>>> INTRODUÇÃO À COMPUTAÇÃO:
>>> 2 bits ou um algoritmo misterioso?

Name: Moacir Antonelli Ponti[†]

Date: Fevereiro de 2025

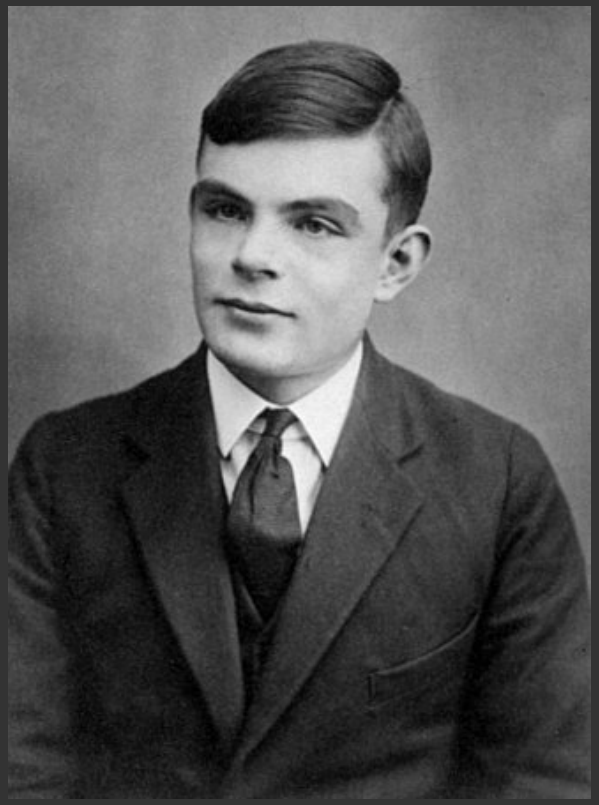
[†]moacir@icmc.usp.br

>>> ‘‘Computação’’?

* Computadores?
* Internet?
* Aplicativos?
* Jogos?
* Redes sociais?

>>> Porque os computadores são como são?

Kurt Gödel



Alan Turing

Escreveu (originalmente em alemão) em 1931:

Para toda classe recursiva ω-consistente κ de fórmulas, há uma correspondência com sinais de classe recursiva ρ, tais que nem ν Gen ρ, nem Neg (v Gen ρ) pertence a Flg (κ) (onde ν é a variável livre de ρ).

Conceito de incompletude

Agora em Português:

se considerarmos um conjunto de regras básicas para estudar números (como a teoria dos números), sempre haverá afirmaçes sobre esses números que você não consegue provar nem refutar usando essas regras.

>>> Turing

Desenvolveu a Máquina de Turing para responder: ‘‘é possível codificar um programa que verifique se um outro programa termina?’’, publicando em 1938 seus resultados sobre o ‘‘problema da parada’’

