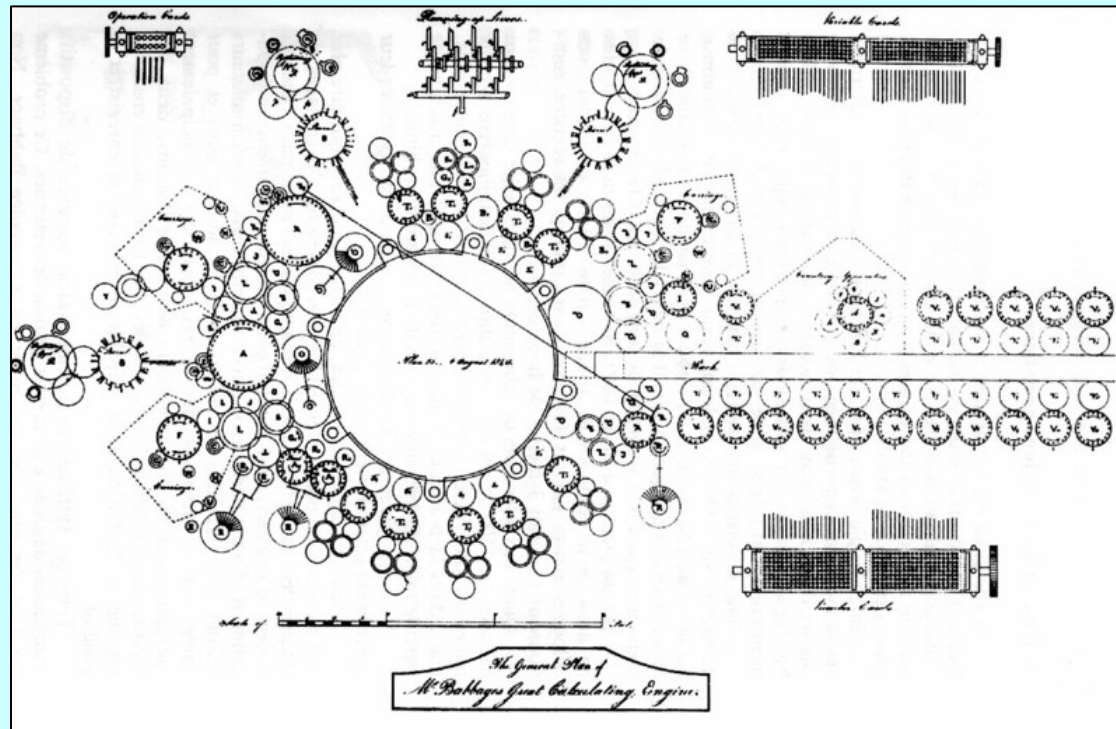


# The Analytical Engine



Chris Gregg, based on slides by Eric Roberts

CS 208E

September 23, 2021

# CS 208E—Topic Overview

Week 1	Introductions; course overview; Babbage machines; Ada Lovelace
Week 2	Karel the Robot; beginning JavaScript; algorithms
Week 3	Binary arithmetic; digital logic
Week 4	Stored-program machines; the Toddler machine
Week 5	Guest Lecture: Dr. Enigma, Turing machines; the Busy Beaver problem; undecidability
Week 6	Computational complexity; the $P = NP$ question
Week 7	Cryptography; public-key cryptography; digital signatures
Week 8	Networking; networking algorithms; Artificial Intelligence
Week 9	Early Programming Languages / Reflections on Trusting Trust
Week 10	Historiography of Computing



**Augusta Ada Byron,  
Lady Lovelace (1815–1852)**

Augusta Ada Byron, the daughter of the English poet Lord Byron and his wife Anne, was encouraged to pursue her interests in science and mathematics at a time when few women were allowed to study those subjects. At the age of 17, Ada met Charles Babbage and became fascinated by his machines. Ada was convinced of the potential of Babbage's Analytical Engine and wrote extensive notes on its design, along with several complex mathematical programs that have led many people to characterize her as the first programmer. In 1980, the U.S. Department of Defense named the programming language Ada in her honor.

