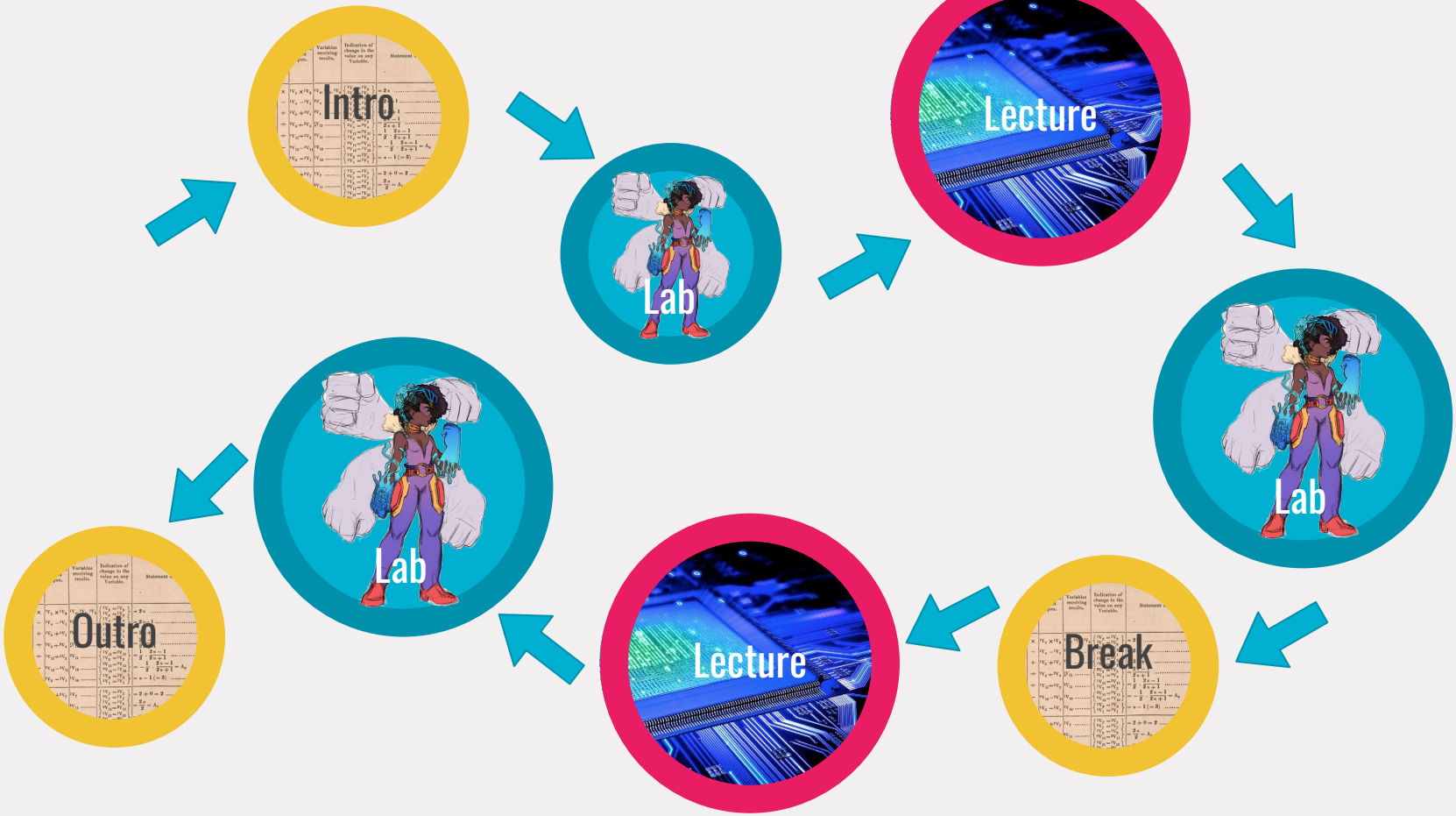
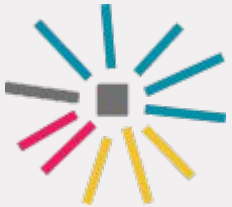

Intro
Lab
Lecture
Lab
Break
Lecture
Lab
Outro

HARDWARE

PALIMPSEST

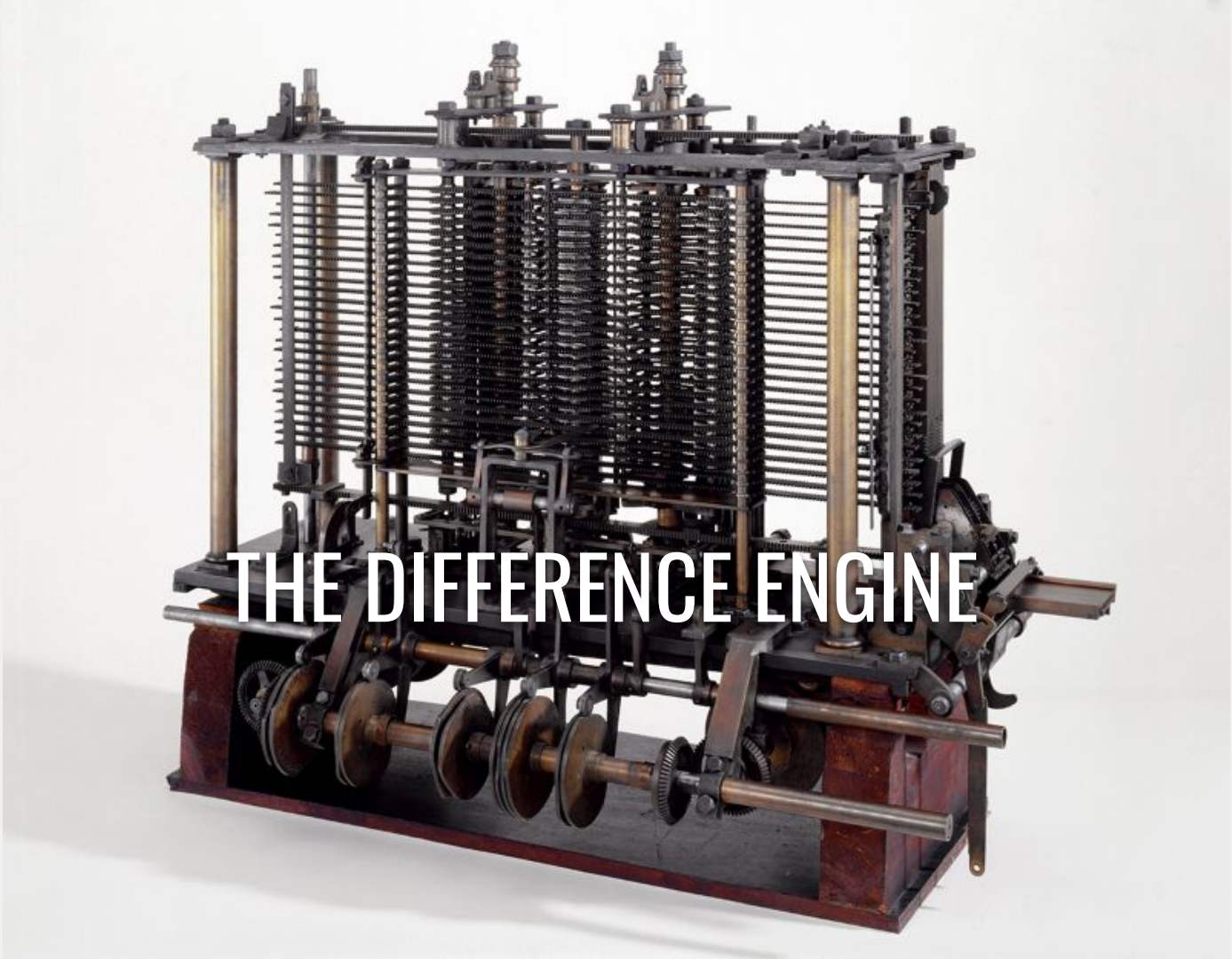


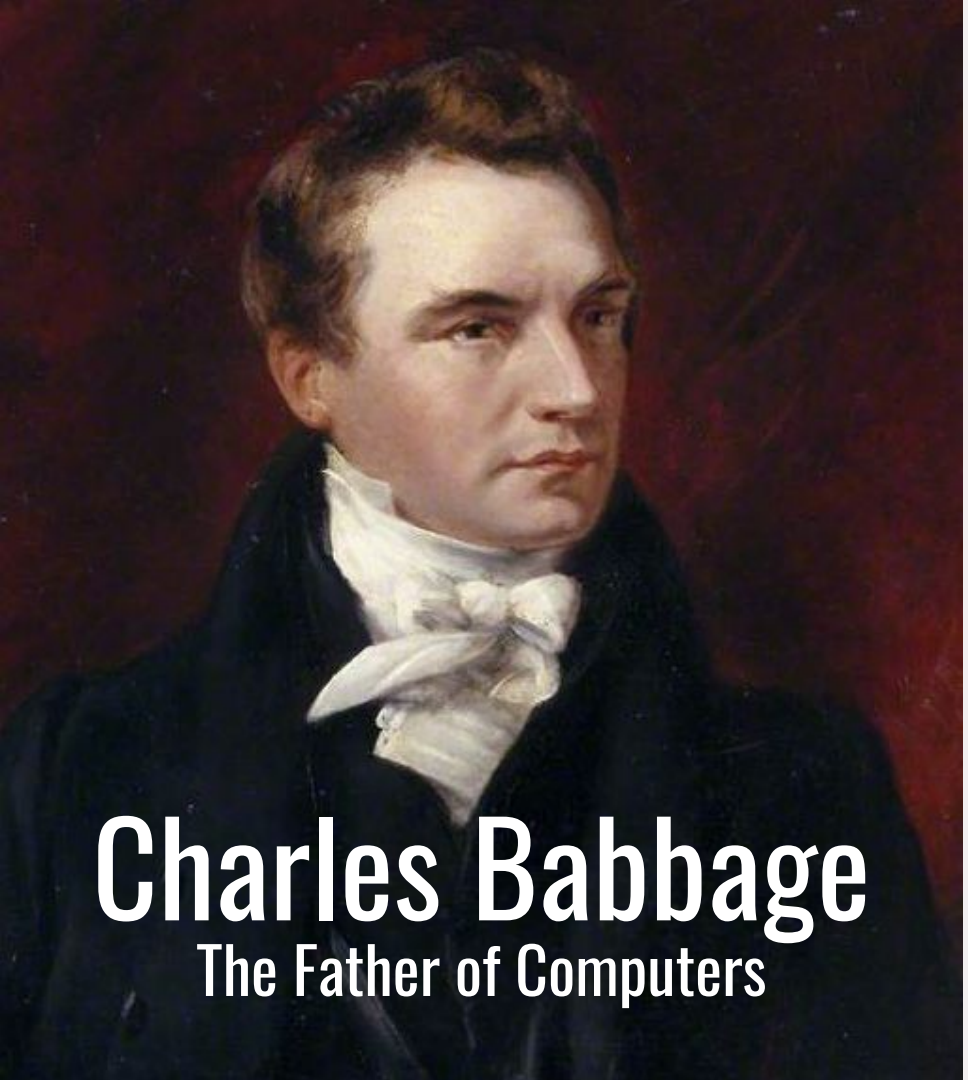
HARDWARE

What is a computer even?

- Human/Computer Relationship
- Inputs
- Storage
- Processor
- Outputs
- Binary
- Exercise

HARDWARE


THE DIFFERENCE ENGINE



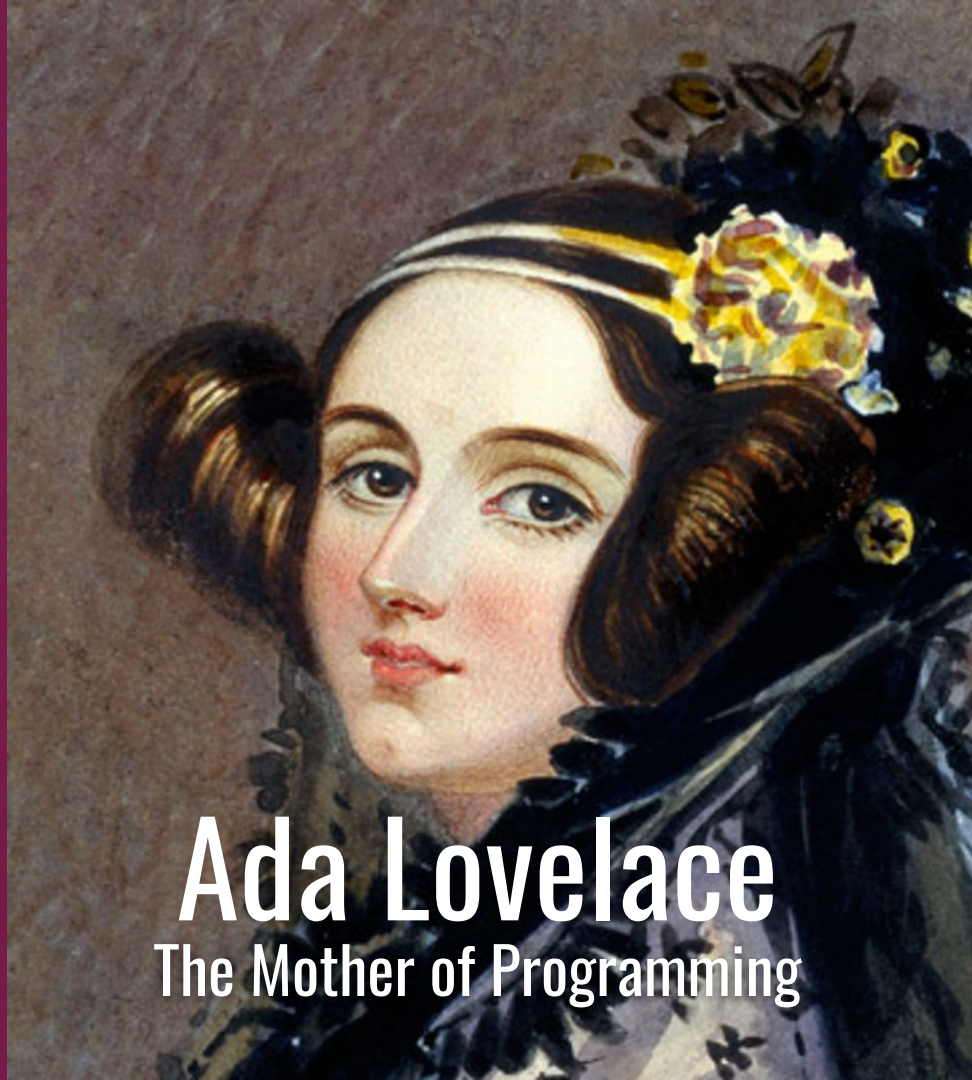
Charles Babbage

The Father of Computers

"As soon as an Analytical Engine exists, it will necessarily guide the future course of science."

“The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform...

But it is likely to exert an indirect and reciprocal influence on science itself.”

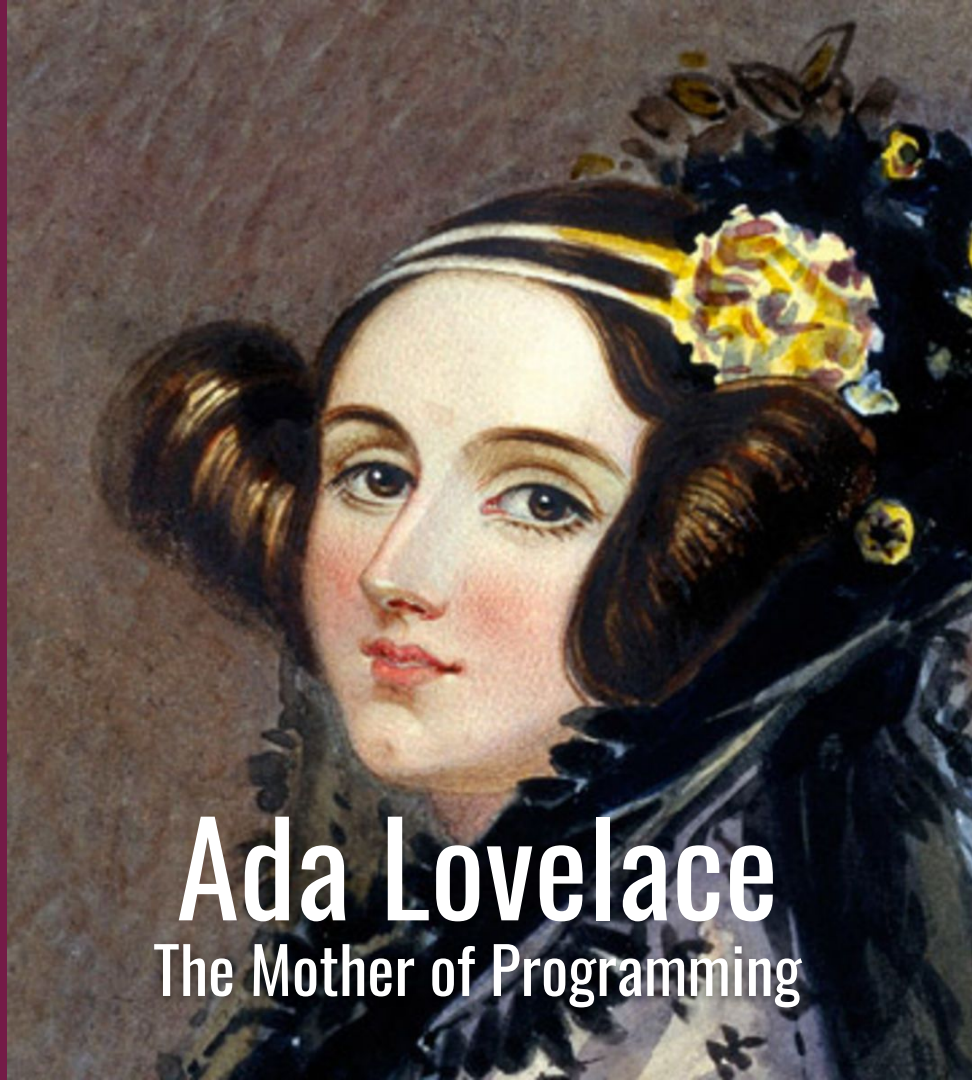


Ada Lovelace

The Mother of Programming

“The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform...