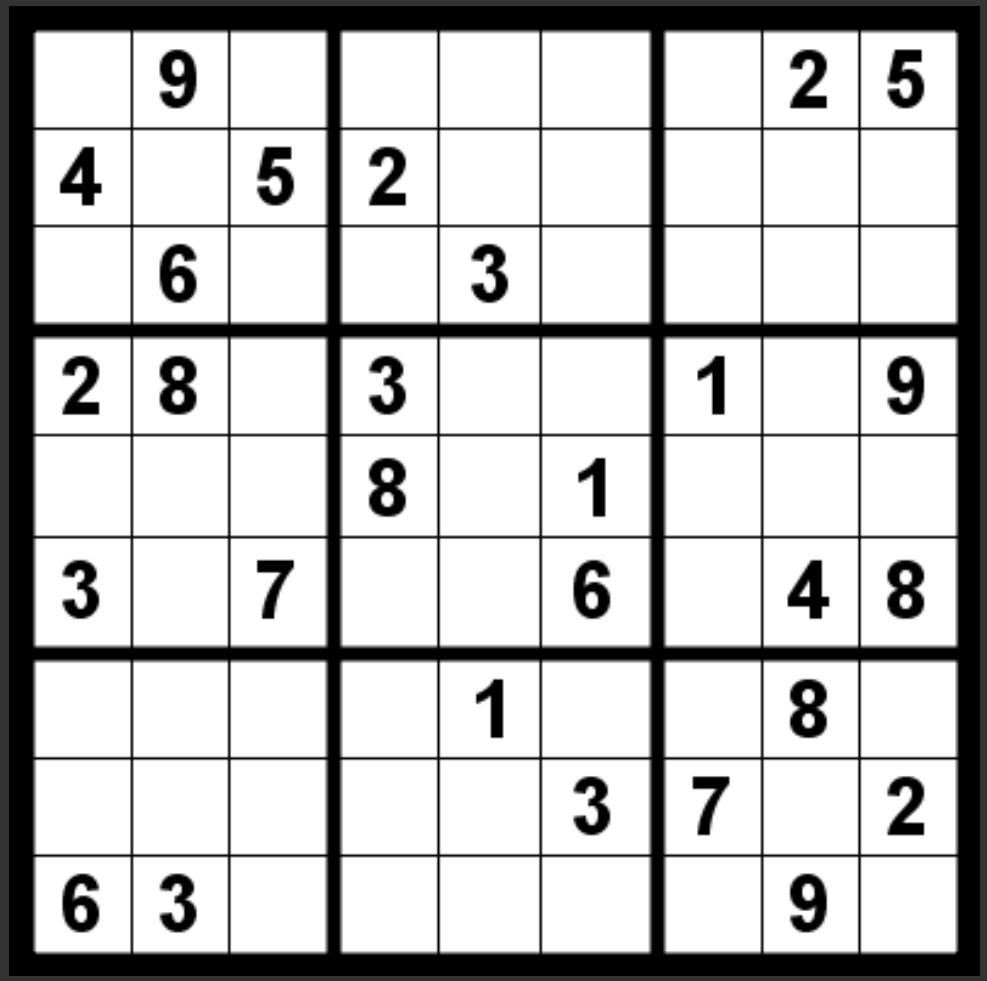
Conceito de indecidibilidade

>>> Problemas computáveis e sua complexidade

Ordenar um conjunto finito de números:

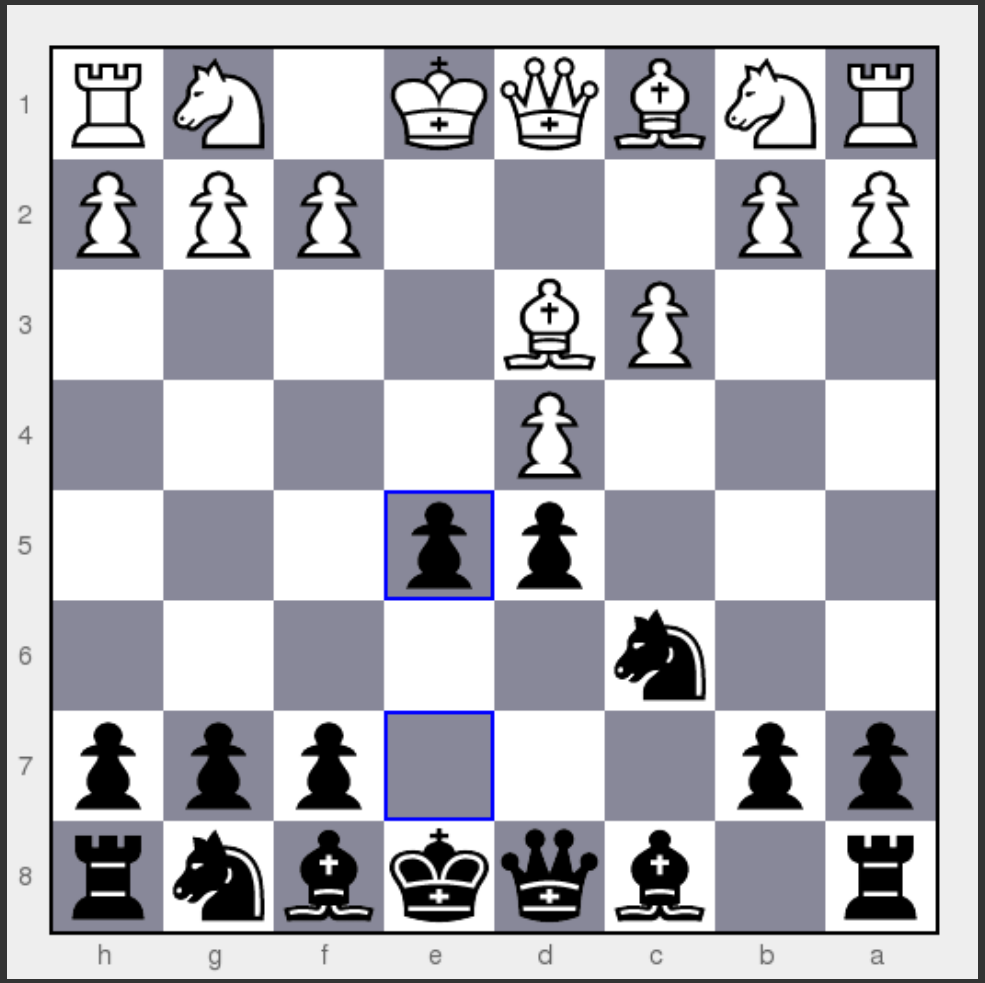
$$4, 3, 2, 5$$

>>> Problemas computáveis e sua complexidade



	9						2	5
4		5	2					
	6			3				
2	8		3			1		9
			8		1			
3		7			6		4	8
				1			8	
					3	7		2
6	3						9	

>>> Problemas computáveis e sua complexidade



>>> Problemas computáveis e sua complexidade

* Ordenar números: solução e verificação viáveis
* Sudoku: apenas verificação viável
* Xadrez: N.D.A.

