MAC 110 — Introdução à Ciência da Computação

Aula 24

Nelson Lago

BMAC – 2024

Números de ponto flutuante e suas pegadinhas

Números de ponto flutuante

- **Em python, números inteiros podem ser tão grandes quanto necessário**

Números de ponto flutuante

- **Em python, números inteiros podem ser tão grandes quanto necessário**
  - (Até o limite da memória disponível)

Números de ponto flutuante

- **Em python, números inteiros podem ser tão grandes quanto necessário**
  - (Até o limite da memória disponível)
  - Em outras linguagens, geralmente há um tamanho máximo

Números de ponto flutuante

- **Em python, números inteiros podem ser tão grandes quanto necessário**
  - (Até o limite da memória disponível)
  - Em outras linguagens, geralmente há um tamanho máximo
- **A menos que você esteja lidando com números muito grandes, cada um deles ocupa pouca memória**

Números de ponto flutuante

- **Mas e números como $\frac{2}{3}$ ou $\sqrt{2}$?**

Números de ponto flutuante

- **Mas e números como $\frac{2}{3}$ ou $\sqrt{2}$?**
  - Não é possível representá-los com precisão em um sistema com memória finita

Números de ponto flutuante

- **Mas e números como $\frac{2}{3}$ ou $\sqrt{2}$?**
  - Não é possível representá-los com precisão em um sistema com memória finita
- **E números "pequenos", como** $4{,}30 \times 10^{-7}$**?**

Números de ponto flutuante

- **Mas e números como $\frac{2}{3}$ ou $\sqrt{2}$?**
  - Não é possível representá-los com precisão em um sistema com memória finita
- **E números "pequenos", como $4{,}30 \times 10^{-7}$?**
  - Não são incomuns!

Números de ponto flutuante

- **Mas e números como $\frac{2}{3}$ ou $\sqrt{2}$?**
  - Não é possível representá-los com precisão em um sistema com memória finita
- **E números "pequenos", como $4{,}30 \times 10^{-7}$?**
  - Não são incomuns!
- **Uma representação ingênua é escolher um valor mínimo, como 10^{-12}, e tratá-lo como "1"**

Números de ponto flutuante

- **Mas e números como $\frac{2}{3}$ ou $\sqrt{2}$?**
  - Não é possível representá-los com precisão em um sistema com memória finita
- **E números "pequenos", como $4{,}30 \times 10^{-7}$?**
  - Não são incomuns!
- **Uma representação ingênua é escolher um valor mínimo, como 10^{-12}, e tratá-lo como "1"**
  - Mas com isso os inteiros "normais" passam a gastar mais memória

Números de ponto flutuante

- **Mas e números como $\frac{2}{3}$ ou $\sqrt{2}$?**
  - Não é possível representá-los com precisão em um sistema com memória finita
- **E números "pequenos", como $4{,}30 \times 10^{-7}$?**
  - Não são incomuns!
- **Uma representação ingênua é escolher um valor mínimo, como 10^{-12}, e tratá-lo como "1"**
  - Mas com isso os inteiros "normais" passam a gastar mais memória
  - Além disso, os números próximos ao mínimo têm pouca precisão

Números de ponto flutuante

- **Mas e números como $\frac{2}{3}$ ou $\sqrt{2}$?**
  - Não é possível representá-los com precisão em um sistema com memória finita
- **E números "pequenos", como $4{,}30 \times 10^{-7}$?**
  - Não são incomuns!
- **Uma representação ingênua é escolher um valor mínimo, como 10^{-12}, e tratá-lo como "1"**
  - Mas com isso os inteiros "normais" passam a gastar mais memória
  - Além disso, os números próximos ao mínimo têm pouca precisão
  - O que acontece se precisarmos do número 10^{-13}?