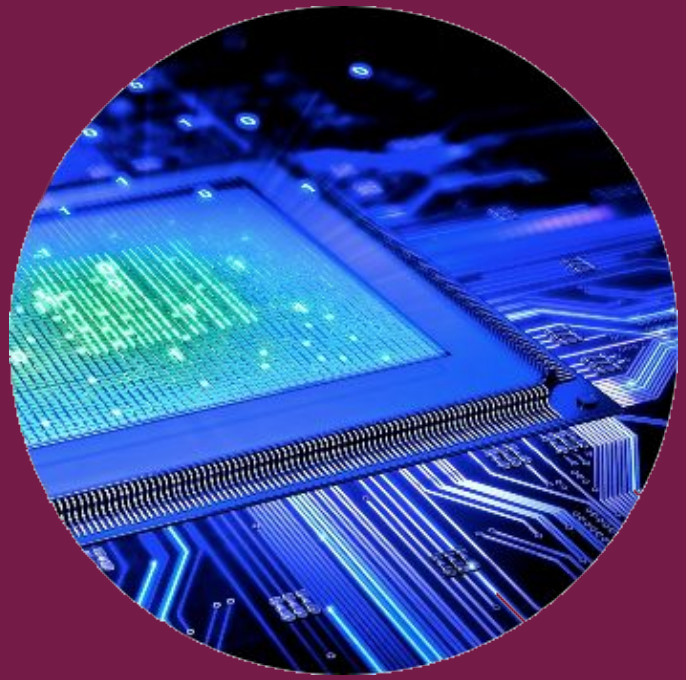
But it is likely to exert an indirect and reciprocal influence on science itself.”



Ada Lovelace

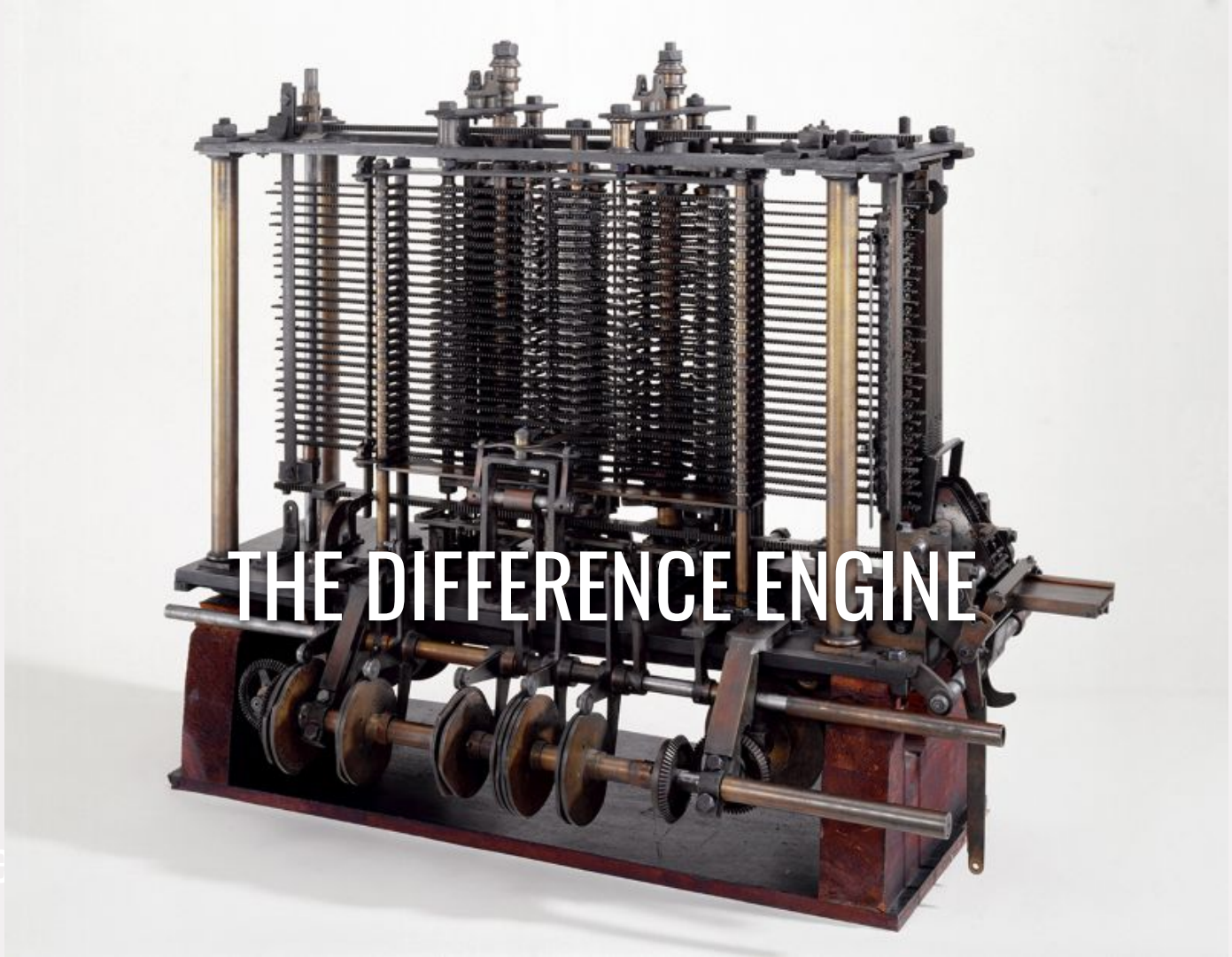
The Mother of Programming

How do we order a computer to do anything?



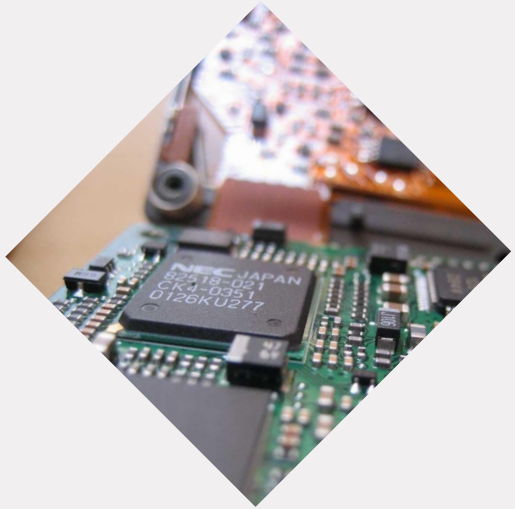
What language does it speak?