# Ada Lovelace: The First Programmer

<https://www.youtube.com/watch?v=IZptxisyVqQ>

# The Analytical Engine

- As Babbage built prototypes of his Difference Engine, he began to envision a much more powerful computing device he called the Analytical Engine.
- Babbage's initial notes on the Analytical Engine appear in 1837, but the most complete description appears in a 1842 paper by Luigi Federico Menabrea, who was reporting on a lecture Babbage gave in 1840. Ada Lovelace translated Menabrea's paper from French into English and provided notes that were three times longer than the original.
- The essential difference between the Difference Engine and the Analytical Engine is that the Analytical Engine was designed to be *programmable*, allowing users to perform any sequence of calculations. The programs were encoded on punched cards in the manner of the Jacquard loom, which Ada and her mother had seen in their visits to the English industrial areas.



# Jacquard Loom

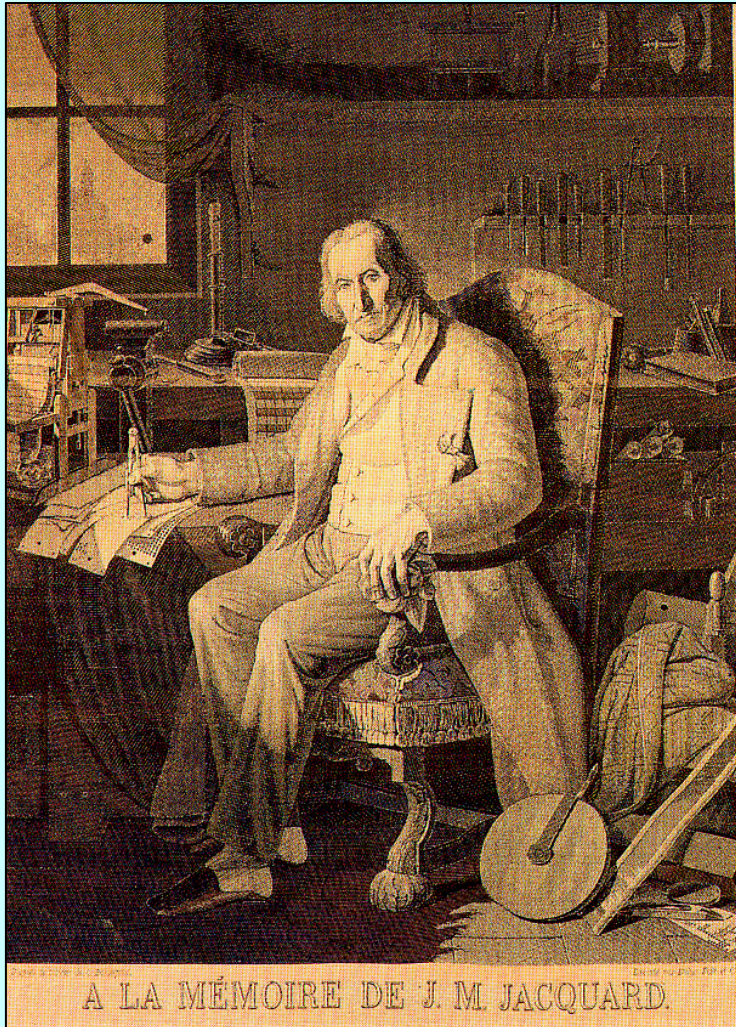


# Jacquard Loom

<https://www.youtube.com/watch?v=K6NgMNvK52A>

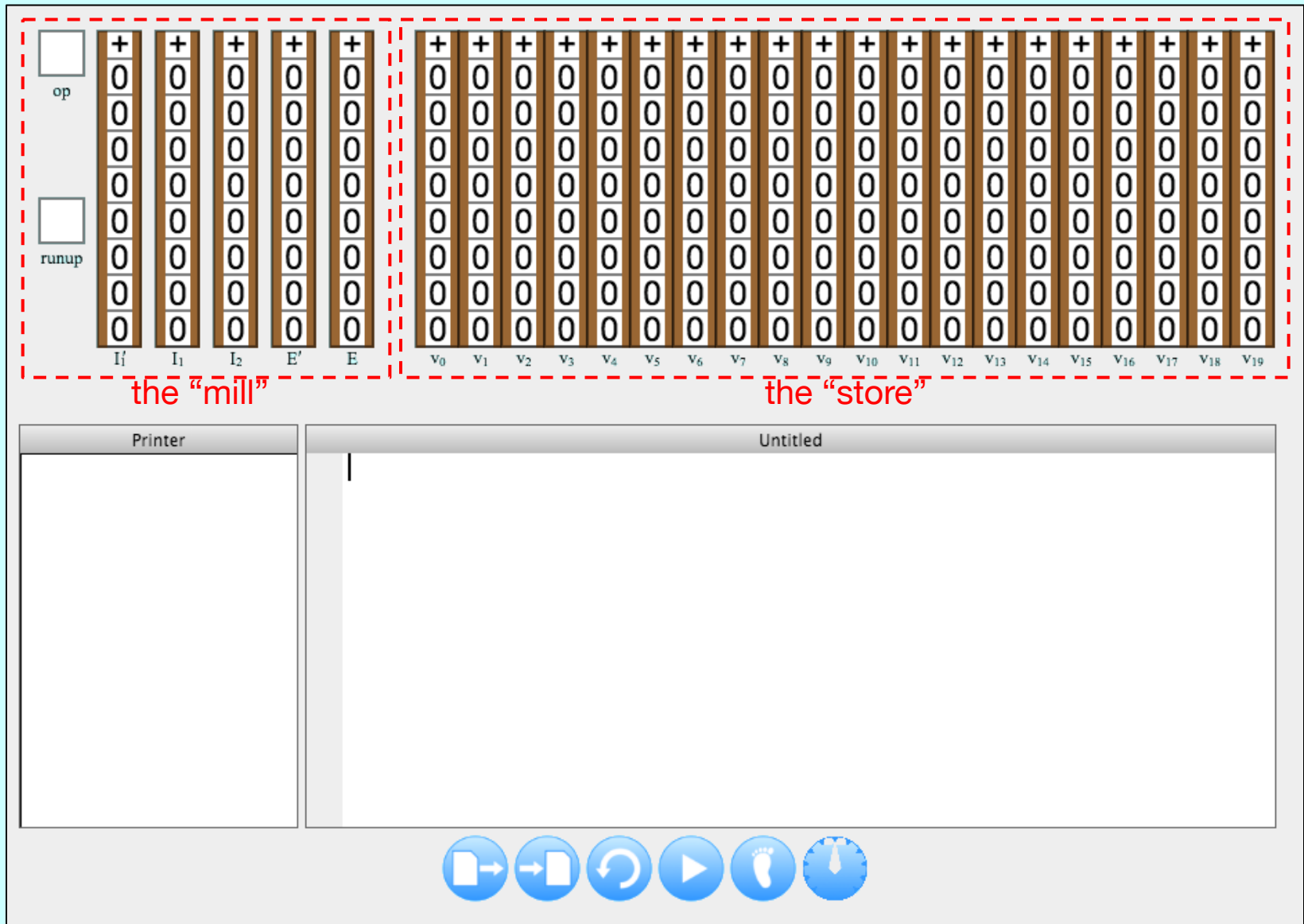


# Products of the Jacquard Loom





# Structure of the Analytical Engine



# The Store and the Mill



# The Analytical Engine: Computerphile

<https://www.youtube.com/watch?v=5rtKoKFGFSM>

Notes:

<http://www.eprg.org/computerphile/new-adataalk.pdf>

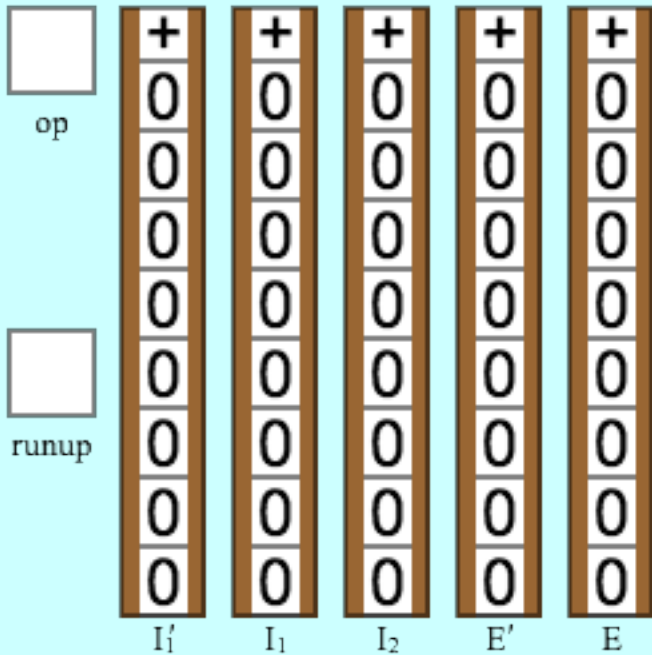
# The “Store”

+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$v_0$	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_6$	$v_7$	$v_8$	$v_9$	$v_{10}$	$v_{11}$	$v_{12}$	$v_{13}$	$v_{14}$	$v_{15}$	$v_{16}$	$v_{17}$	$v_{18}$	$v_{19}$

- Each column holds a single integer as in the Difference Engine.
- Numbers in the Analytical Engine are signed.
- Each column has a numeric address:  $v_0$ ,  $v_1$ ,  $v_2$ ,  $v_3$ , and so on.



# The “Mill”



- The **op** indicator holds the current operation (+, −, ×, ÷)
- The mill has five columns:
  - I<sub>1</sub> and I<sub>2</sub> are the input values
  - E is the output value
  - I<sub>1</sub>' and E' are used to store extra digits for the × and ÷ operations
- The **runup** indicator is set when the result of an operation changes sign.

# Instructions for the Analytical Engine

<b>N</b> <i>address value</i>	Store <i>value</i> in <i>address</i>
<b>+</b>	Set the machine to addition
<b>-</b>	Set the machine to subtraction
<b>×</b>	Set the machine to multiplication
<b>÷</b>	Set the machine to division
<b>L</b> <i>address</i>	Load from <i>address</i> , preserving data
<b>Z</b> <i>address</i>	Load from <i>address</i> , clearing data
<b>S</b> <i>address</i>	Store egress register in <i>address</i>
<b>P</b> <i>address</i>	Print value in <i>address</i>

# Program to Add Two Numbers

```
N 0 25 /* First number is in v0 */
N 1 17 /* Second number is in v1 */

+ /* Set machine for addition */
L 0 /* Load first number into I1 */
L 1 /* A second load does the add */
S 2 /* Store result in v2 */
P 2 /* Print the result
```

# Adding Control Operations

<b>N</b> <i>address value</i>	Store <i>value</i> in <i>address</i>
<b>+</b>	Set the machine to addition
<b>-</b>	Set the machine to subtraction
<b>×</b>	Set the machine to multiplication
<b>÷</b>	Set the machine to division
<b>L</b> <i>address</i>	Load from <i>address</i> , preserving data
<b>Z</b> <i>address</i>	Load from <i>address</i> , clearing data
<b>S</b> <i>address</i>	Store egress register in <i>address</i>
<b>P</b> <i>address</i>	Print value in <i>address</i>
<b>B</b> <i>number</i>	Move backward specified number of cards
<b>F</b> <i>number</i>	Move forward specified number of cards
<b>?B</b> <i>number</i>	Move backward if <b>runup</b> lever is set
<b>?F</b> <i>number</i>	Move forward if <b>runup</b> lever is set



# Exercise: Produce a Table of Squares

Use the Analytical Engine to produce a table of squares.