Números de ponto flutuante

- **Mas e números como $\frac{2}{3}$ ou $\sqrt{2}$?**
  - Não é possível representá-los com precisão em um sistema com memória finita
- **E números "pequenos", como $4{,}30 \times 10^{-7}$?**
  - Não são incomuns!
- **Uma representação ingênua é escolher um valor mínimo, como 10^{-12}, e tratá-lo como "1"**
  - Mas com isso os inteiros "normais" passam a gastar mais memória
  - Além disso, os números próximos ao mínimo têm pouca precisão
  - O que acontece se precisarmos do número 10^{-13}?
    - » *(ele não é tãããão pequeno assim...)*

Números de ponto flutuante

- **Mas e números como $\frac{2}{3}$ ou $\sqrt{2}$?**
  - Não é possível representá-los com precisão em um sistema com memória finita
- **E números "pequenos", como $4{,}30 \times 10^{-7}$?**
  - Não são incomuns!
- **Uma representação ingênua é escolher um valor mínimo, como 10^{-12}, e tratá-lo como "1"**
  - Mas com isso os inteiros "normais" passam a gastar mais memória
  - Além disso, os números próximos ao mínimo têm pouca precisão
  - O que acontece se precisarmos do número 10^{-13}?
    - » *(ele não é tãããão pequeno assim...)*
- **Essa solução é usada apenas em alguns casos especiais**

Números de ponto flutuante

- O que fazemos é representar esses números usando a notação científica: **mantissa** $\times 10^{\textbf{expoente}}$

Números de ponto flutuante

- **O que fazemos é representar esses números usando a notação científica: mantissa $\times 10^{\text{expoente}}$**
  - É possível representar números em uma faixa muito maior (na maioria dos sistemas com python, de ~2,225 $\times 10^{-308}$ a ~1,797 $\times 10^{308}$)

Números de ponto flutuante

- **O que fazemos é representar esses números usando a notação científica: mantissa $\times 10^{\text{expoente}}$**
  - É possível representar números em uma faixa muito maior (na maioria dos sistemas com python, de ~2,225 $\times 10^{-308}$ a ~1,797 $\times 10^{308}$)
  - A precisão dos números é independente de sua grandeza

Números de ponto flutuante

- **O que fazemos é representar esses números usando a notação científica: mantissa $\times 10^{\text{expoente}}$**
  - É possível representar números em uma faixa muito maior (na maioria dos sistemas com python, de ~2,225 $\times 10^{-308}$ a ~1,797 $\times 10^{308}$)
  - A precisão dos números é independente de sua grandeza
    - » *depende apenas do tamanho da mantissa (na maioria dos sistemas com python, cerca de 15 dígitos decimais)*

Números de ponto flutuante

- **O que fazemos é representar esses números usando a notação científica: mantissa $\times 10^{\text{expoente}}$**
  - É possível representar números em uma faixa muito maior (na maioria dos sistemas com python, de ~2,225 $\times 10^{-308}$ a ~1,797 $\times 10^{308}$)
  - A precisão dos números é independente de sua grandeza
    - » *depende apenas do tamanho da mantissa (na maioria dos sistemas com python, cerca de 15 dígitos decimais)*
  - O consumo de memória é pequeno

Números de ponto flutuante

- **O que fazemos é representar esses números usando a notação científica: mantissa $\times 10^{\text{expoente}}$**
  - É possível representar números em uma faixa muito maior (na maioria dos sistemas com python, de ~2,225 $\times 10^{-308}$ a ~1,797 $\times 10^{308}$)
  - A precisão dos números é independente de sua grandeza
    - » *depende apenas do tamanho da mantissa (na maioria dos sistemas com python, cerca de 15 dígitos decimais)*
  - O consumo de memória é pequeno
- **MAS**

Números de ponto flutuante

- **O que fazemos é representar esses números usando a notação científica: mantissa $\times 10^{\text{expoente}}$**
  - É possível representar números em uma faixa muito maior (na maioria dos sistemas com python, de ~2,225 $\times 10^{-308}$ a ~1,797 $\times 10^{308}$)
  - A precisão dos números é independente de sua grandeza
    - » *depende apenas do tamanho da mantissa (na maioria dos sistemas com python, cerca de 15 dígitos decimais)*
  - O consumo de memória é pequeno
- **MAS**
- **O limite de precisão pode causar surpresas**