Computers speak

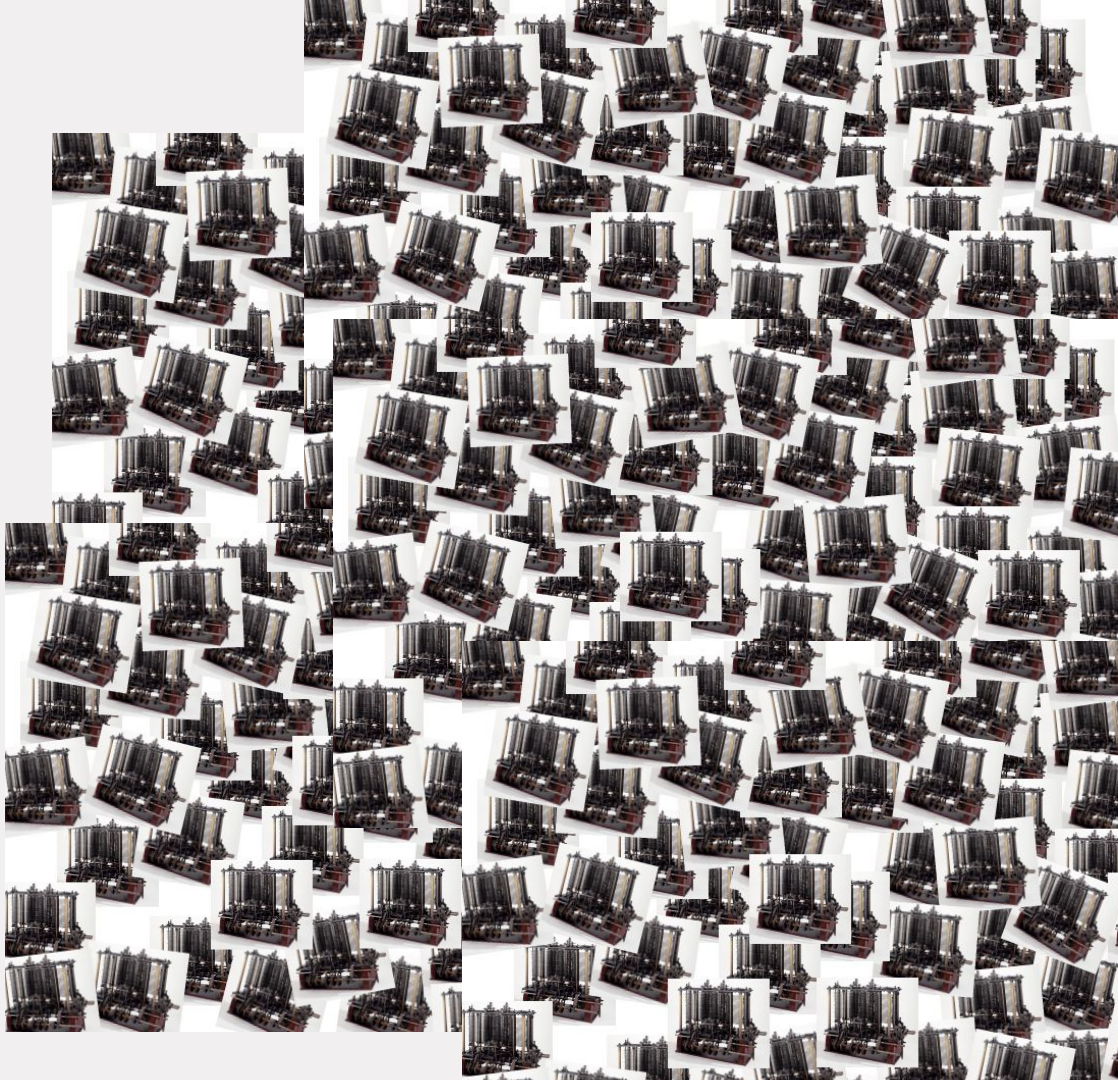


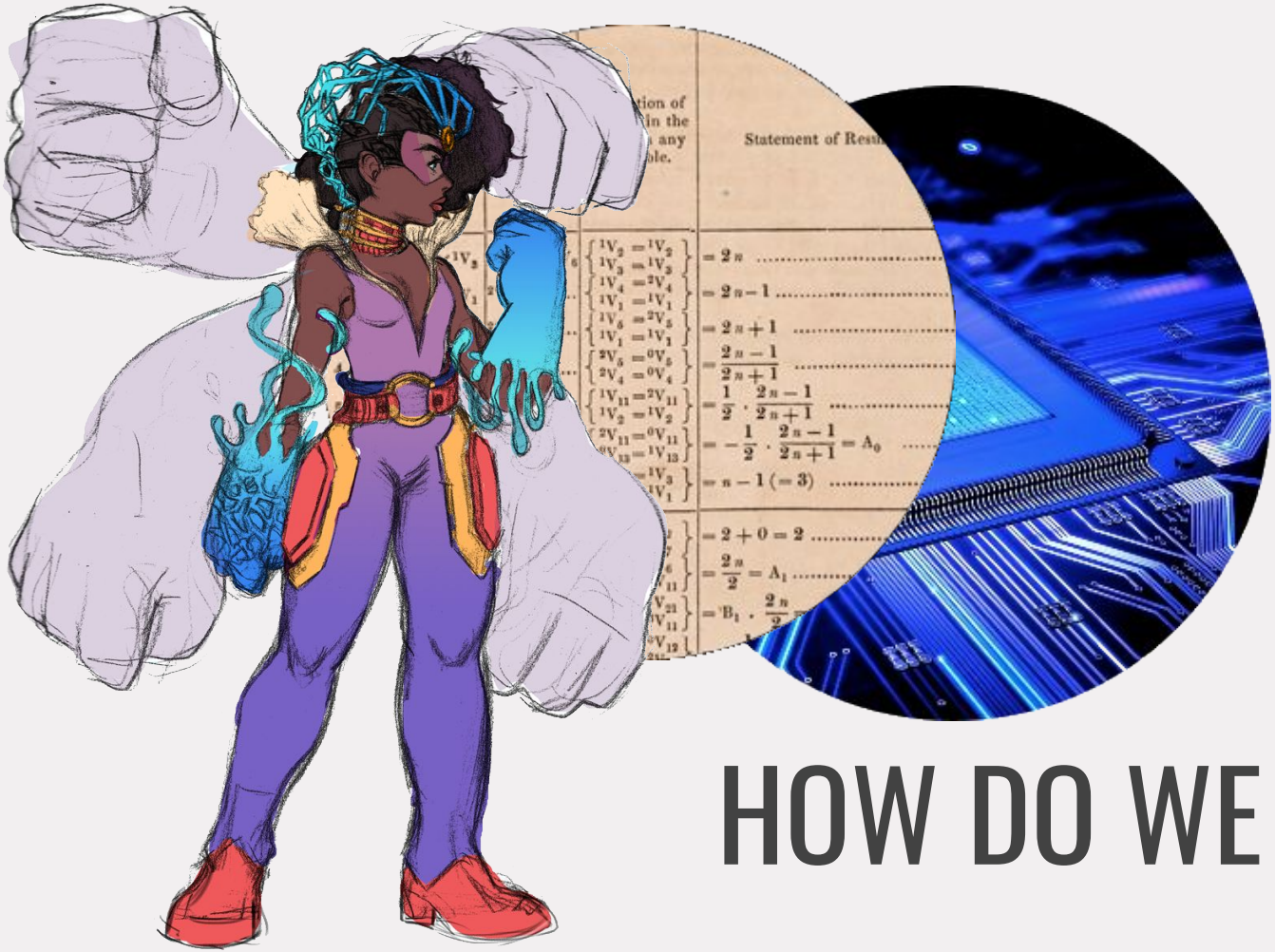
THE DIFFERENCE ENGINE

The


NEC JAPAN
8241B-051
GK-0351
0126KU277

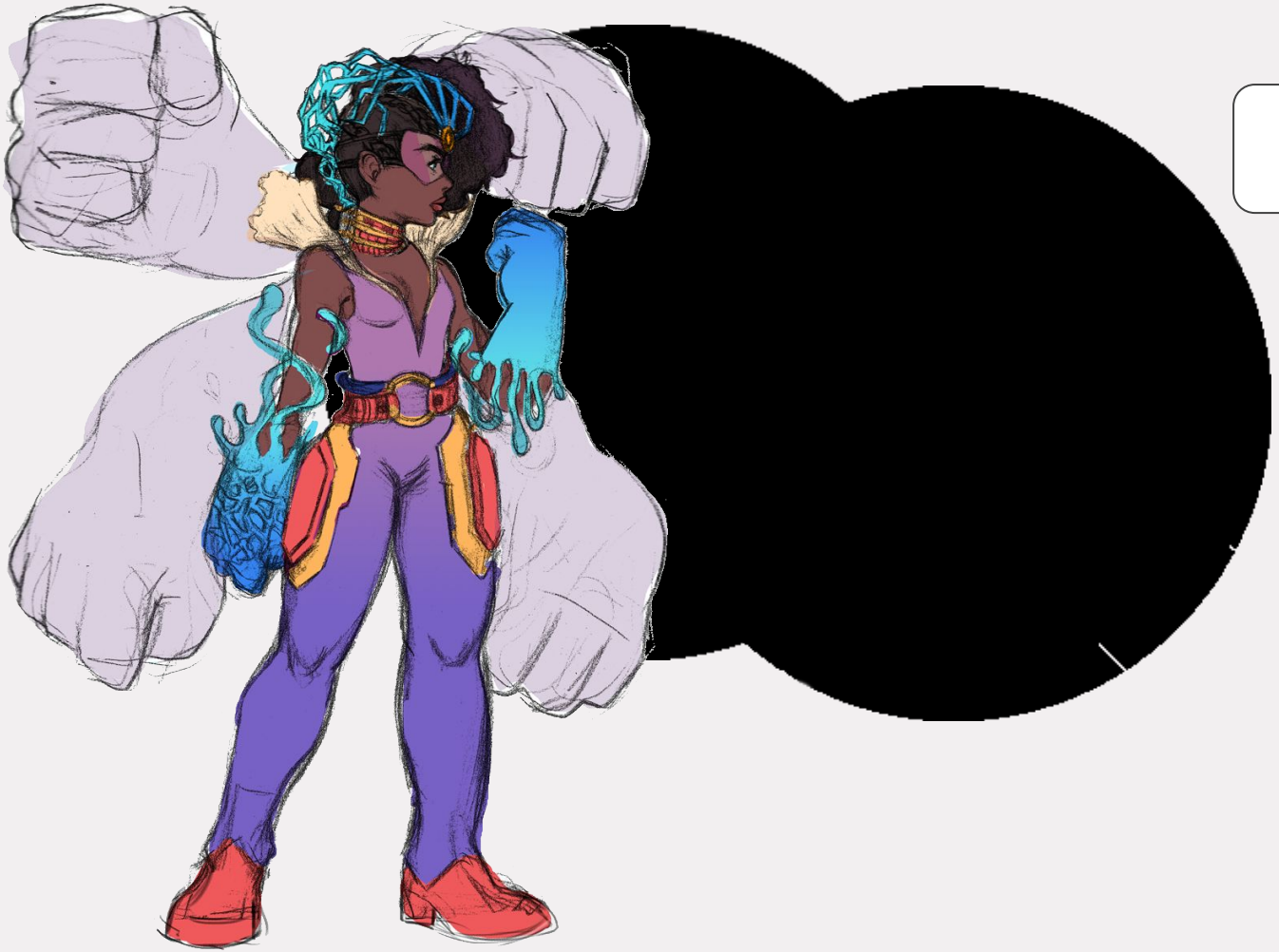
>



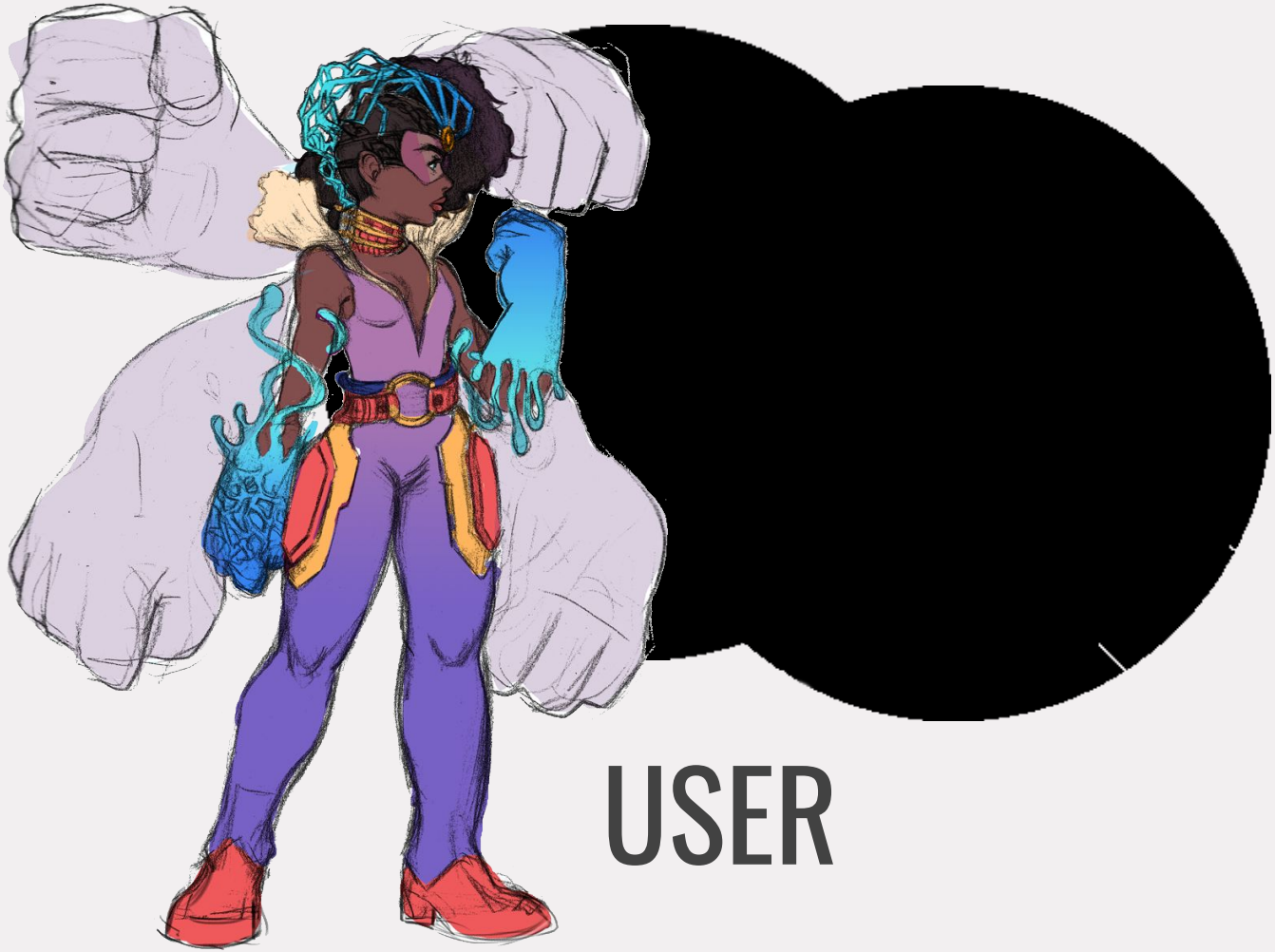


HOW DO WE FIT?

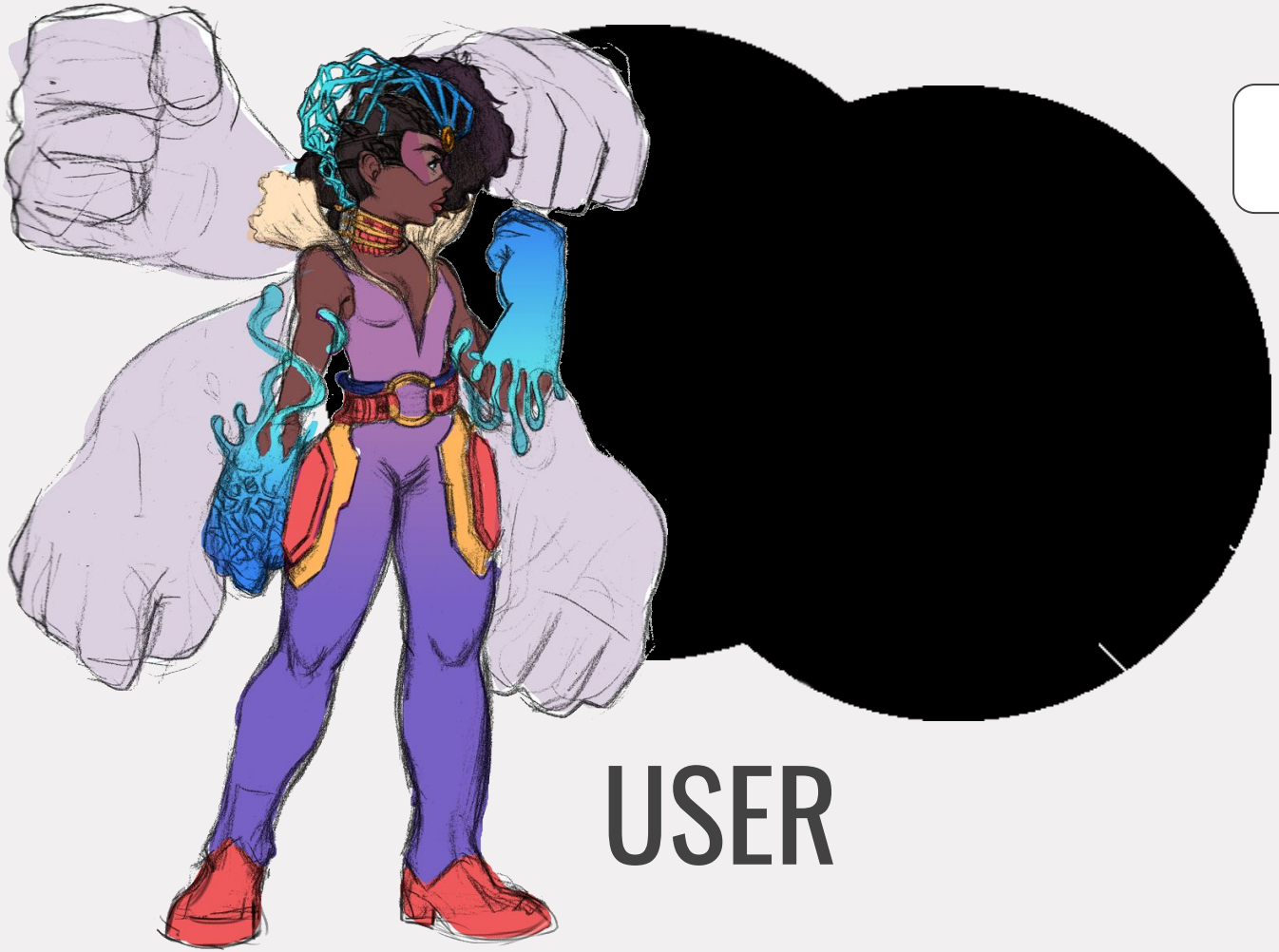

I WANT A THING.


CAN YOU CODE?


NO.

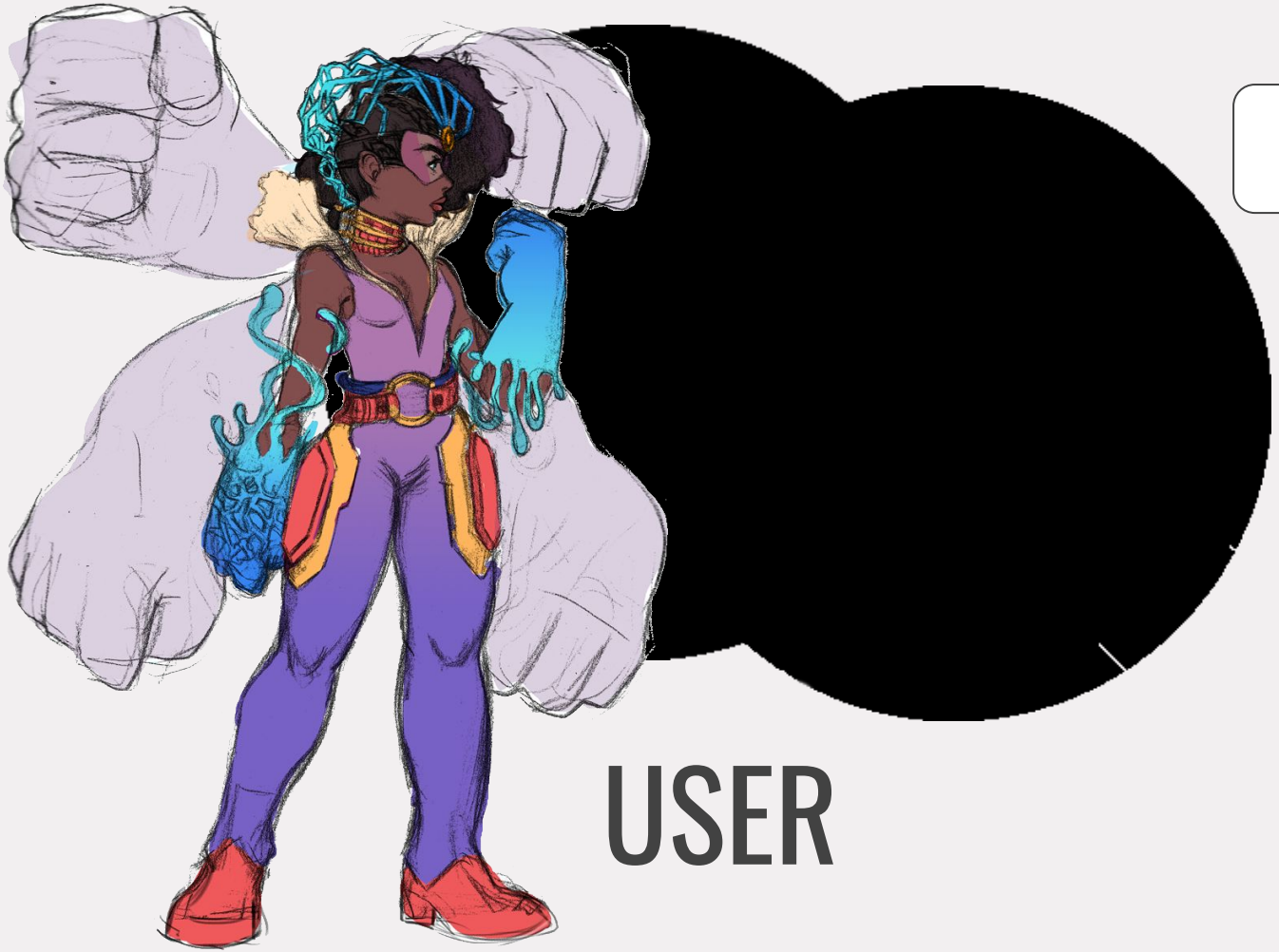


USER



GIVE ME MONEY.

USER



I WILL MAKE THE THING.

USER

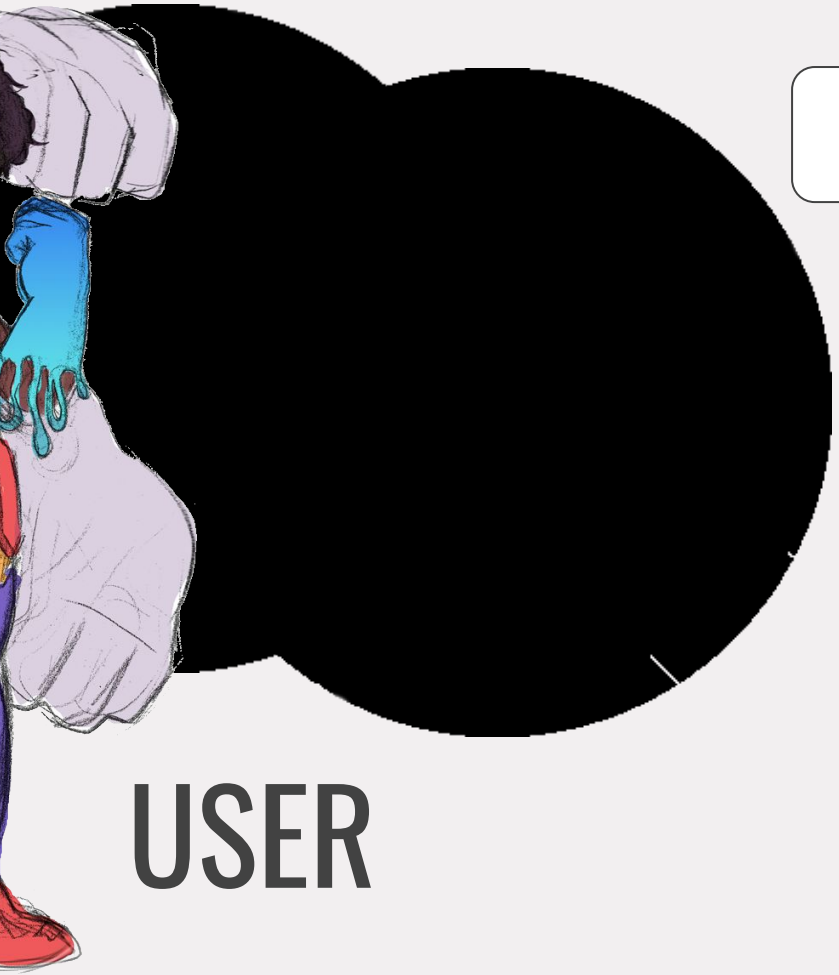
OR


YES.


I WILL MAKE THE THING.

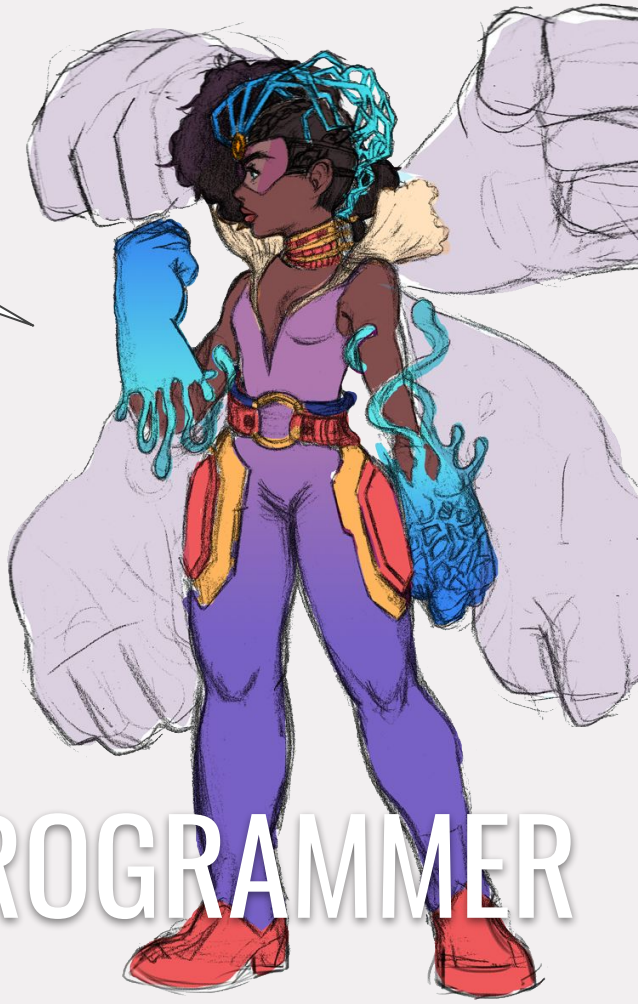
PROGRAMMER

BTW

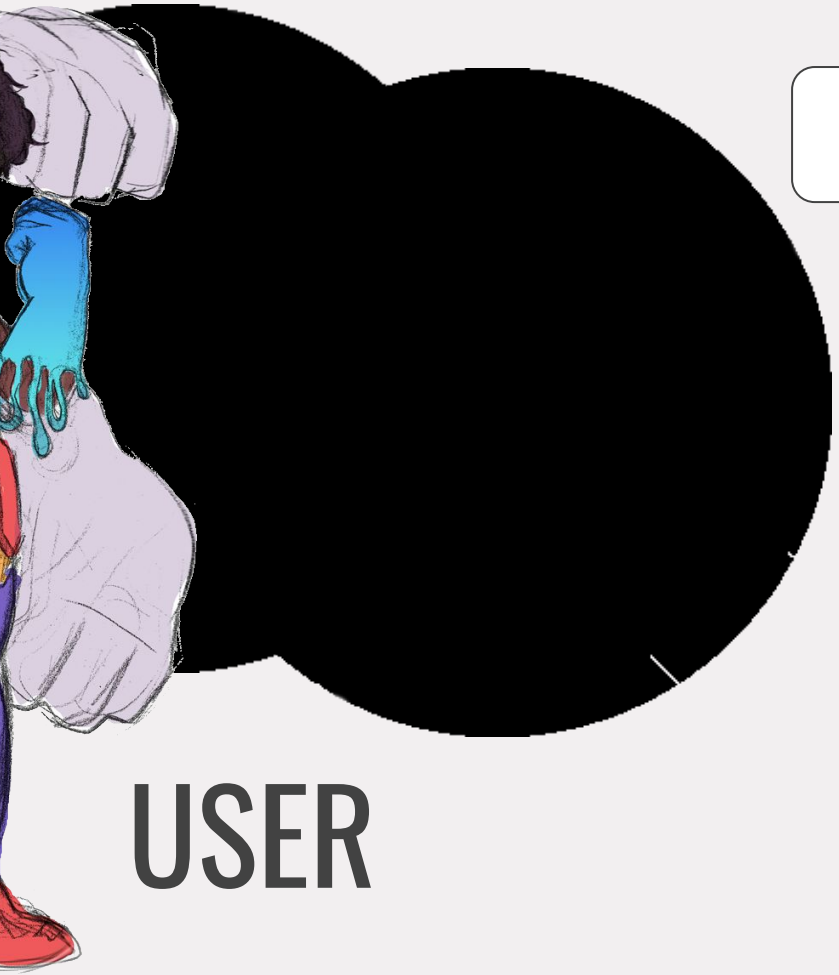


USER

GIVE ME MONEY.