Printer
0
1
4
9
16
25
49
64
81

The simplest approach is to simulate the operation used by the Difference Engine to accomplish the same task.

# Multiplication and Division

- Babbage recognized that multiplying two integers produces a result that typically has twice the number of digits that appear in the original values.
- To take account of this fact, the multiplication operation for the Analytical Engine produces its result in a pair of columns. Column E shows the *low-order digits* of the product, which are the digits on the right that include the units value. Column E' shows the *high-order digits*.
- The division operation uses two columns to store the dividend. You set up the division by loading the low-order digits into column  $I_1$  and the high-order digits (if any) into column  $I_1'$ .
- As long as you know that the numbers won't exceed the number of digits in a column, you can ignore E' and  $I_1'$  altogether.

# Exercise: Calculate Factorials

How would you program the Analytical Engine to calculate the factorial of a number supplied as data using a number card.

$$N! = 1 \times 2 \times 3 \times \dots \times N$$

# Bernoulli Numbers

Bernoulli number - Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/Bernoulli\_number

Article Talk

Read Edit View history

Search

## Bernoulli number

From Wikipedia, the free encyclopedia

In **mathematics**, the **Bernoulli numbers**  $B_n$  are a **sequence of rational numbers** with deep connections to **number theory**. The values of the first few Bernoulli numbers are

$$B_0 = 1, B_1 = \pm\frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0, B_4 = -\frac{1}{30}, B_5 = 0, B_6 = \frac{1}{42}, B_7 = 0, B_8 = -\frac{1}{30}.$$

If the convention  $B_1 = -\frac{1}{2}$  is used, this sequence is also known as the **first Bernoulli numbers** (A027641 / A027642 in OEIS); with the convention  $B_1 = +\frac{1}{2}$  is known as the **second Bernoulli numbers** (A164555 / A027642). Except for this one difference, the first and second Bernoulli numbers agree. Since  $B_n = 0$  for all odd  $n > 1$ , and many formulas only involve even-index Bernoulli numbers, some authors write  $B_n$  instead of  $B_{2n}$ .

The Bernoulli numbers appear in the **Taylor series** expansions of the **tangent** and **hyperbolic tangent** functions, in formulas for the sum of powers of the first positive integers, in the **Euler–Maclaurin formula**, and in expressions for certain values of the **Riemann zeta function**.

The Bernoulli numbers were discovered around the same time by the Swiss mathematician **Jakob Bernoulli**, after whom they are named, and independently by Japanese mathematician **Seki Kōwa**. Seki's discovery was posthumously published in 1712<sup>[1][2]</sup> in his work *Katsuyo Sampo*, Bernoulli's, also posthumously, in his *Ars Conjectandi* of 1713. **Ada Lovelace's note G** on the **analytical engine** from 1842 describes an algorithm for generating Bernoulli numbers with **Babbage's machine**.<sup>[3]</sup> As a result, the Bernoulli numbers have the distinction of being the subject of one of the first **computer programs**.