>>> ‘‘Computação’’?

1. Codificar informação $\rightarrow$ código
2. Processar esse código

>>> Representando a informação.

A menor quantidade de informação que podemos codificar?

* Uma cor? Há pelo menos 100 mil perceptíveis
* Uma letra do alfabeto? 26 letras
* Um número? 10 dígitos

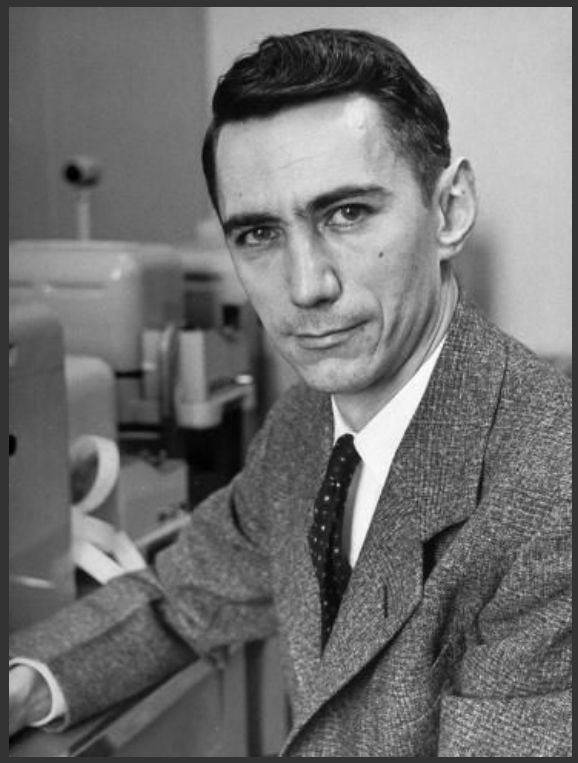
1

0

Ligado

Desligado

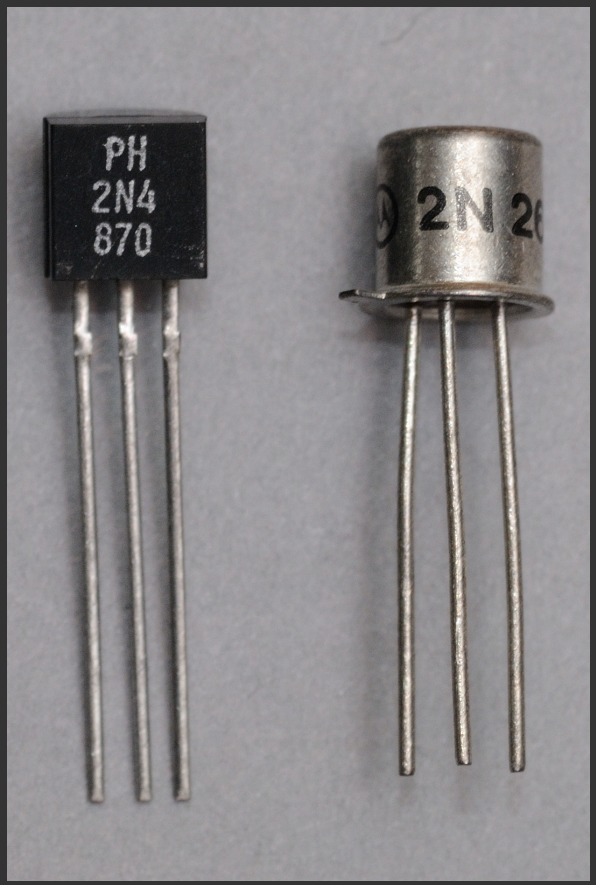
>>> Porque os computadores são como são (2) ?