PROGRAMMER



USER

I WILL MAKE THE THING.

PROGRAMMER

PROGRAMMER

PROGRAMMER

PROGRAMMER


I WILL MAKE THE THING WITH CODE.
AT HOME.
LISTENING TO MY OWN MUSIC.

PROGRAMMER

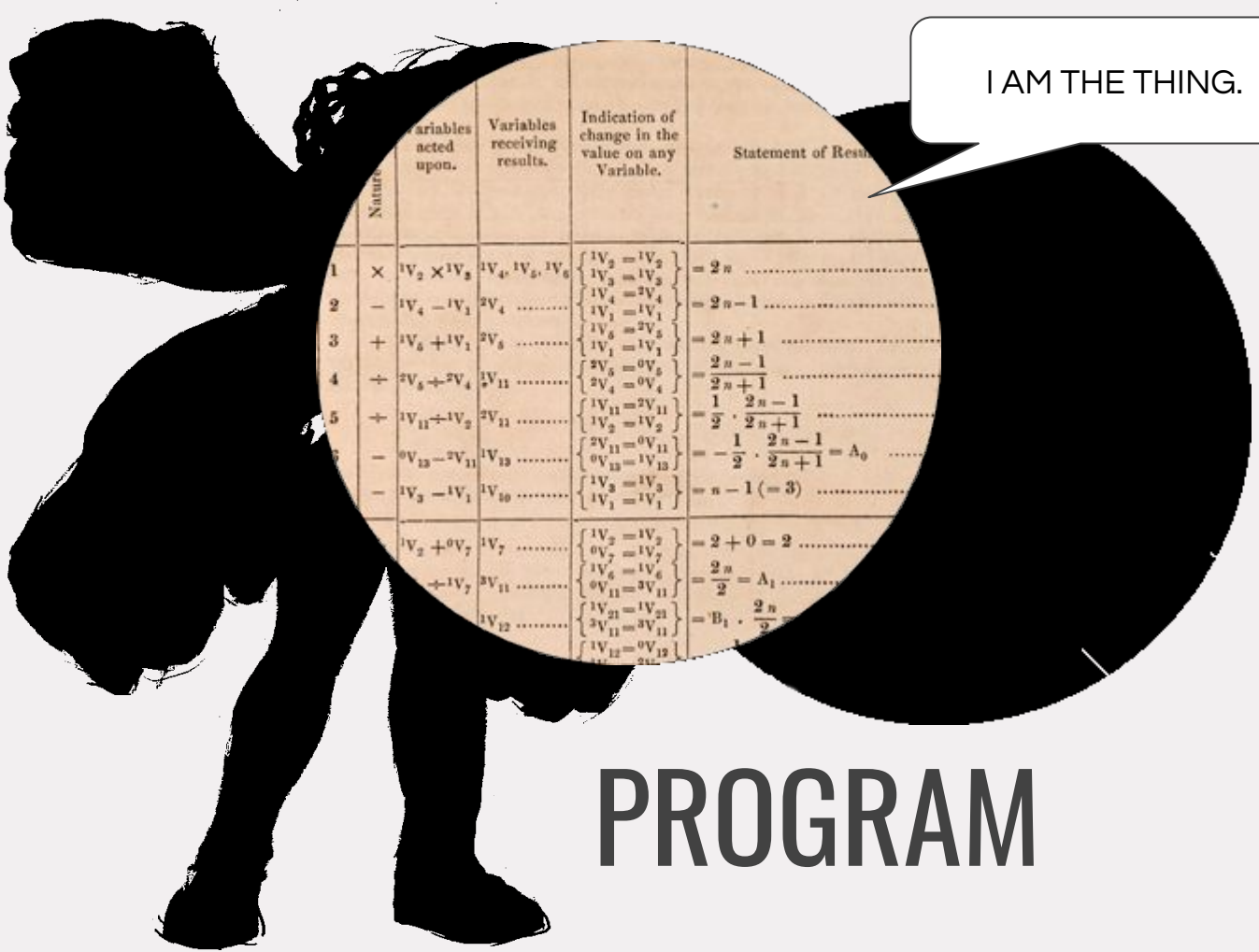

I WILL MAKE THE THING WITH CODE.
AT HOME.
LISTENING TO MY OWN MUSIC.
IN MY PJs.

PROGRAMMER

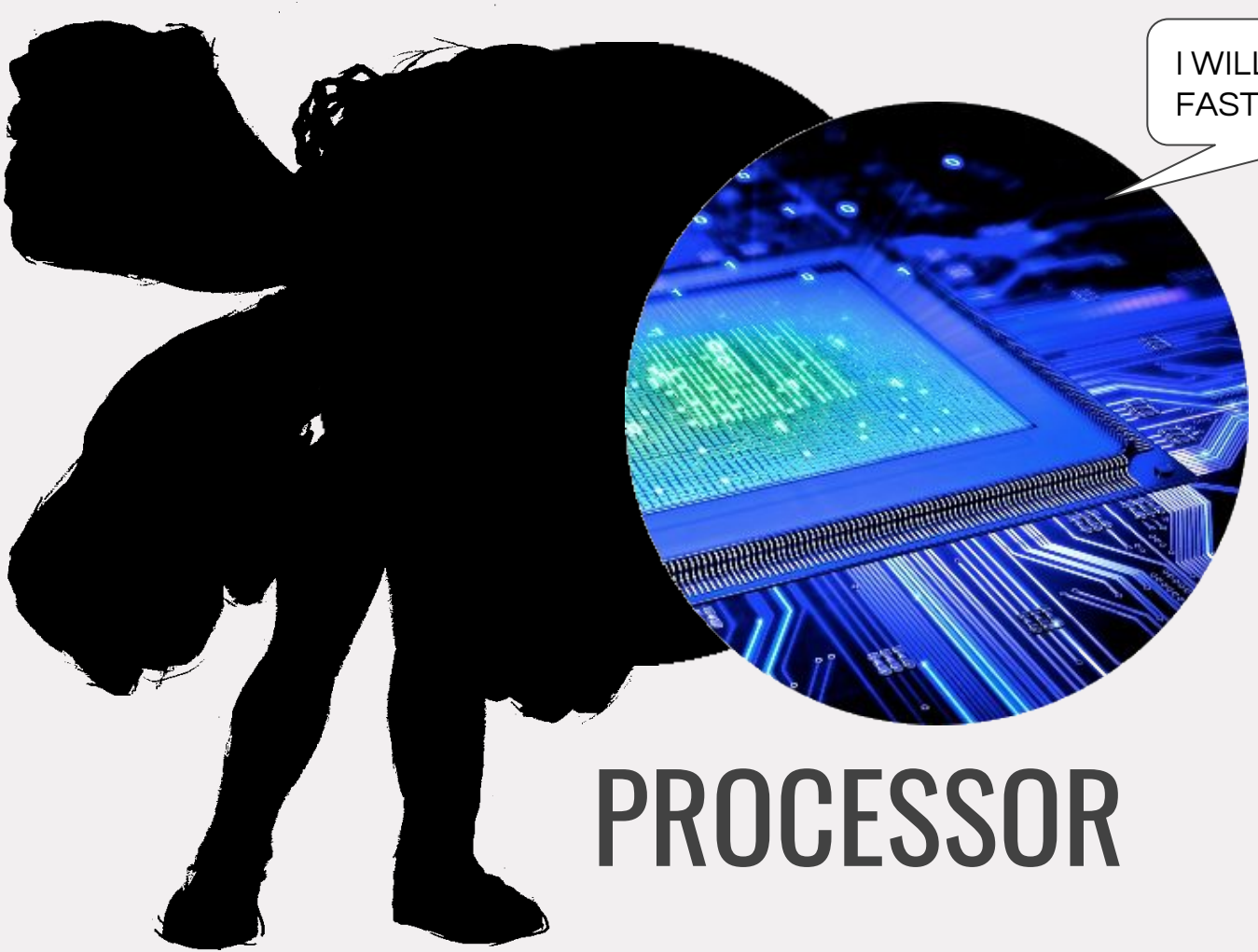

I WILL MAKE THE THING WITH CODE.
AT HOME.
LISTENING TO MY OWN MUSIC.
IN MY PJs.
RN.

PROGRAMMER

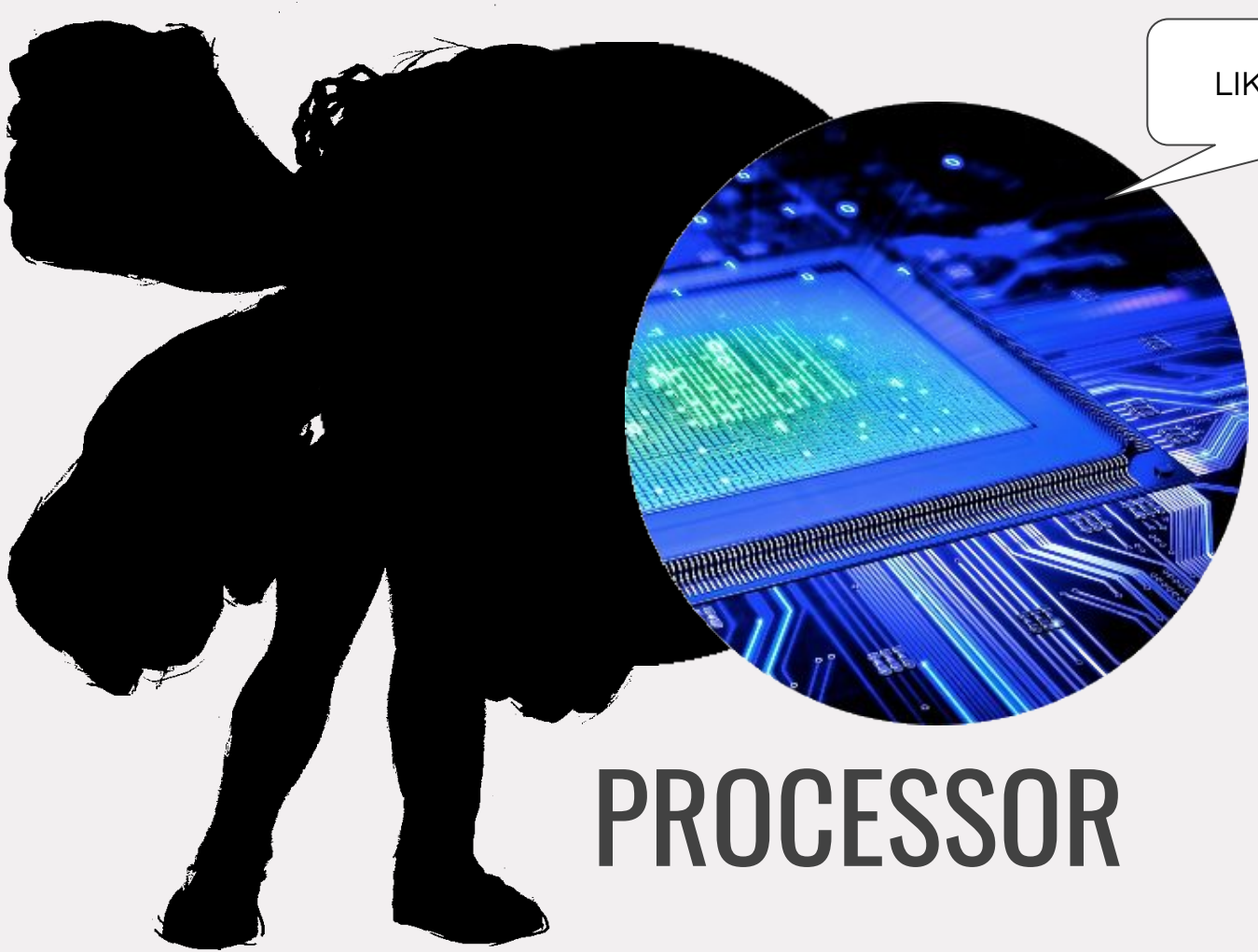
PROGRAMMER


I AM THE THING.
ariables acted upon.
Variables receiving results.
Indication of change in the value on any Variable.
Statement of Resu
Natur
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 − $^1V_4 - {}^1V_1$ $^2V_4$ $\left\{\begin{matrix} {}^1V_4 = {}^2V_4 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\} = 2n-1$
3 + $^1V_5 + {}^1V_1$ $^2V_5$ $\left\{\begin{matrix} {}^1V_5 = {}^2V_5 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\} = 2n+1$
4 ÷ $^2V_5 \div {}^2V_4$ $^1V_{11}$ $\left\{\begin{matrix} {}^2V_5 = {}^0V_5 \\ {}^2V_4 = {}^0V_4 \end{matrix}\right\} = \frac{2n-1}{2n+1}$
5 + $^1V_{11} \div {}^1V_2$ $^2V_{11}$ $\left\{\begin{matrix} {}^1V_{11} = {}^2V_{11} \\ {}^1V_2 = {}^1V_2 \end{matrix}\right\} = \frac{1}{2} \cdot \frac{2n-1}{2n+1}$
− $^0V_{13} - {}^2V_{11}$ $^1V_{13}$ $\left\{\begin{matrix} {}^2V_{11} = {}^0V_{11} \\ {}^0V_{13} = {}^1V_{13} \end{matrix}\right\} = -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$
− $^1V_3 - {}^1V_1$ $^1V_{10}$ $\left\{\begin{matrix} {}^1V_3 = {}^1V_3 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\} = n-1\,(=3)$
$^1V_2 + {}^0V_7$ $^1V_7$ $\left\{\begin{matrix} {}^1V_2 = {}^1V_2 \\ {}^0V_7 = {}^1V_7 \end{matrix}\right\} = 2+0=2$
$+ {}^1V_7$ $^3V_{11}$ $\left\{\begin{matrix} {}^1V_6 = {}^1V_6 \\ {}^0V_{11} = {}^3V_{11} \end{matrix}\right\} = \frac{2n}{2} = A_1$
$^1V_{12}$ $\left\{\begin{matrix} {}^1V_{21} = {}^1V_{21} \\ {}^3V_{11} = {}^3V_{11} \end{matrix}\right\} = B_1 \cdot \frac{2n}{2}$
$\left\{{}^1V_{12} = {}^0V_{12}\right.$