Números de ponto flutuante

- **Imagine uma representação em que a mantissa tem até 4 dígitos e o expoente pode variar de 10^{-4} até 10^{5}**

Números de ponto flutuante

- **Imagine uma representação em que a mantissa tem até 4 dígitos e o expoente pode variar de 10^{-4} até 10^5**
  - O maior número representável seria $9{,}999 \times 10^5$, ou seja, 999.900

Números de ponto flutuante

- **Imagine uma representação em que a mantissa tem até 4 dígitos e o expoente pode variar de 10^{-4} até 10^{5}**
  - O maior número representável seria $9{,}999 \times 10^{5}$, ou seja, 999.900
- **Não seria possível representar o número 34.567!**

Números de ponto flutuante

- **Imagine uma representação em que a mantissa tem até 4 dígitos e o expoente pode variar de 10^{-4} até 10^{5}**
  - O maior número representável seria $9{,}999 \times 10^{5}$, ou seja, 999.900
- **Não seria possível representar o número 34.567!**
  - Seria preciso usar $3{,}457 \times 10^{4}$, ou seja, 34.570

E como lidar com isso?

E como lidar com isso?

Na maior parte do tempo,
não é preciso se preocupar

Imprecisões com ponto flutuante

- **Testes de igualdade**

Imprecisões com ponto flutuante

- **Testes de igualdade**
  - Números que deveriam ser iguais, mas são diferentes por erros de arredondamento, vão causar erros em condicionais (`if`, `while`)

Imprecisões com ponto flutuante

- **Testes de igualdade**
  - Números que deveriam ser iguais, mas são diferentes por erros de arredondamento, vão causar erros em condicionais (`if`, `while`)

```
import math
if math.sqrt(2)**2 == 2:
    …
```

Imprecisões com ponto flutuante

- **Testes de igualdade**
  - Números que deveriam ser iguais, mas são diferentes por erros de arredondamento, vão causar erros em condicionais (`if`, `while`)

```
import math
if math.sqrt(2)**2 == 2:
    …
```

Imprecisões com ponto flutuante

- **Testes de igualdade**
  - Números que deveriam ser iguais, mas são diferentes por erros de arredondamento, vão causar erros em condicionais (`if`, `while`)

```
import math
if math.sqrt(2)**2 == 2:
    …
```

```
import math
epsilon = 1e-15 # É preciso escolher um valor "razoável"
if abs(math.sqrt(2)**2 - 2) < epsilon:
    …
```

Imprecisões com ponto flutuante

- **Multiplicações e divisões não costumam causar problemas**

Imprecisões com ponto flutuante

- **Multiplicações e divisões não costumam causar problemas**
  - Independentemente da grandeza (expoente), os cálculos são feitos com toda a precisão da mantissa

Imprecisões com ponto flutuante

- **Multiplicações e divisões não costumam causar problemas**
  - Independentemente da grandeza (expoente), os cálculos são feitos com toda a precisão da mantissa
    - » $(1,234 \times 10^{8}) \times (5,678 \times 10^{-3})$

Imprecisões com ponto flutuante

- **Multiplicações e divisões não costumam causar problemas**
  - Independentemente da grandeza (expoente), os cálculos são feitos com toda a precisão da mantissa
    - » $(1,234 \times 10^{8}) \times (5,678 \times 10^{-3}) = 1,234 \times 5,678 \times 10^{5}$

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**
  - Somas/subtrações com números de grandezas muito diferentes

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**
  - Somas/subtrações com números de grandezas muito diferentes

```
» print("{:.25e}".format(2e10 + 4e-10))
```

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**
  - Somas/subtrações com números de grandezas muito diferentes

```
» print("{:.25e}".format(2e10 + 4e-10))
  2.0000000000000000000000000e+10
```

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**
  - Somas/subtrações com números de grandezas muito diferentes

```
» print("{:.25e}".format(2e10 + 4e-10))
  2.0000000000000000000000000e+10 😱
```

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**
  - Somas/subtrações com números de grandezas muito diferentes

```
» print("{:.25e}".format(2e10 + 4e-10))
  2.0000000000000000000000000e+10 😱
```