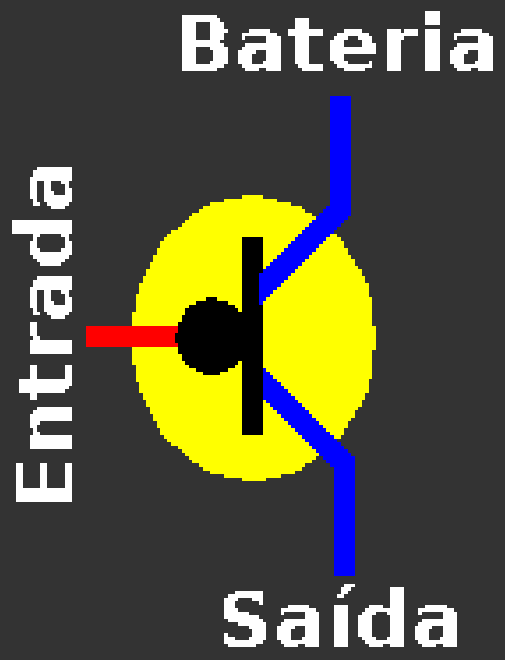


Claude Shannon

John Von Neumann

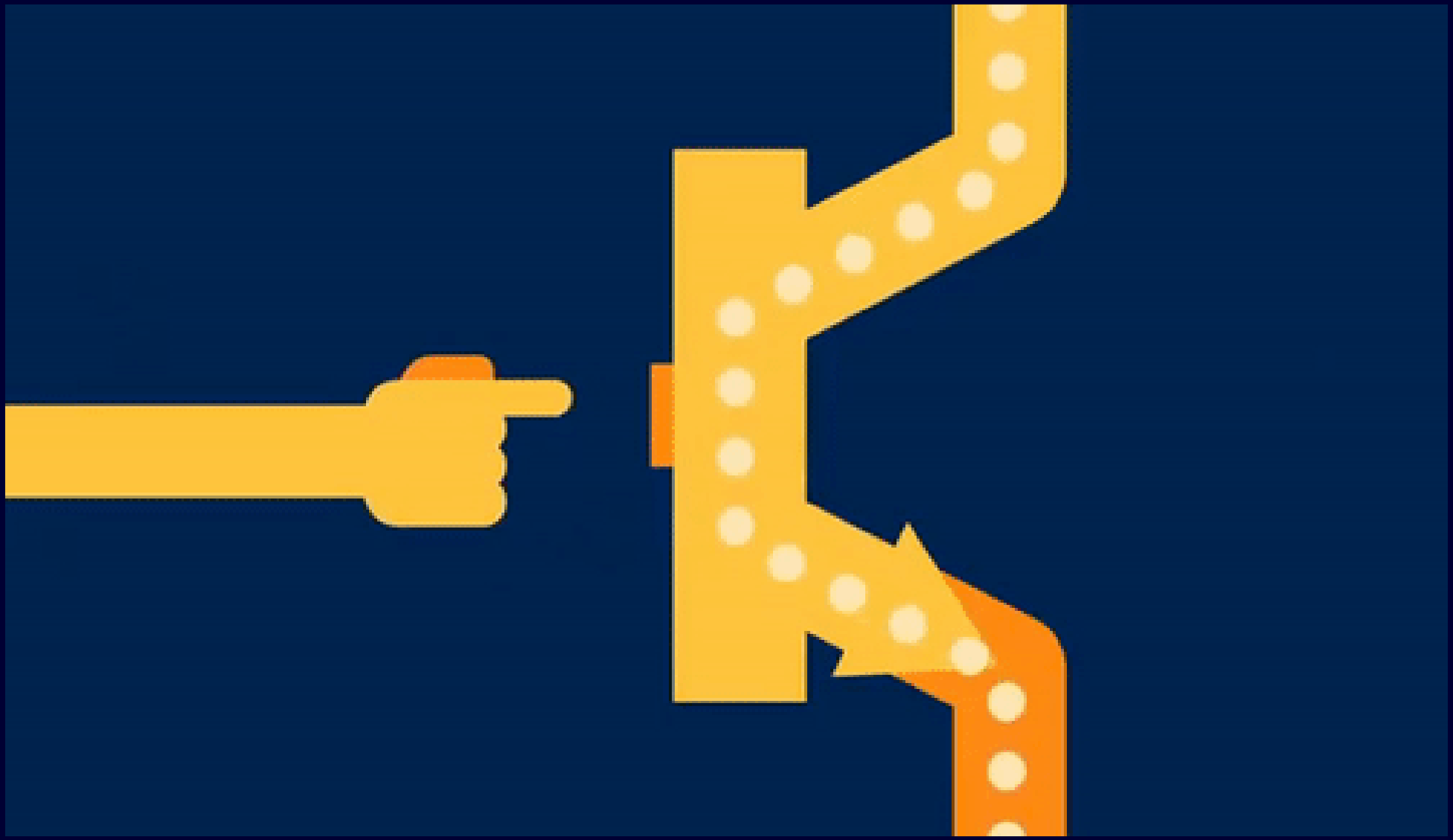
>>> Transistor

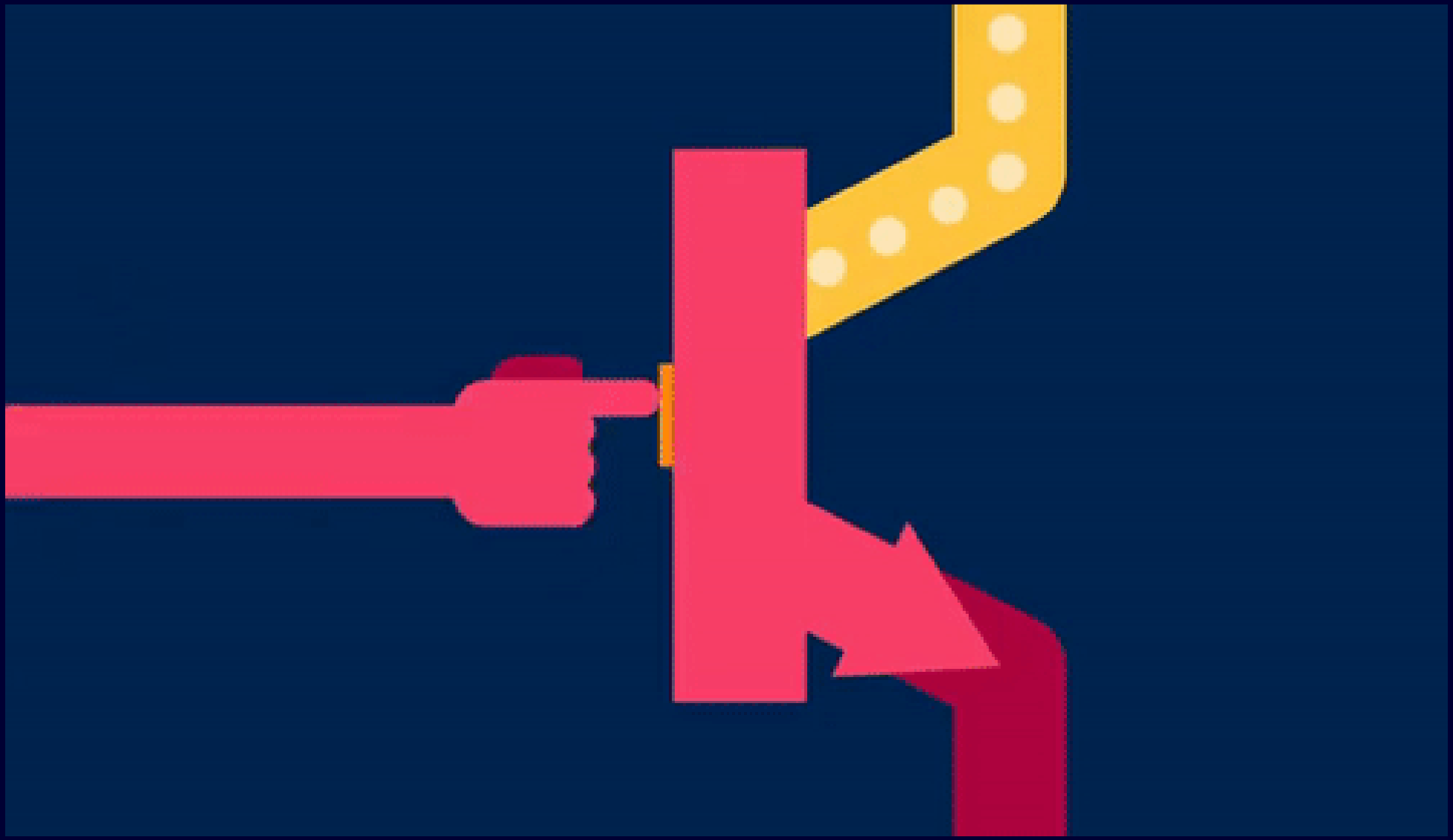




>>> Porque os computadores são como são (3) ?

John Bardeen, William Shockley e Walter Brattain






[1152 bits]