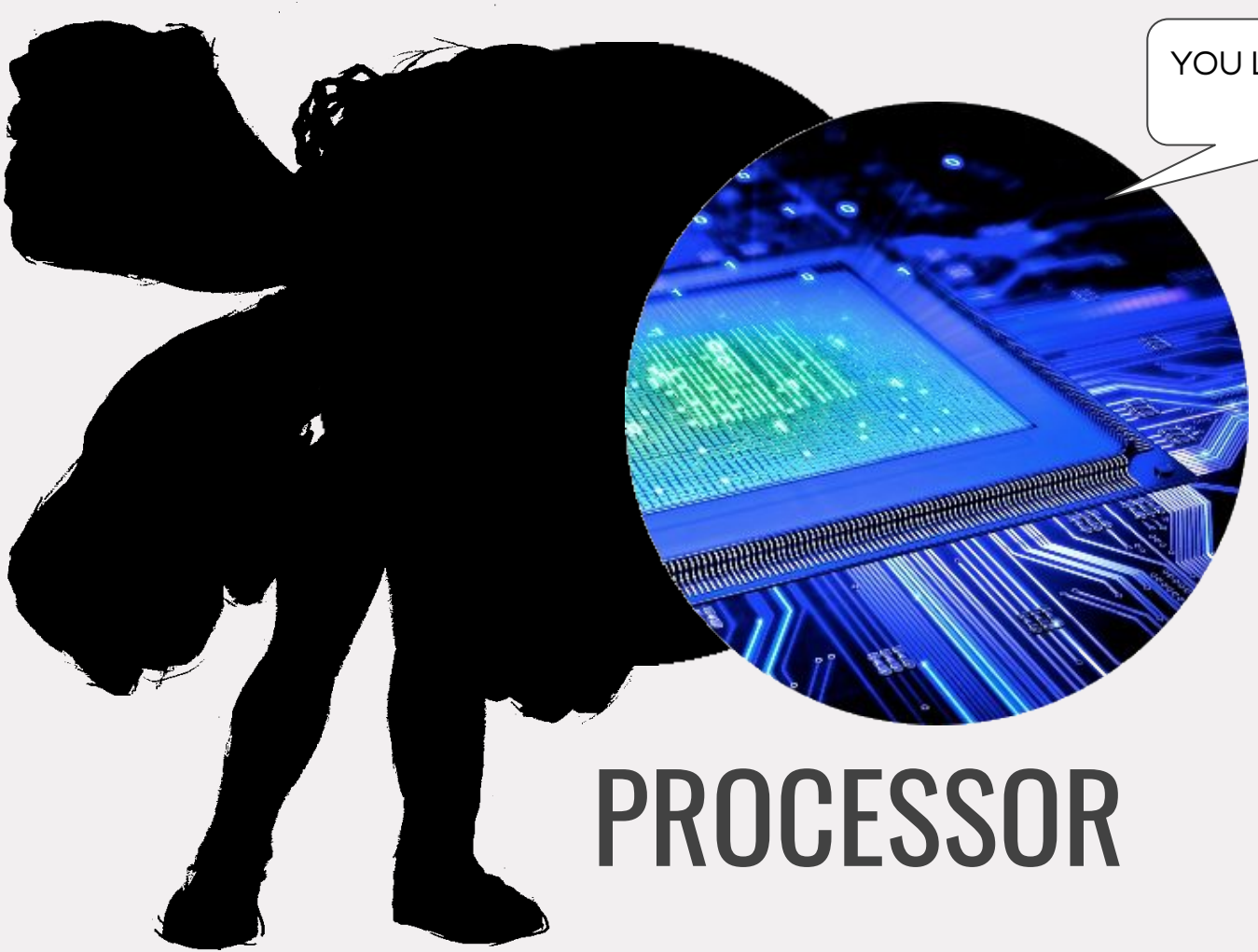
PROGRAM



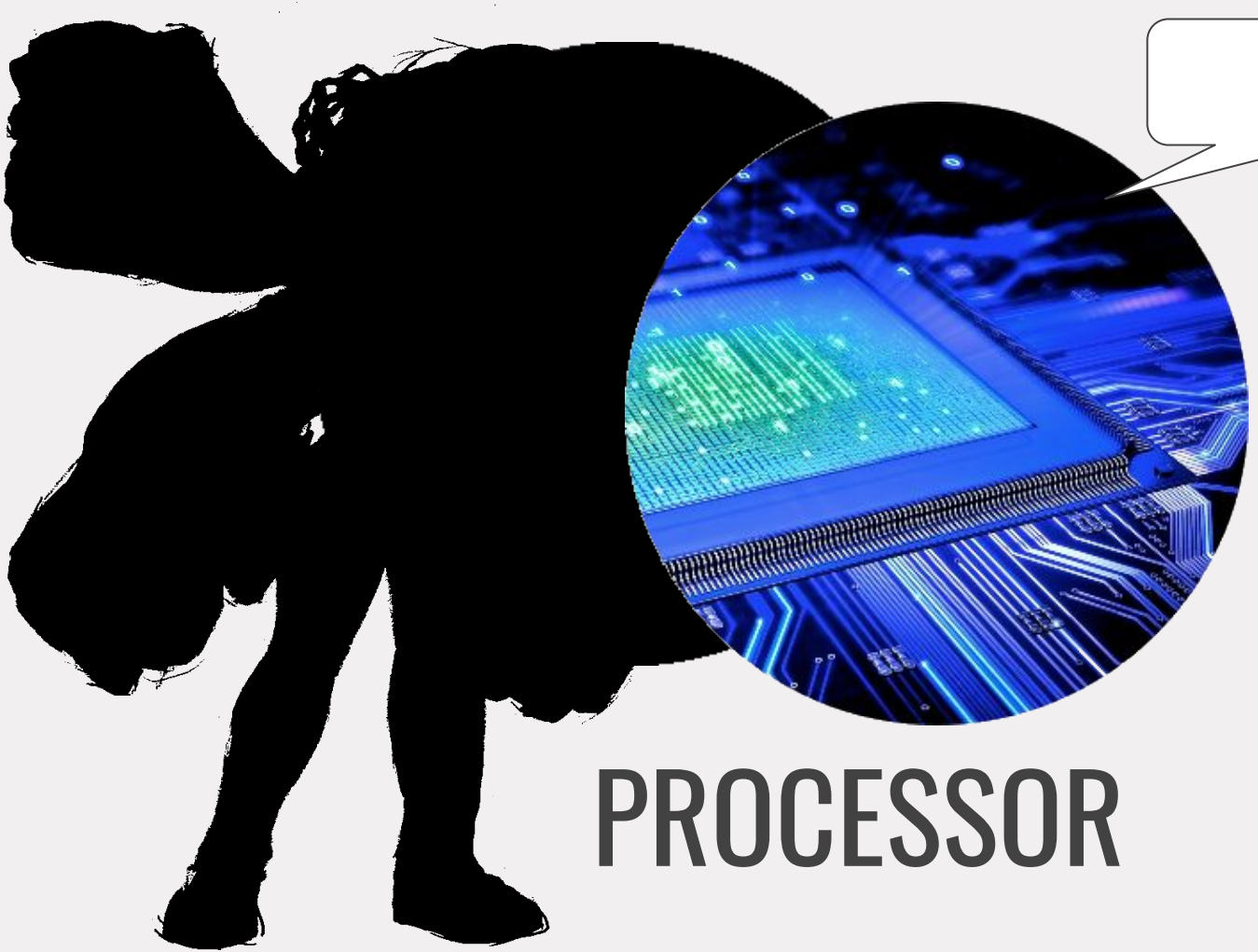
PROCESSOR



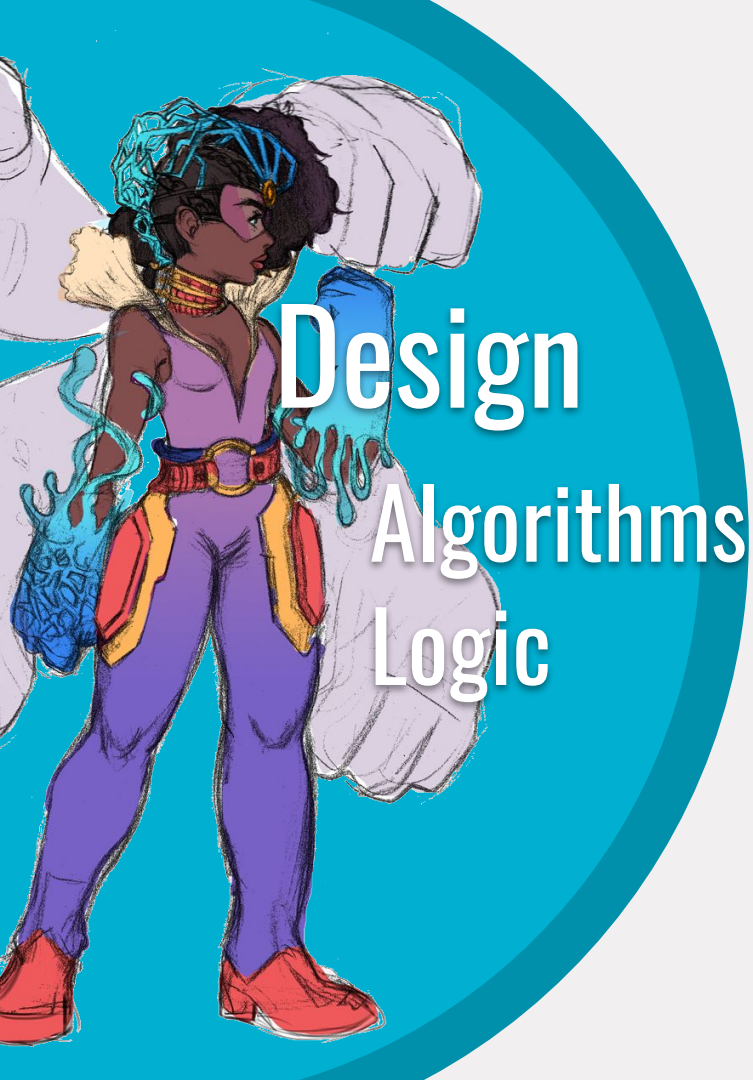
PROCESSOR



PROCESSOR



PROCESSOR

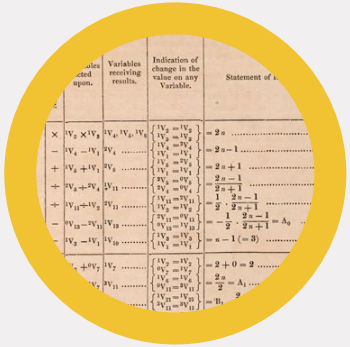


Design

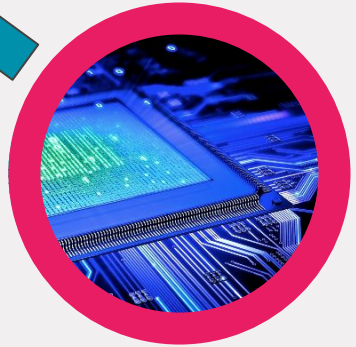
Algorithms

Logic

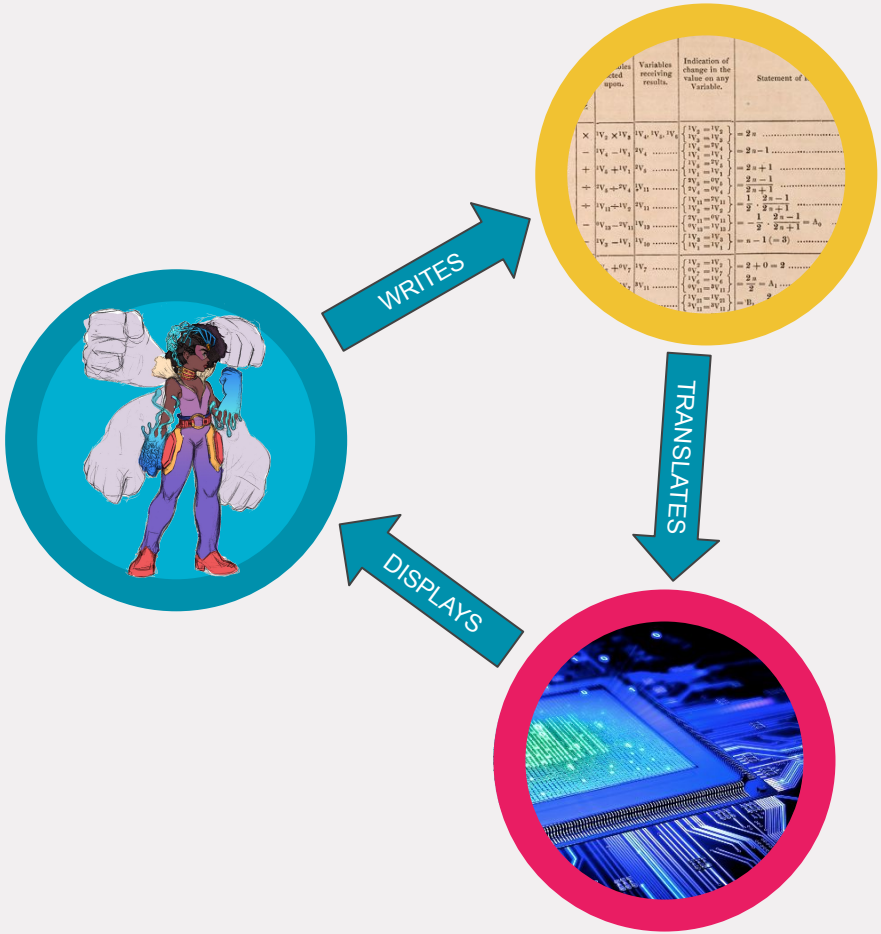
WRITES



TRANSLATES



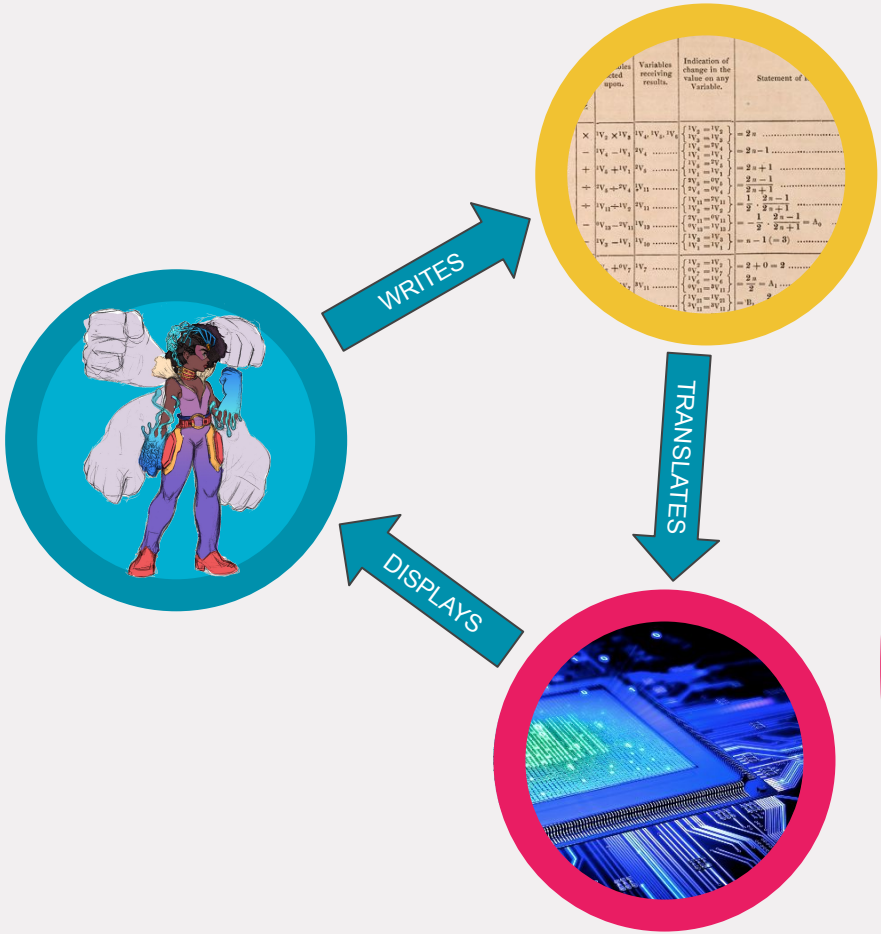
DISPLAYS



Software

Applications

System



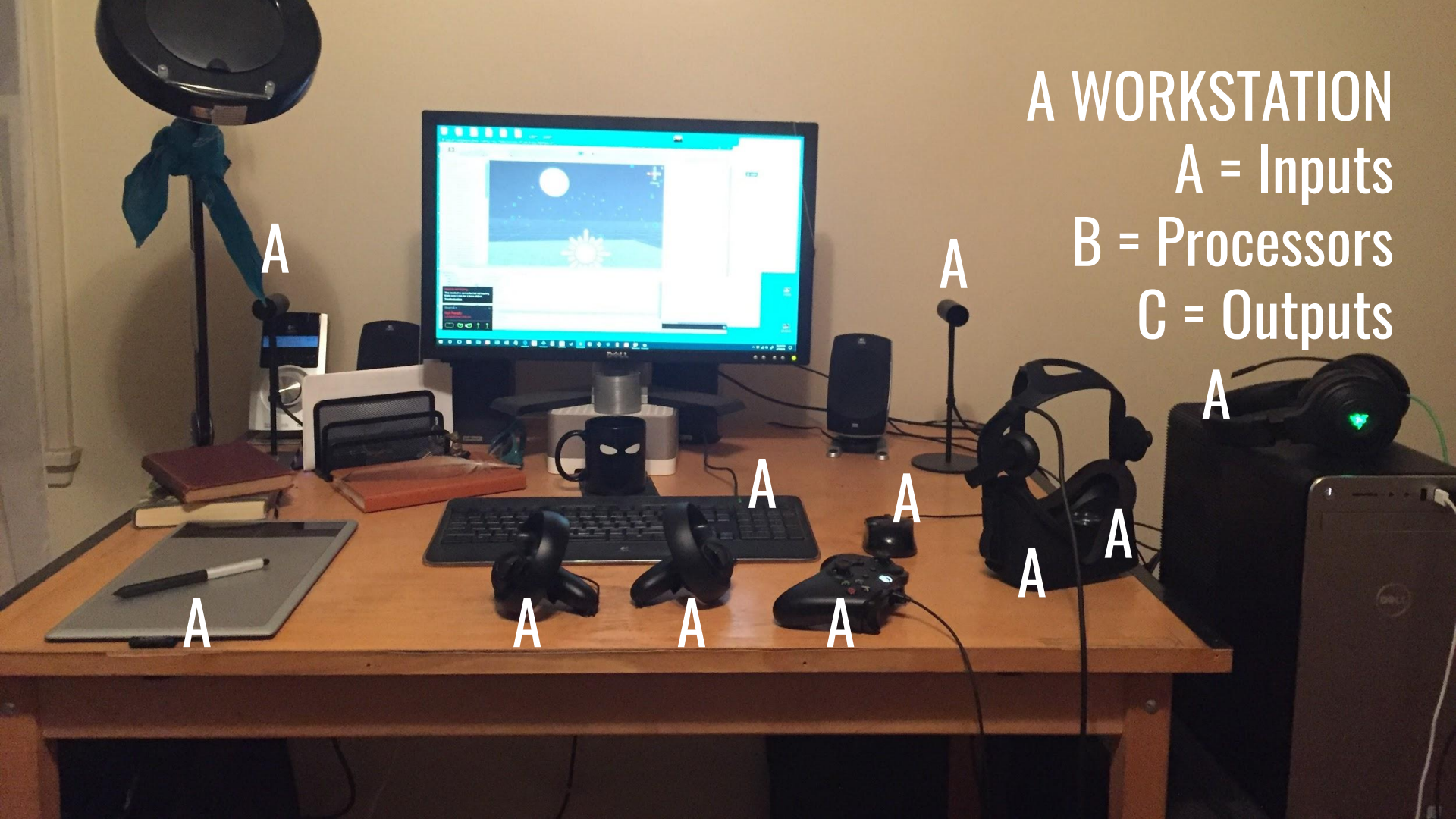
Hardware

Inputs
Storage
Processors
Outputs

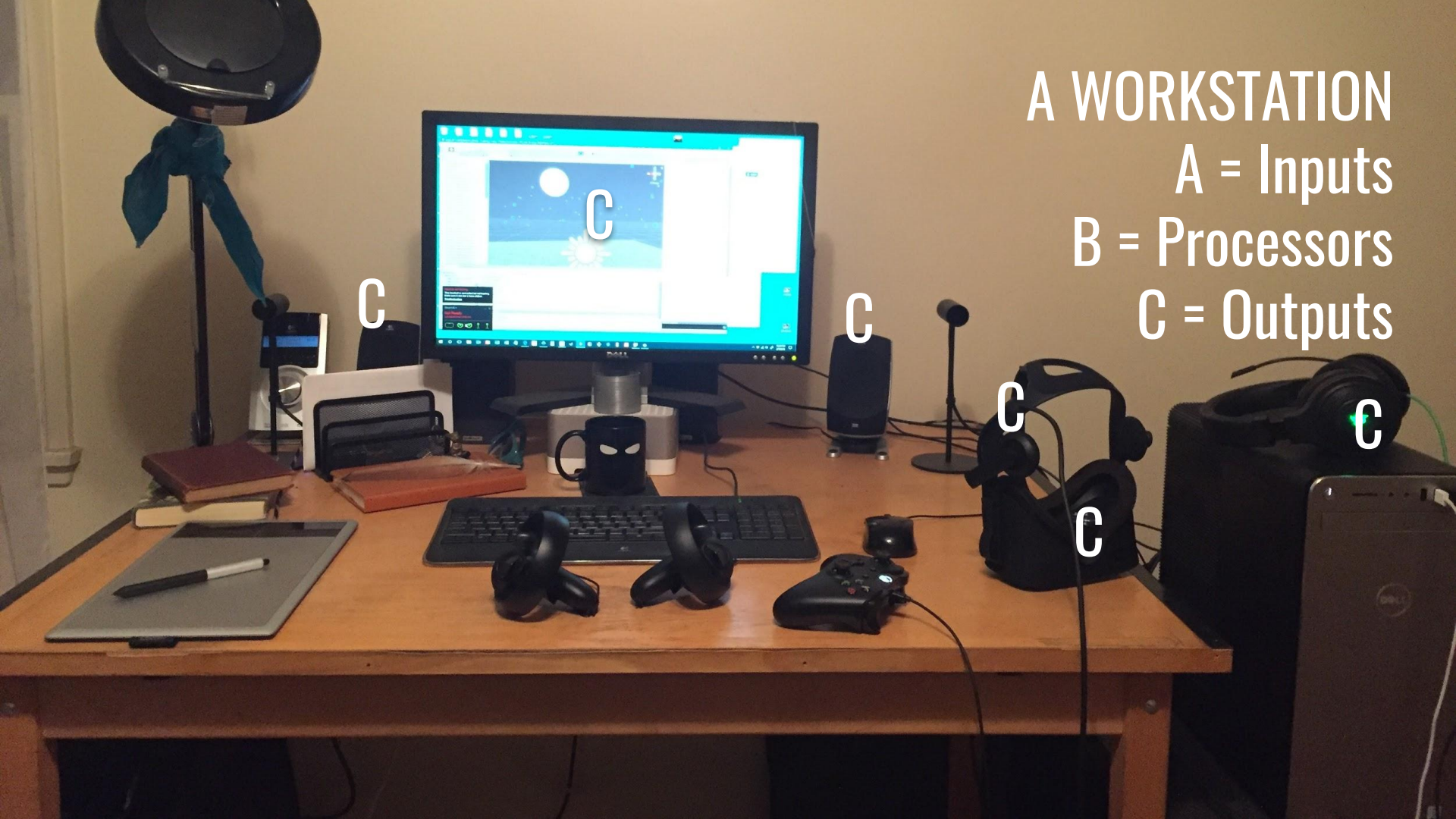

A WORKSTATION

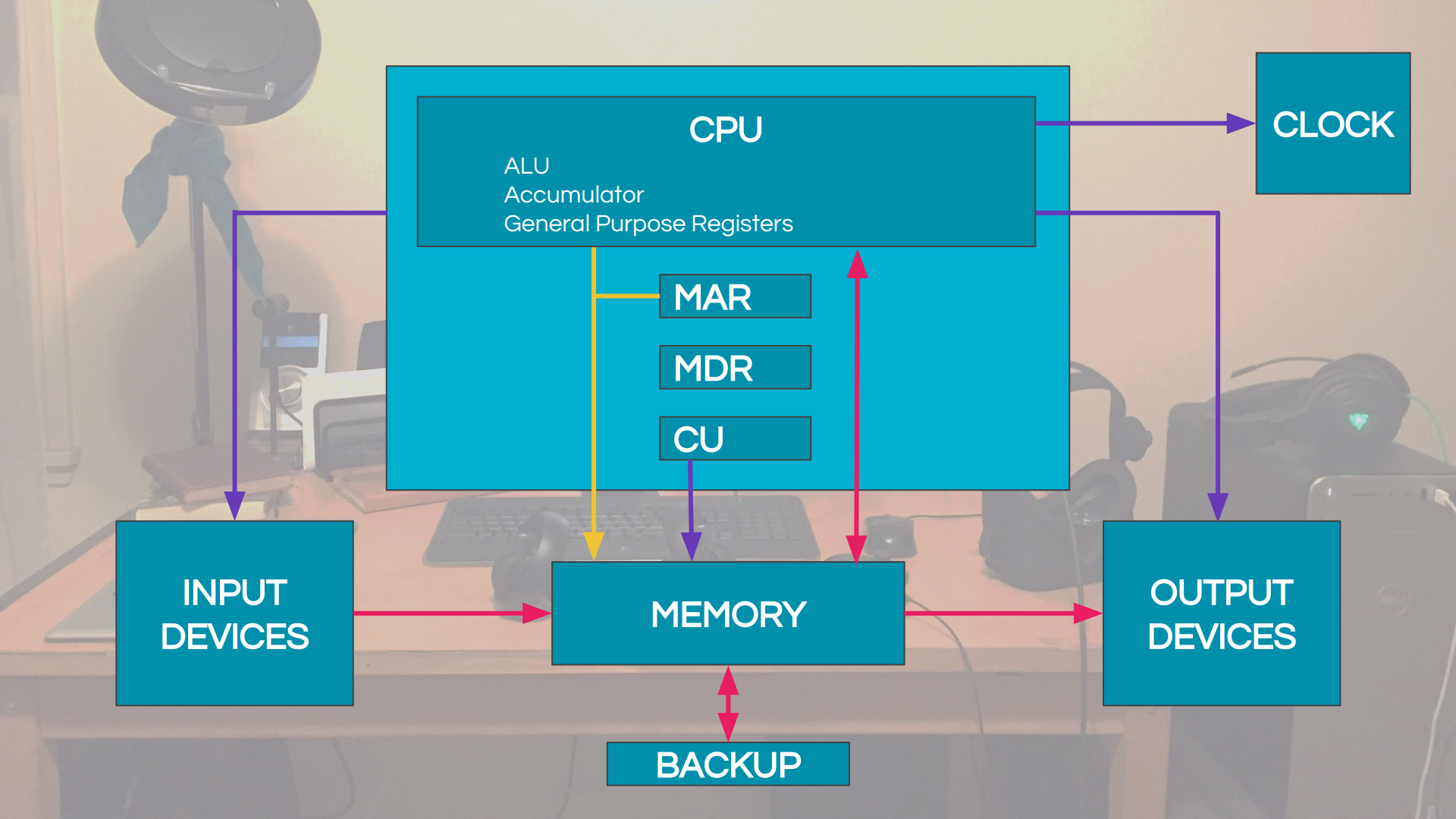

A WORKSTATION
Inputs
Processors
Outputs


A WORKSTATION
A = Inputs
B = Processors
C = Outputs


A WORKSTATION
A = Inputs
B = Processors
C = Outputs
A
A
A
A
A
A
A
A
A
A
A


A WORKSTATION
A = Inputs
B = Processors
C = Outputs
B


A WORKSTATION
A = Inputs
B = Processors
C = Outputs
C
C
C
C
C
C


CPU
ALU
Accumulator
General Purpose Registers
MAR
MDR
CU
CLOCK
INPUT DEVICES
MEMORY
OUTPUT DEVICES
BACKUP

Constraints... Let's Break It Down

- **You** write programs in a **High Level Language**
- the **Compiler** translates that
- into the **Low Level**, **Assembly Language**
- the **Assembler** translates that