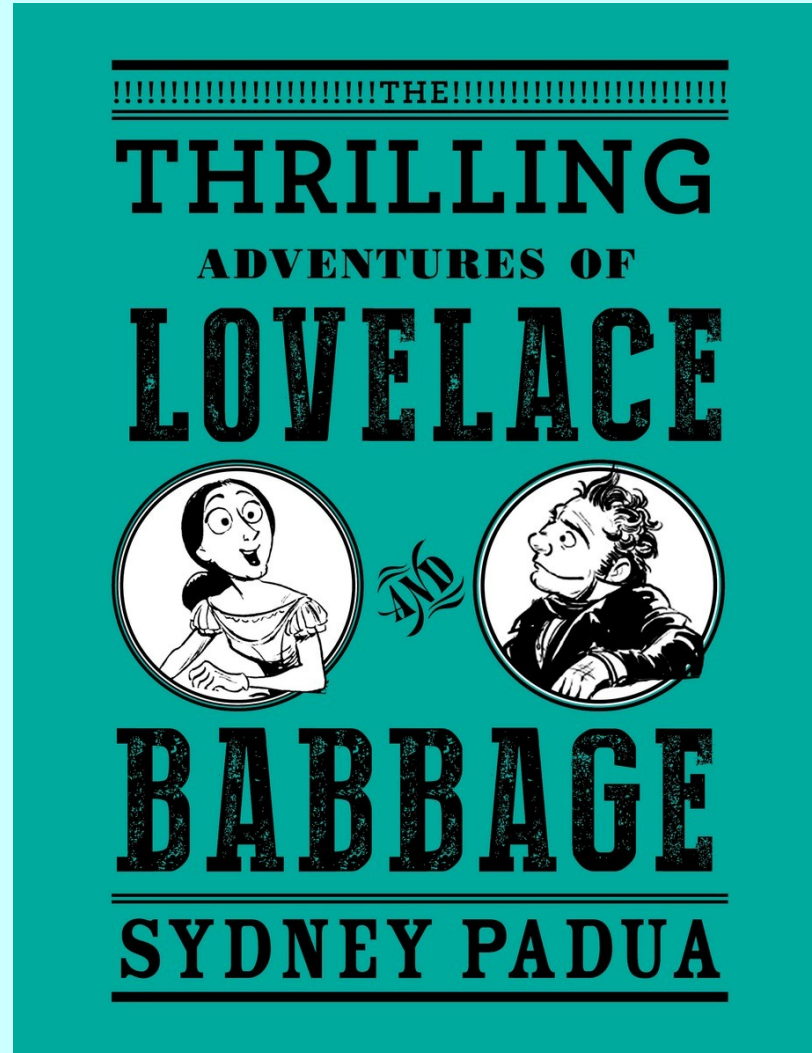
n	$B_n$
0	1
1	$\pm\frac{1}{2}$
2	$\frac{1}{6}$
3	0
4	$-\frac{1}{30}$
5	0
6	$\frac{1}{42}$
7	0
8	$-\frac{1}{30}$