>>> Agrupando padres de 8 em 8 bits: letras

01000001 ← A
01000010 ← B
01000011 ← C
01000100 ← D
01000101 ← E
01000110 ← F
...
01011010 ← Z

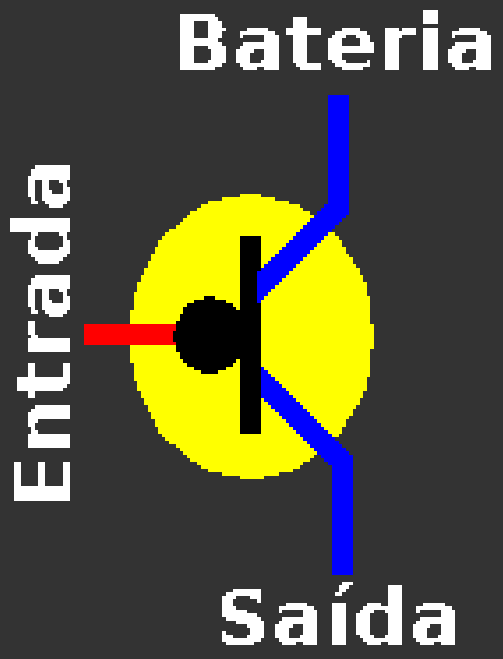
>>> Agrupando padres de 8 em 8 bits: dígitos decimais

000000 ← 0
000001 ← 1
000010 ← 2
000011 ← 3
000100 ← 4
000101 ← 5
000110 ← 6
000111 ← 7