- into **Machine Language** (binary!)
- the **Control Unit** interprets that
- for the **Microarchitecture**
- where the **Microsequencer** interprets the binary
- for the **Logic-Design** at the **Device Level**
- made up of **Semiconductors / Silicon Transistors**

...and **CONSTRAINED** by the properties of **Atoms**, **Electrons**, and **Quantum Dynamics!**

Compiler

Assembler

Control Unit

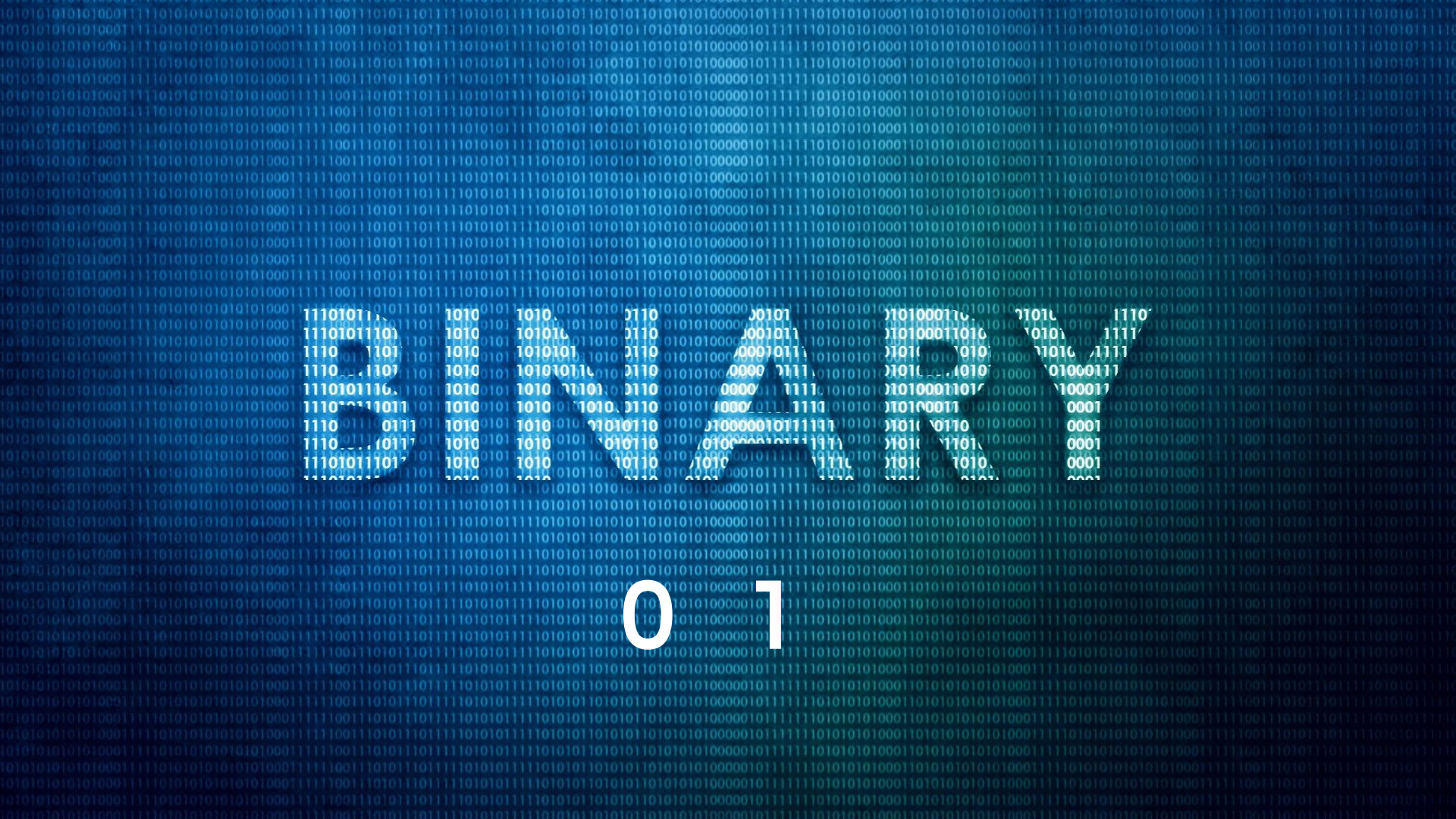
Microsequencer

Frosting

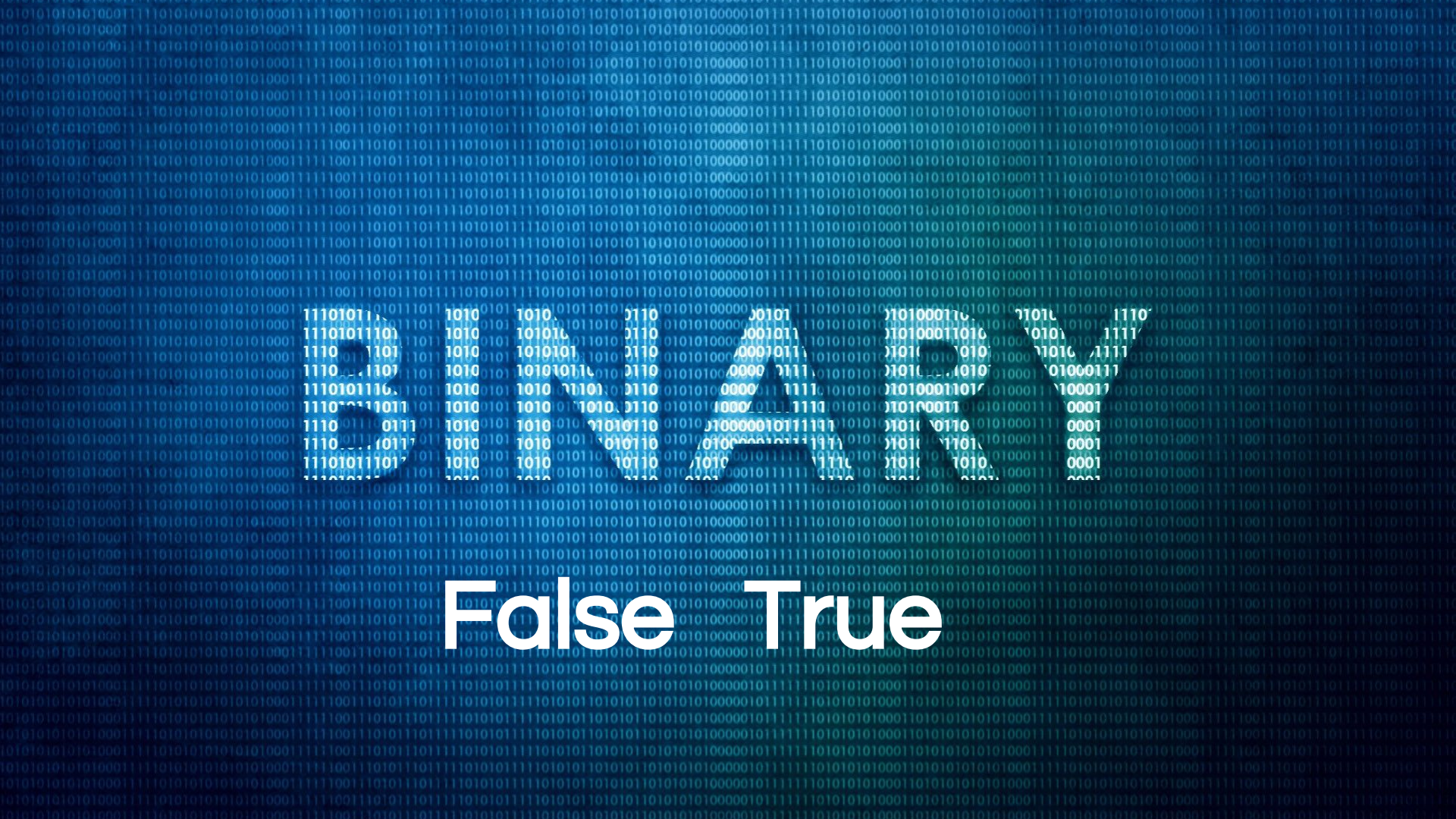
More frosting

What is

?


01


Closed Open


DAY
False True


Off On

A computer is:

A binary digital machine

- Basic unit of information is the binary digit, or bit

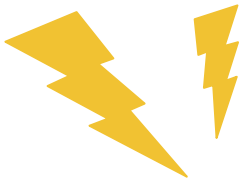
- Values with more than two states require multiple bits.
  - Two bits have four possible states: 00, 01, 10, 11
  - Three bits have eight possible states: 000, 001, 010, 011, 100, 101, 110, 111
  - A collection of n bits has 2^n possible states.

What do 0 or 1 mean?

- The computer is an electronic machine
- Bits (which can have a value of 0 or 1) are…

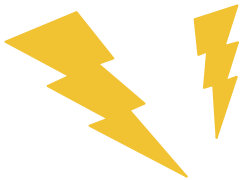
What do 0 or 1 mean?

- The computer is an electronic machine
- Bits (which can have a value of 0 or 1) are...