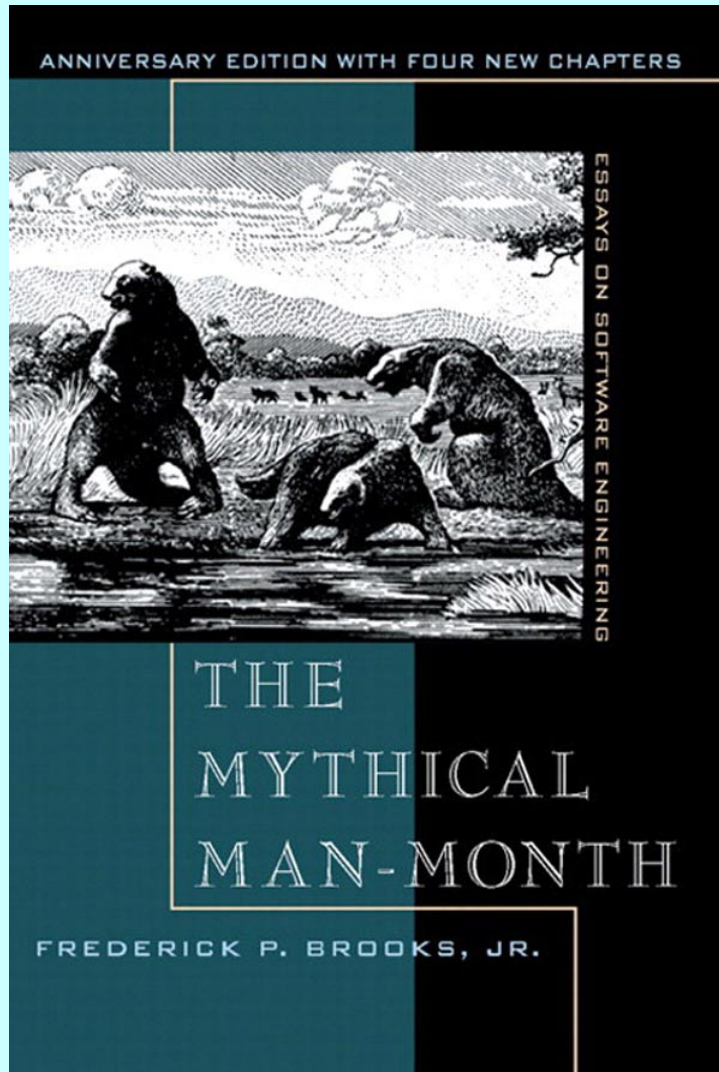
# Ada's Program for Bernoulli Numbers

Number of Operations. Nature of Operations.		Variables for Data.						Working Variables.									Variables for Results.		
		<sup>1</sup> V <sub>0</sub>	<sup>1</sup> V <sub>1</sub>	<sup>1</sup> V <sub>2</sub>	<sup>1</sup> V <sub>3</sub>	<sup>1</sup> V <sub>4</sub>	<sup>1</sup> V <sub>5</sub>	<sup>0</sup> V <sub>6</sub>	<sup>0</sup> V <sub>7</sub>	<sup>0</sup> V <sub>8</sub>	<sup>0</sup> V <sub>9</sub>	<sup>0</sup> V <sub>10</sub>	<sup>0</sup> V <sub>11</sub>	<sup>0</sup> V <sub>12</sub>	<sup>0</sup> V <sub>13</sub>	<sup>0</sup> V <sub>14</sub>	<sup>0</sup> V <sub>15</sub>	<sup>0</sup> V <sub>16</sub>	
		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		m	n	d	m'	n'	d'										$\frac{dn'-d'n}{mn'-m'n} = x$	$\frac{d'm-dm'}{mn'-m'n} = y$	
1	x	m	.....	.....	.....	n'	.....	mn'	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
2	x	.....	n	.....	.....	m'	.....	.....	m'n	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
3	x	.....	.....	d	.....	.....	.....	.....	.....	dn'	.....	.....	.....	.....	.....	.....	.....	.....	.....
4	x	.....	0	.....	.....	d'	.....	.....	.....	d'n	.....	.....	.....	.....	.....	.....	.....	.....	.....
5	x	0	.....	.....	.....	0	.....	.....	.....	.....	d'm	.....	.....	.....	.....	.....	.....	.....	.....
6	x	.....	.....	0	0	.....	.....	.....	.....	.....	.....	dm'	.....	.....	.....	.....	.....	.....	.....
7	-	.....	.....	.....	.....	.....	0	0	.....	.....	.....	.....	(mn'-m'n)	.....	.....	.....	.....	.....	.....
8	-	.....	.....	.....	.....	.....	.....	.....	0	0	.....	.....	.....	(dn'-d'n)	.....	.....	.....	.....	.....
9	-	.....	.....	.....	.....	.....	.....	.....	.....	.....	0	0	.....	.....	(d'm-dm')	.....	.....	.....	.....
10	÷	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	(mn'-m'n)	0	.....	.....	$\frac{dn'-d'n}{mn'-m'n} = x$	.....	.....
11	÷	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	0	.....	0	.....	.....	$\frac{d'm-dm'}{mn'-m'n} = y$	.....

# How the Analytical Engine Worked



# The Mythical Man-Month



## 11. Plan to Throw One Away

In most projects, the first system built is barely usable. It may be too slow, too big, awkward to use, or all three. There is no alternative but to start again, smarting but smarter, and build a redesigned version in which these problems are solved. . . .

The management question, therefore, is not *whether* to build a pilot system and throw it away. You *will* do that. The only question is whether to plan in advance to build a throwaway, or to promise to deliver the throwaway to customers.

The End