>>> Níveis de Abstração

A computação funciona bem porque criamos níveis de abstração

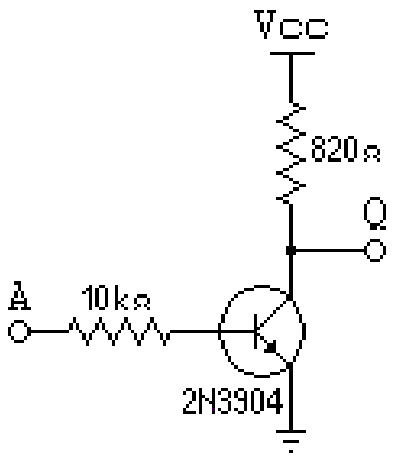
>>> Níveis de Abstração



>>> Combinando transistores (1)

Truth Table	
Input	Output
A	Q
1	0
0	1

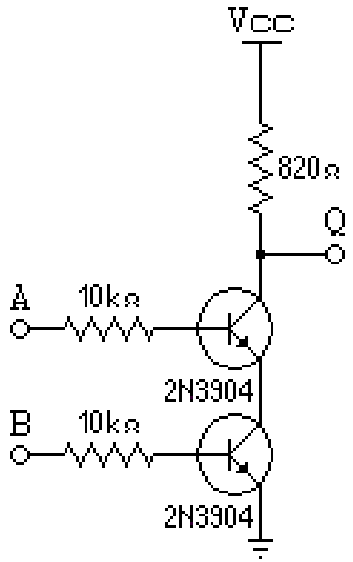
Transistor Inverter Gate



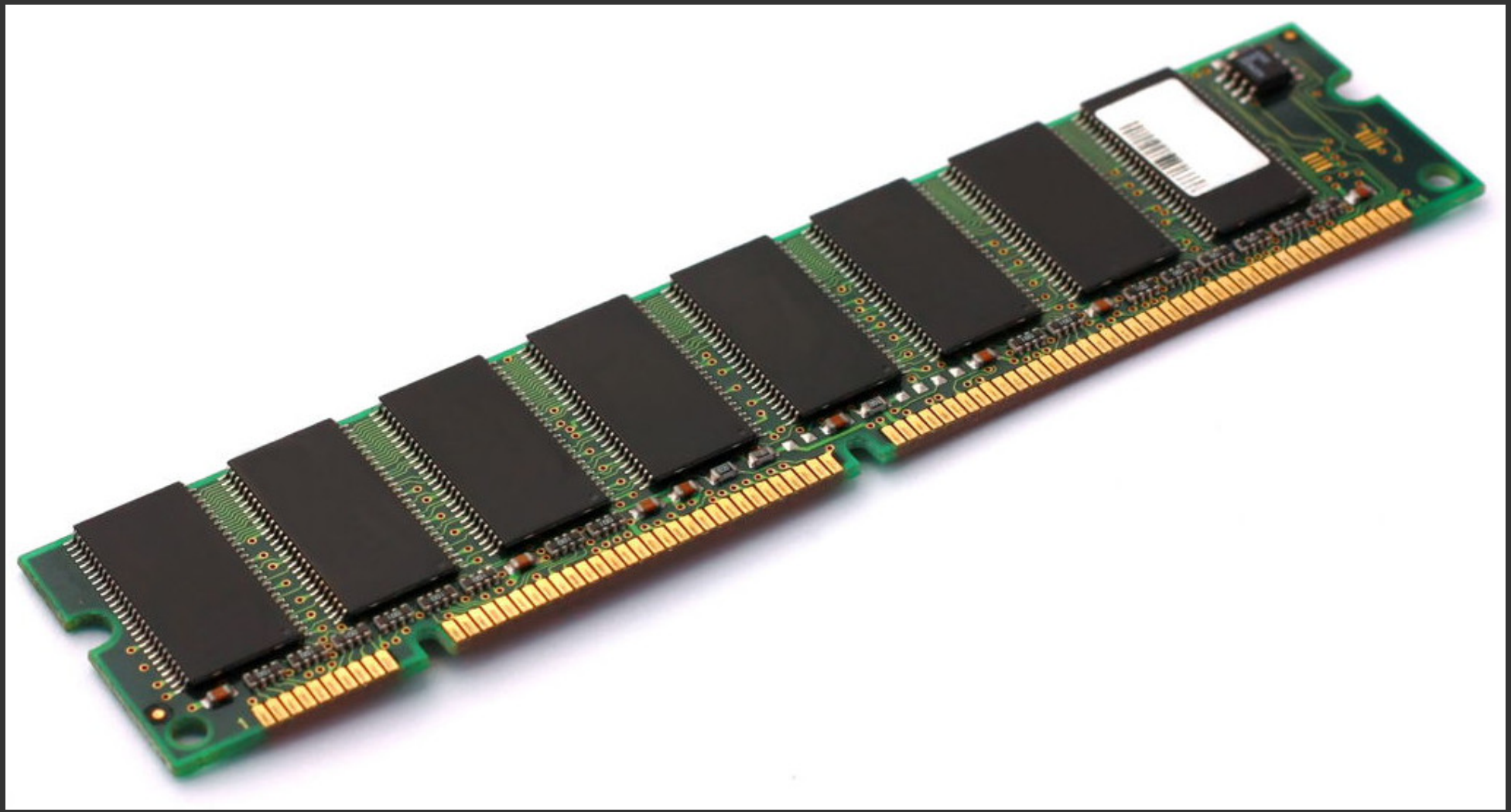
>>> Combinando transistores (2)