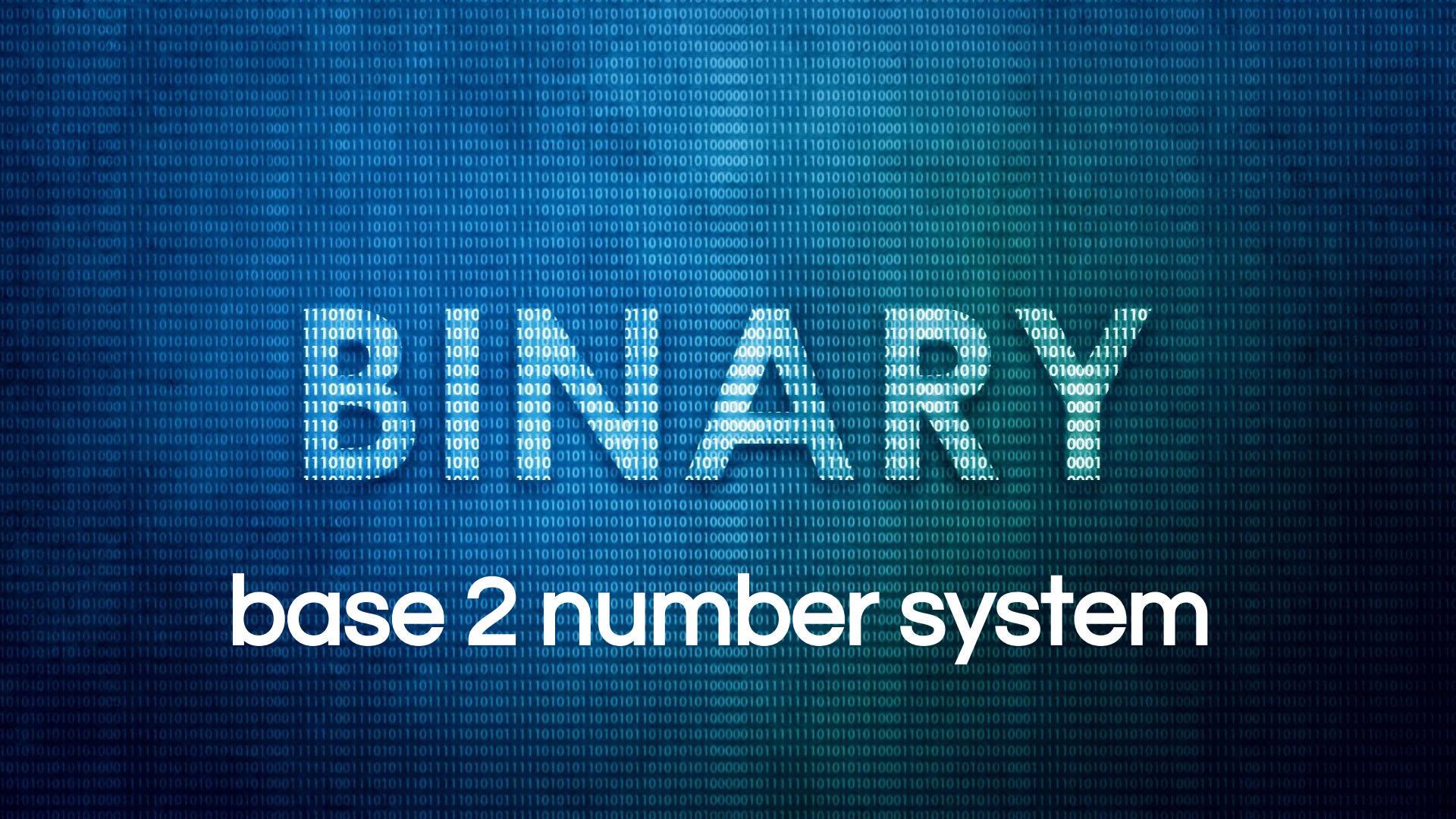


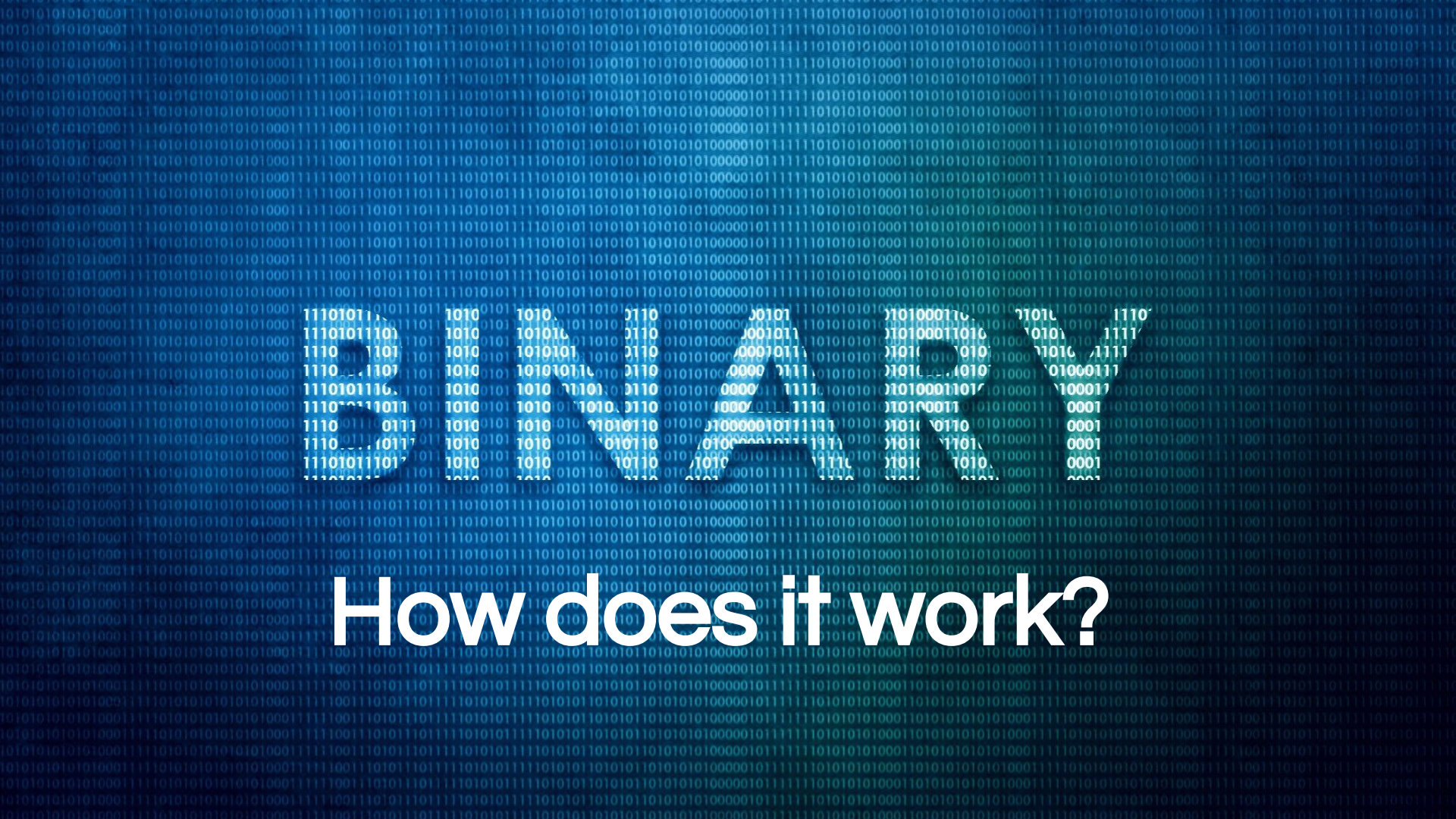
What do 0 or 1 mean?

- The computer is an electronic machine
- Bits (which can have a value of 0 or 1) are…



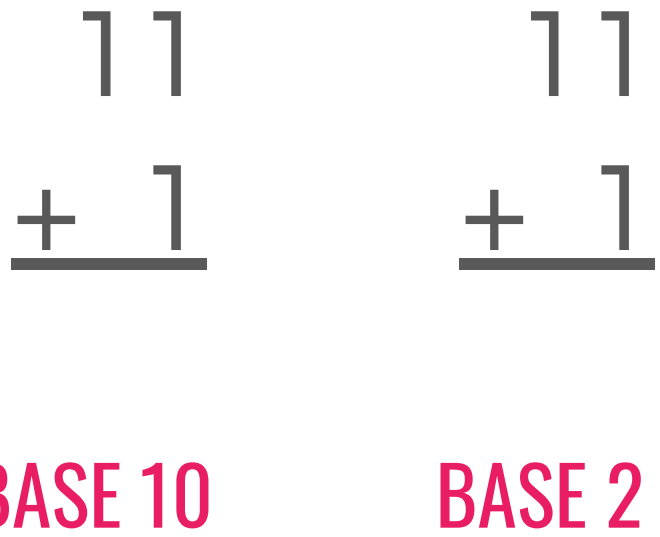
- Voltage in one of two states:
  - **presence** of voltage – we call this state **“1”**
  - **absence** of voltage – we call this state **“0”**


base 2 number system


How does it work?

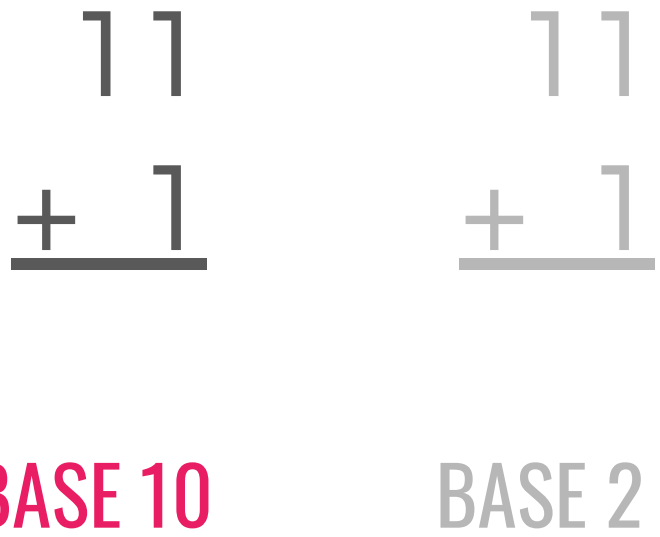
BINARY ARITHMETIC

- Base-2 Addition
  - just like base-10!
  - add from right to left, carrying



BINARY ARITHMETIC

- Base-2 Addition
  - just like base-10!
  - add from right to left, carrying



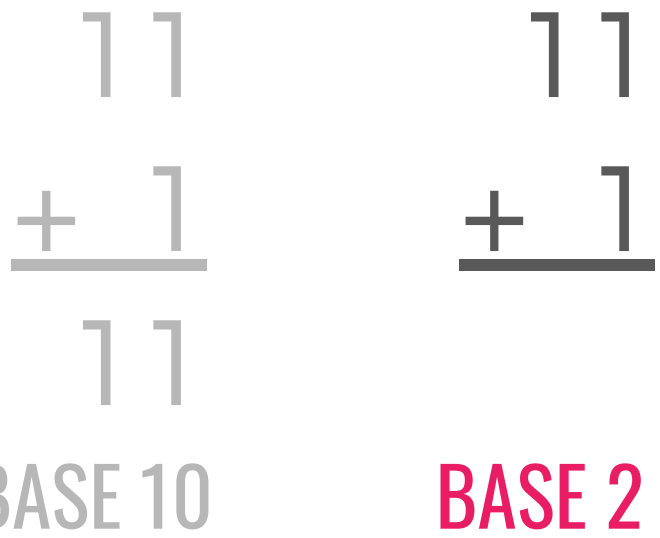
BINARY ARITHMETIC

- Base-2 Addition
  - just like base-10!
  - add from right to left, carrying

BASE 10	BASE 2
10	10
+ 1	+ 1
11	

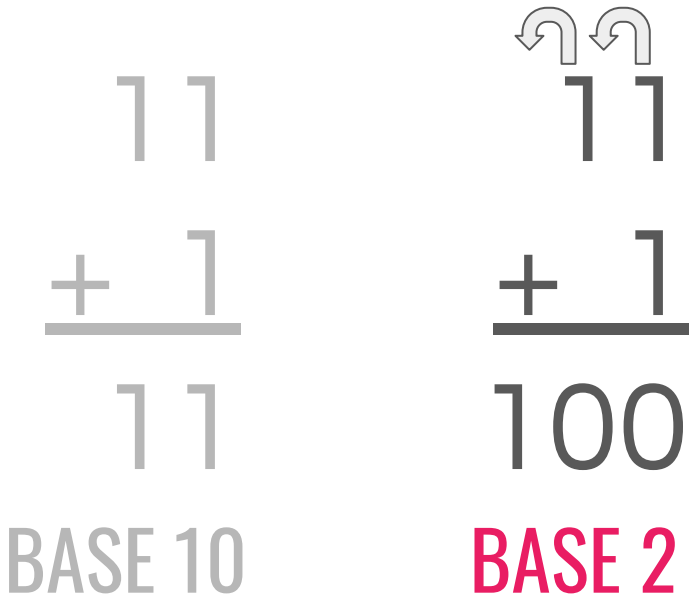
BINARY ARITHMETIC

- Base-2 Addition
  - just like base-10!
  - add from right to left, carrying



BINARY ARITHMETIC

- Base-2 Addition
  - just like base-10!
  - add from right to left, carrying



BINARY ARITHMETIC

10010	10010	1111
+ 1001	+ 1011	+ 1

BINARY ARITHMETIC

$$\begin{array}{r} 10010 \\ +\ 1001 \\ \hline \end{array} \qquad \begin{array}{r} 10010 \\ +\ 1011 \\ \hline \end{array} \qquad \begin{array}{r} 1111 \\ +\ \ \ 1 \\ \hline \end{array}$$

BINARY ARITHMETIC

$$\begin{array}{r} 10010 \\ +\ 1001 \\ \hline 11011 \end{array} \qquad \begin{array}{r} 10010 \\ +\ 1011 \\ \hline \end{array} \qquad \begin{array}{r} 1111 \\ +\ \ \ 1 \\ \hline \end{array}$$

BINARY ARITHMETIC

$$\begin{array}{r} 10010 \\ +\ 1001 \\ \hline 11011 \end{array} \qquad \begin{array}{r} 10010 \\ +\ 1011 \\ \hline \end{array} \qquad \begin{array}{r} 1111 \\ +\ \quad 1 \\ \hline \end{array}$$

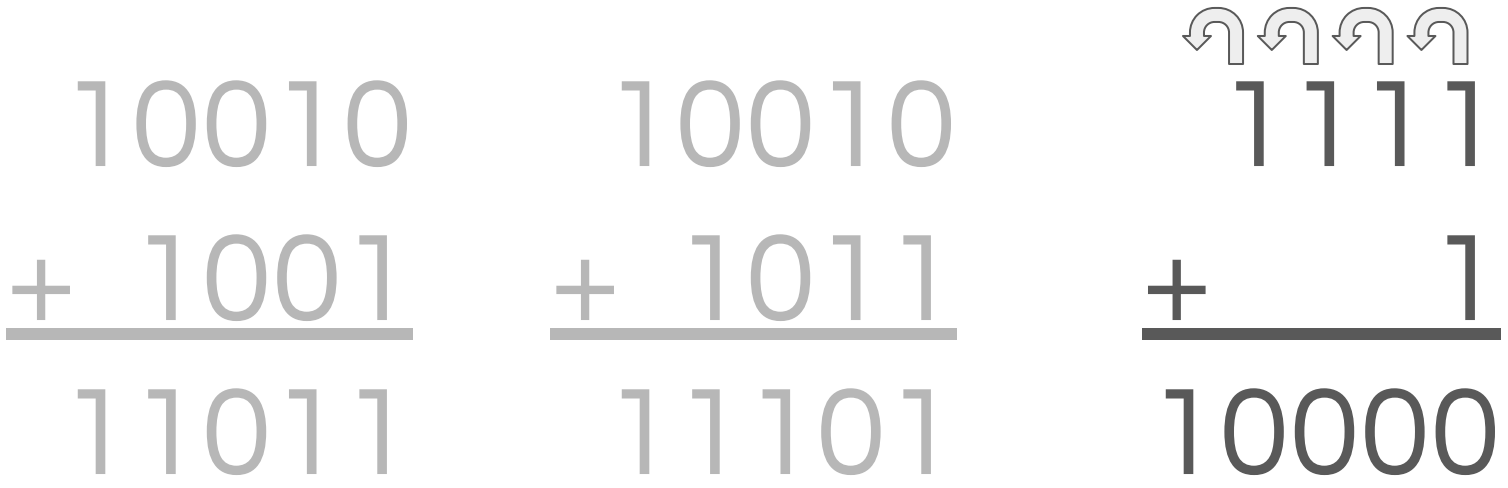
BINARY ARITHMETIC

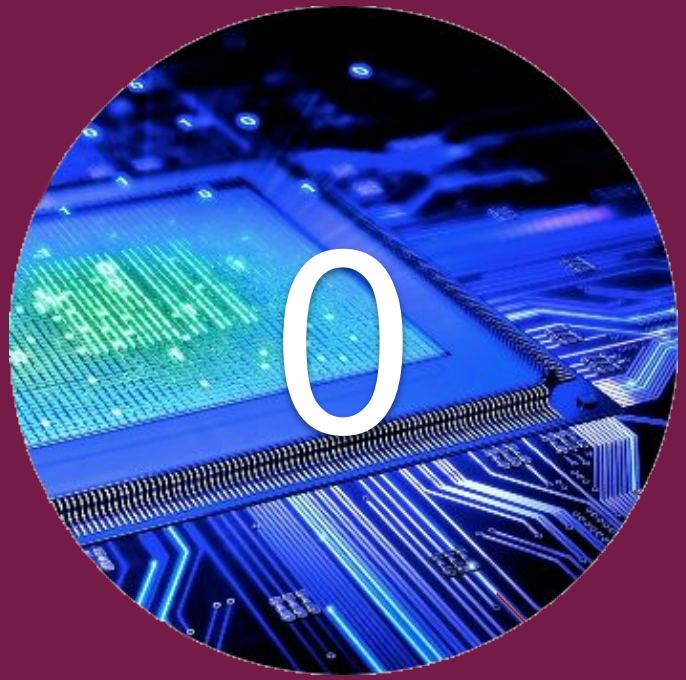
```
  10010      10010      1111
+ 1001     + 1011     +    1
-------    -------    ------
  11011      11101
```

BINARY ARITHMETIC

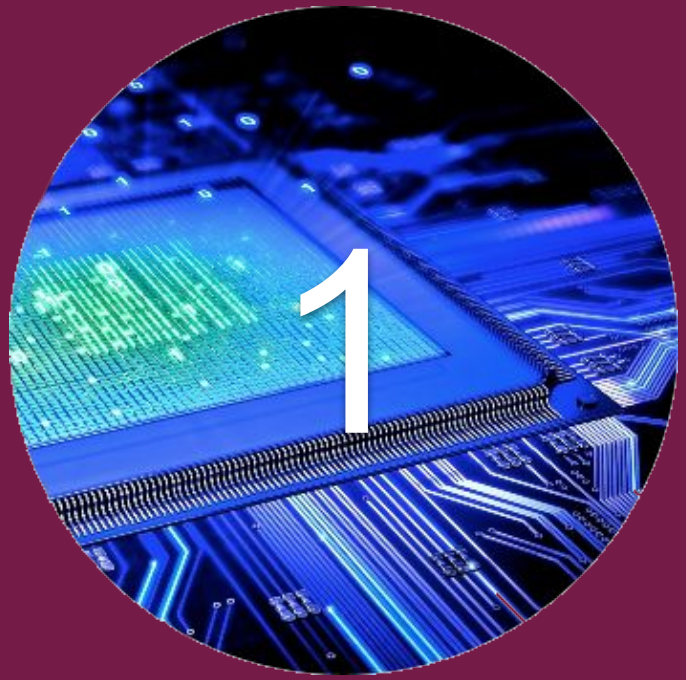
$$\begin{array}{r} 10010 \\ +\ 1001 \\ \hline 11011 \end{array} \qquad \begin{array}{r} 10010 \\ +\ 1011 \\ \hline 11101 \end{array} \qquad \begin{array}{r} 1111 \\ +\ \quad 1 \\ \hline \end{array}$$

BINARY ARITHMETIC





A BIT



A BIT

A BYTE

8 BITS

A BYTE

How many variations are possible?

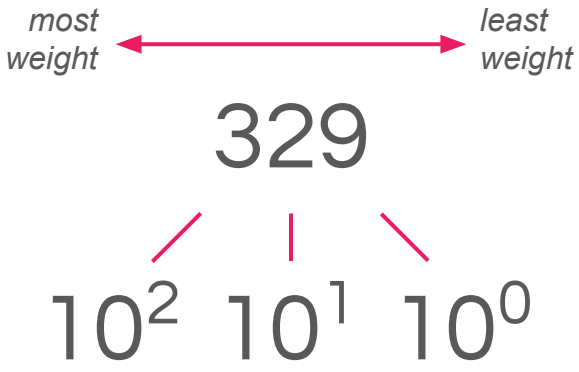
32-bit OS

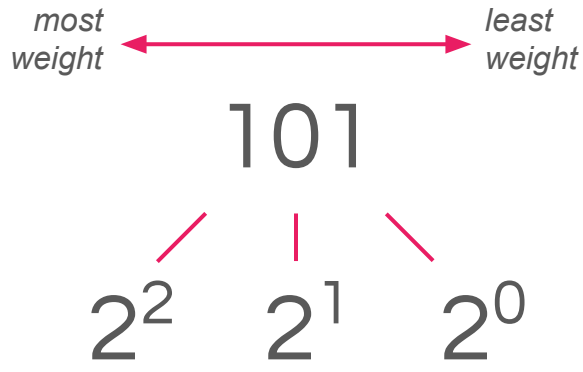
64-bit OS

BINARY NUMBERS 0 to 7

2^2	2^1	2^0	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

WEIGHTED POSITIONAL NOTATION





Decimal

- "329" the "3" is worth 300 because of its position
- What is the "9" worth?

Binary

- "101" the first "1" is worth 4 because of its position
- What is the second "1" worth?

HEXADECIMAL NOTATION

- Also "base 16" or "hex"
- Four bits can represent 16 positions
- $2^4 = 2 \times 2 \times 2 \times 2$

Colors are sometimes represented by hex

#000000 (black) and #FFFFFF (white)

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

HEXADECIMAL NOTATION

#663333

#009999

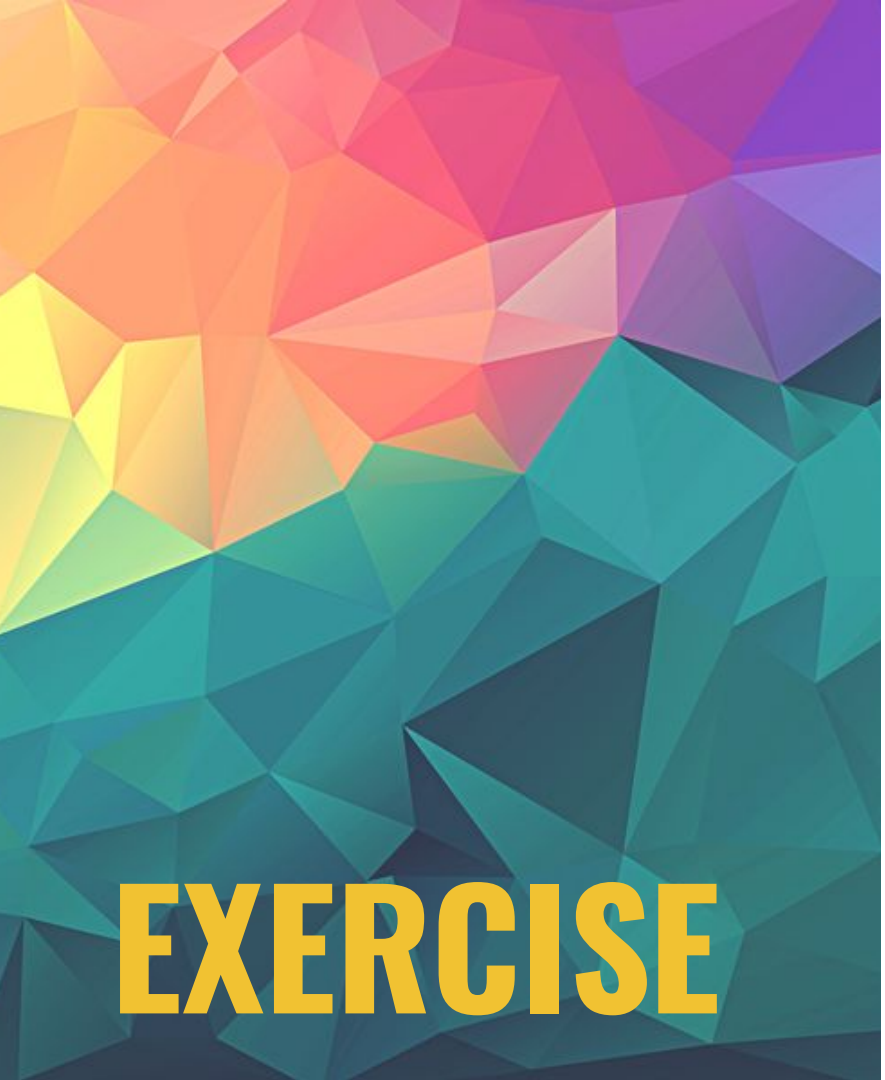
#FFCC33

- Also "base 16" or "hex"
- Four bits can represent 16 positions
- $2^4 = 2 \times 2 \times 2 \times 2$

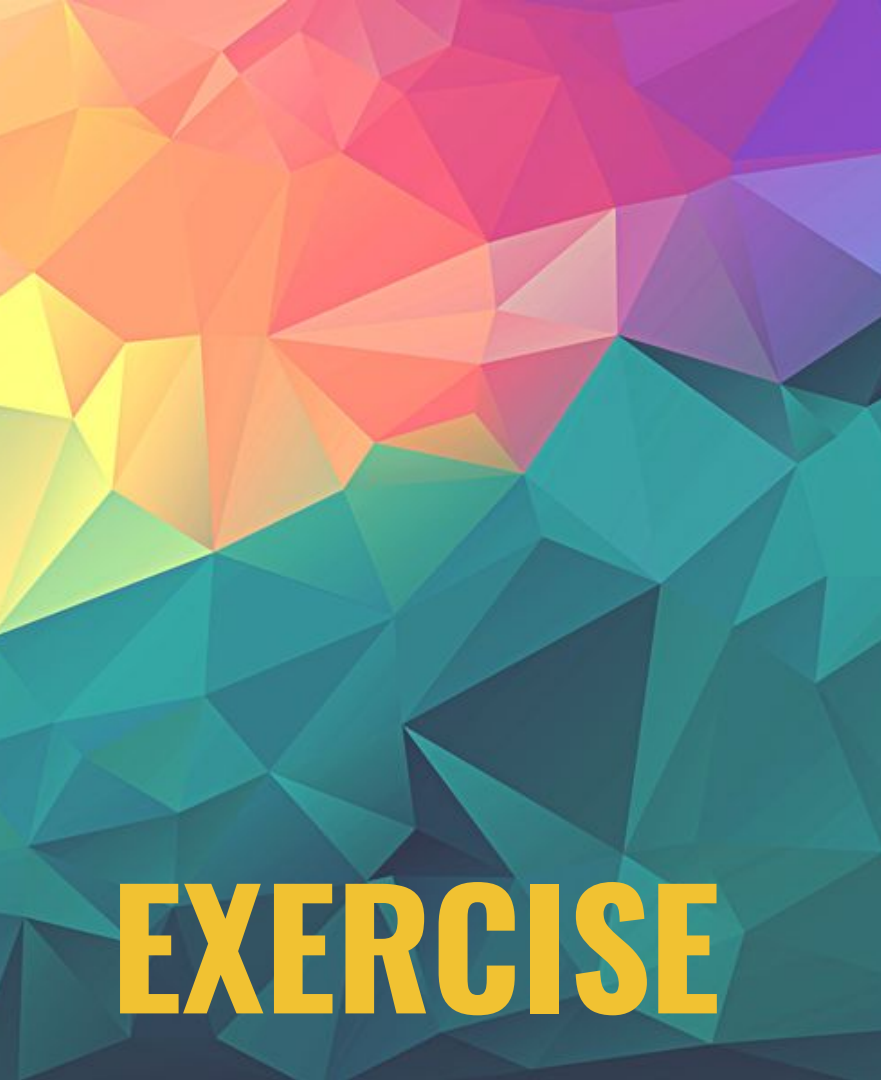
Colors are sometimes represented by hex

#000000 (black) and #FFFFFF (white)

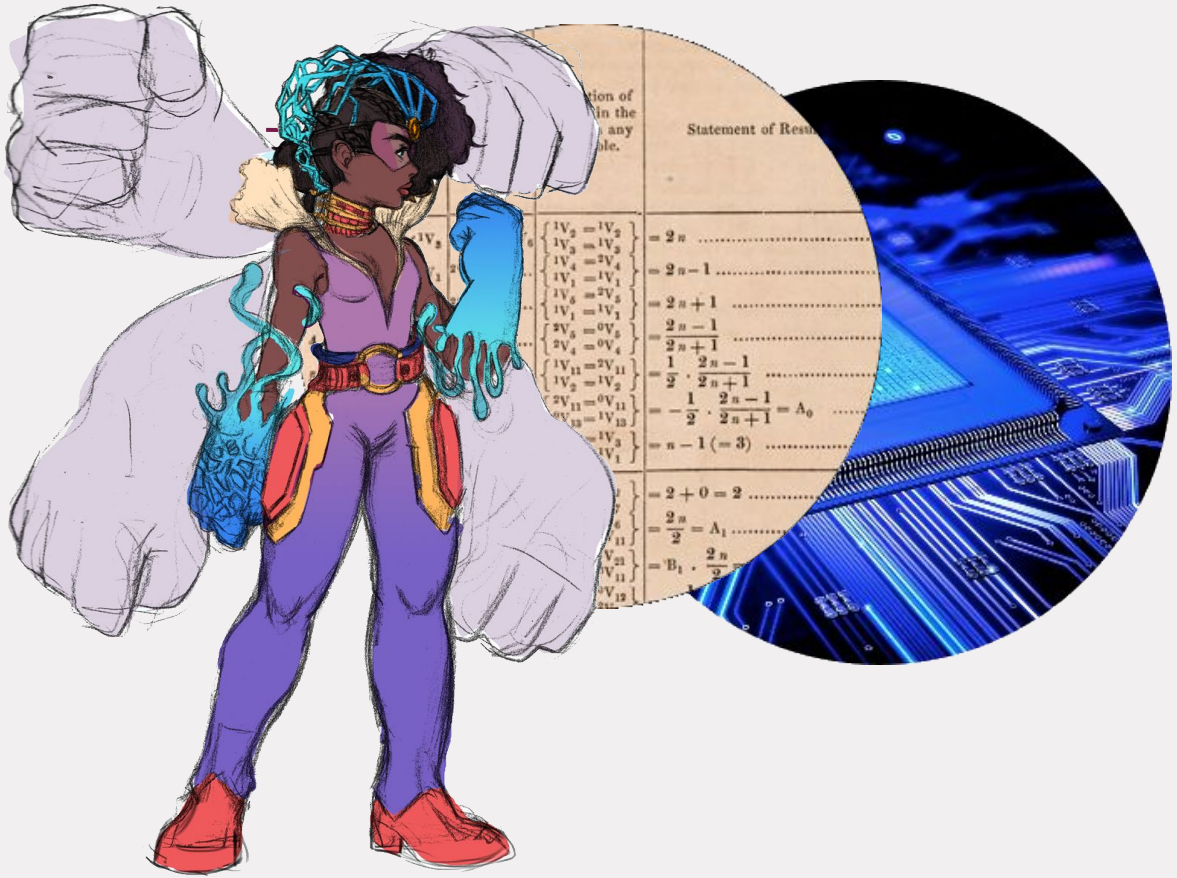
Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15



$$\begin{array}{r} 10111 \\ +\ \ 111 \\ \hline ? \end{array}$$



$$\begin{array}{r} 10111 \\ +\ 00111 \\ \hline ? \end{array}$$



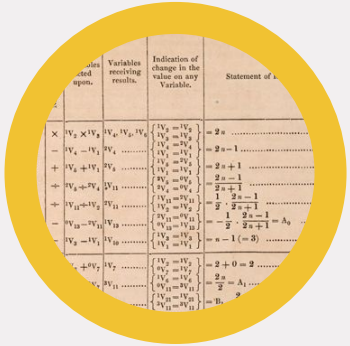
RELATIONSHIP

- Have idea/problem
- Design a solution
- Write code
  - Debug code
- Solve problem
- Save world - have fun - make $$

Programmer

Writes Code

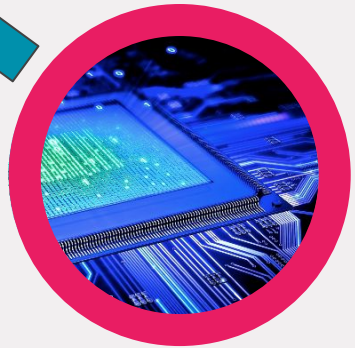
WRITES



Program

Compiles Code into Binary

TRANSLATES



Processor

Processes and displays results

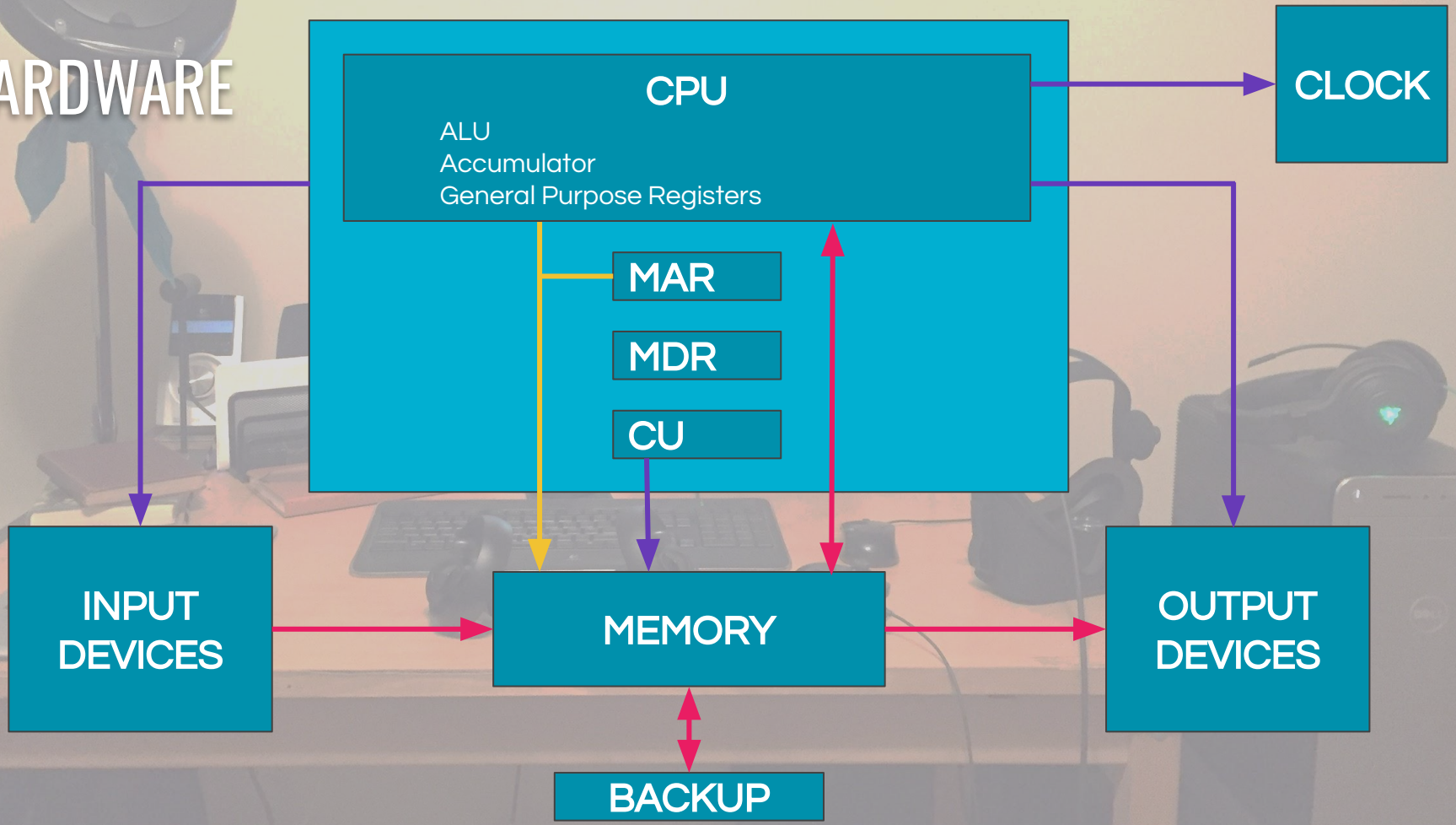
DISPLAYS

HARDWARE

- Inputs
- Storage
- Processor
- Outputs

- Why? → Binary

HARDWARE


CPU
ALU
Accumulator
General Purpose Registers
CLOCK
MAR
MDR
CU
INPUT DEVICES
MEMORY
OUTPUT DEVICES
BACKUP

NEXT UP

Programming Languages