Truth Table		
Input		Output
A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

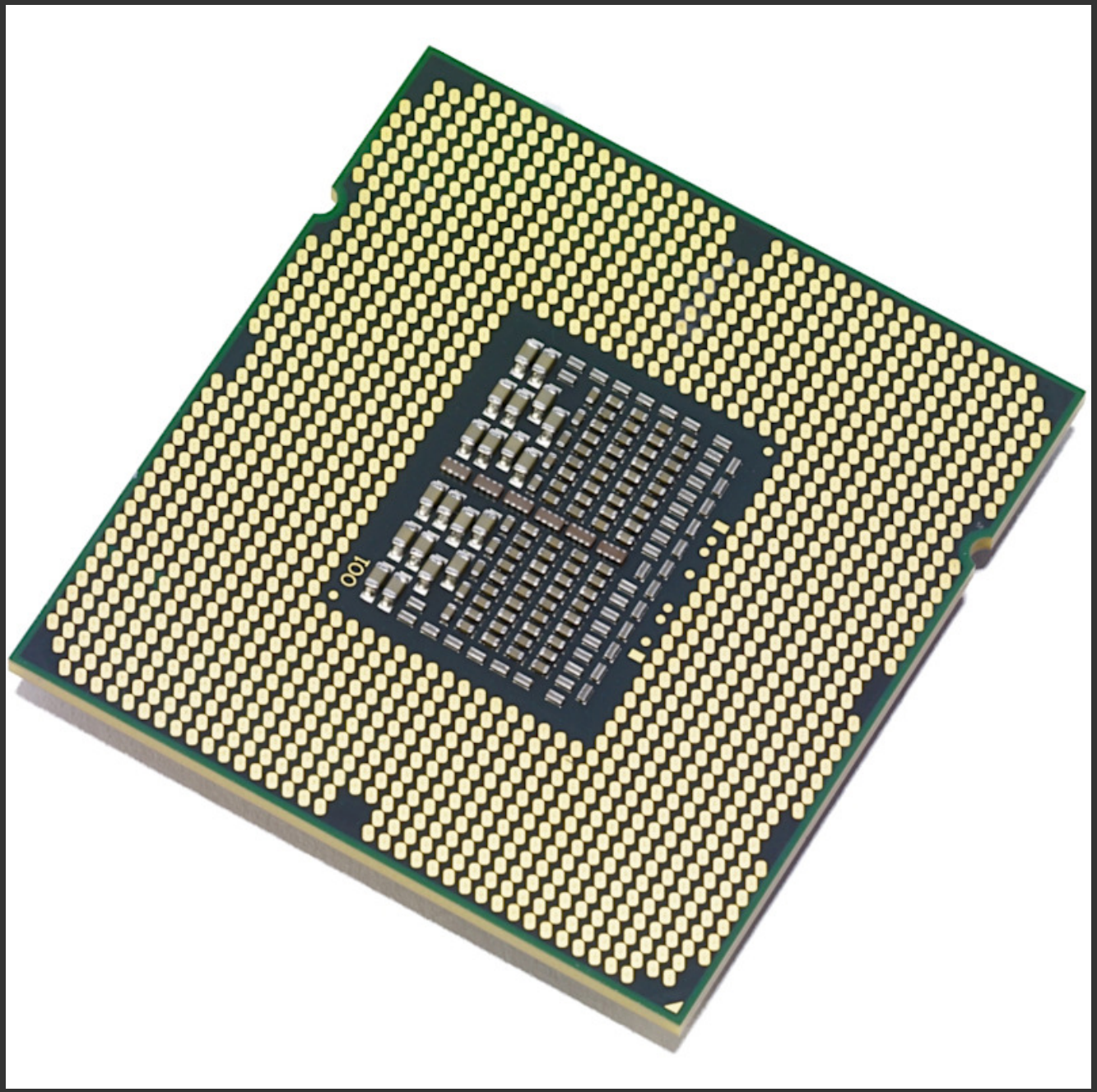
Input Transistor Nand Gate



>>> A Memória



>>> O Processador



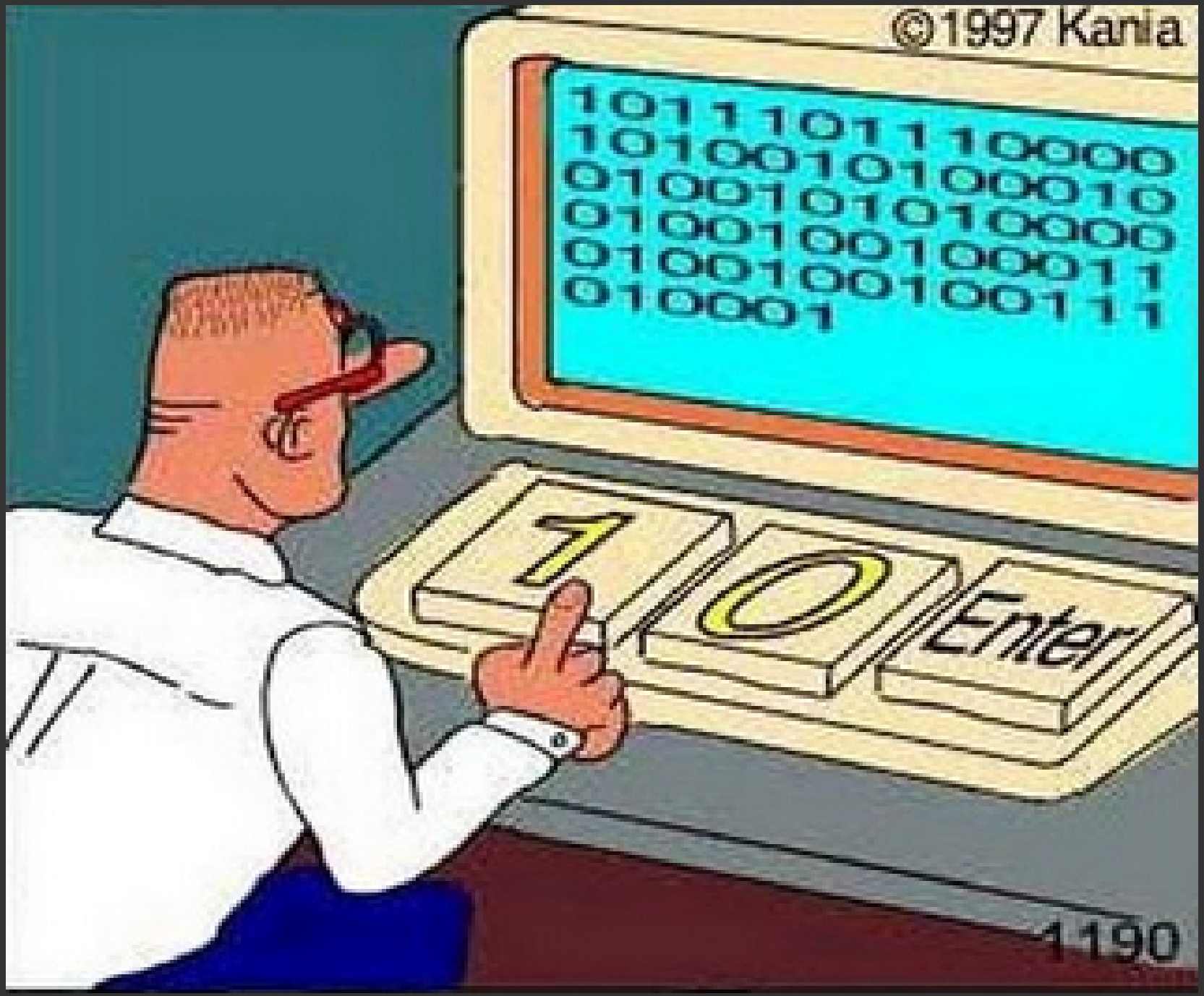
Wow

Programação

Computadores

BITS

E..... como é que a gente resolve problemas com isso?


©1997 Kania
1011101111010000
1010010100010
0100101010000
0100100100011
0100100100111
010001
1
0
Enter
1190

>>> Nível de abstração --- linguagem assembly

```
.section .rodata
    fmt:    .string "%d\n"
.text
.global main
main:
    push    %rbp
    mov     %rsp, %rbp
    mov     $10, %eax
    add     $1, %eax
    add     $20, %eax
    mov     %eax, %esi
    mov     $fmt, %edi
    xor     %eax, %eax
    call    printf
    mov     $0, %eax
    leave
    ret
```

>>> Nível de abstração --- linguagem C

```
int main (void) {
    int a, b;
    a = 10;
    a = a + 1;
    b = a + 20;
    printf("%d", b);
    return 0;
}
```

>>> Nível de abstração --- linguagem Python

```
a = 10
a = a + 1
b = a + 20
print(b)
```

>>> Computação

* Influenciada diretamente pela Matemática, Engenharia Elétrica/Eletrnica e pela Estatística.
* Lida com representaçes da informação e suas abstraçes (códigos)
* Estuda formas de processar esses códigos
* Aplica métodos para problemas do mundo real