  - Subtrações com números muito próximos

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**
  - Somas/subtrações com números de grandezas muito diferentes
    - » `print("{:.25e}".format(2e10 + 4e-10))`

      `2.0000000000000000000000000e+10` 😱
  - Subtrações com números muito próximos
    - » *O resultado da diferença vai ser predominantemente composto pelos dígitos menos significativos, onde o erro é proporcionalmente maior*

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**
  - Somas/subtrações com números de grandezas muito diferentes
    - » `print("{:.25e}".format(2e10 + 4e-10))`

      `2.0000000000000000000000000e+10` 😱
  - Subtrações com números muito próximos
    - » *O resultado da diferença vai ser predominantemente composto pelos dígitos menos significativos, onde o erro é proporcionalmente maior*

      `>>> print("{:.20e}".format(0.1))`

      `1.00000000000000005551e-01` *(erro no dígito 18)*

      `>>> print("{:.20e}".format(.1000000001-.1))` *(diferença no dígito 10)*

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**
  - Somas/subtrações com números de grandezas muito diferentes
    - » `print("{:.25e}".format(2e10 + 4e-10))`

      `2.0000000000000000000000000e+10` 😱
  - Subtrações com números muito próximos
    - » *O resultado da diferença vai ser predominantemente composto pelos dígitos menos significativos, onde o erro é proporcionalmente maior*

      `>>> print("{:.20e}".format(0.1))`

      `1.00000000000000005551e-01` *(erro no dígito 18)*

      `>>> print("{:.20e}".format(.1000000001-.1))` *(diferença no dígito 10)*

      `9.99999943962492920946e-11` *(erro no dígito 8)*

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**
  - Somas/subtrações com números de grandezas muito diferentes
    - » `print("{:.25e}".format(2e10 + 4e-10))`
      `2.0000000000000000000000000e+10` 😱
  - Subtrações com números muito próximos
    - » *O resultado da diferença vai ser predominantemente composto pelos dígitos menos significativos, onde o erro é proporcionalmente maior*

      `>>> print("{:.20e}".format(0.1))`
      `1.00000000000000005551e-01` *(erro no dígito 18)*
      `>>> print("{:.20e}".format(.1000000001-.1))` *(diferença no dígito 10)*
      `9.99999943962492920946e-11` *(erro no dígito 8)*
- **Somatórios podem fazer as imprecisões se combinarem, tornando erros irrelevantes em catastróficos**

Imprecisões com ponto flutuante

- **Somas e subtrações podem causar problemas**
  - Somas/subtrações com números de grandezas muito diferentes
    - » `print("{:.25e}".format(2e10 + 4e-10))`

      `2.0000000000000000000000000e+10` 😱
  - Subtrações com números muito próximos
    - » *O resultado da diferença vai ser predominantemente composto pelos dígitos menos significativos, onde o erro é proporcionalmente maior*

      `>>> print("{:.20e}".format(0.1))`

      `1.00000000000000005551e-01` *(erro no dígito 18)*

      `>>> print("{:.20e}".format(.1000000001-.1))` *(diferença no dígito 10)*

      `9.99999943962492920946e-11` *(erro no dígito 8)*
- **Somatórios podem fazer as imprecisões se combinarem, tornando erros irrelevantes em catastróficos**
  - Em geral, é possível fazer um dado somatório de maneira a minimizar ou maximizar esse problema

Exercício — calculando π

Dado um inteiro $k \geq 0$, é possível calcular um valor aproximado de π através da expansão de Gregory-Leibniz:

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

Escreva um programa que lê um valor para k e realiza o cálculo acima.

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))









    print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))




    while n <= k:



    print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))


    n = 0

    while n <= k:



    print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))


    n = 0

    while n <= k:


        n += 1
    print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))
    pi = 0

    n = 0

    while n <= k:


        n += 1
    print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))
    pi = 0

    n = 0

    while n <= k:
        pi +=          4 / (2*n +1)

        n += 1
    print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))
    pi = 0

    n = 0

    while n <= k:
        pi += sinal * 4 / (2*n +1)

        n += 1
    print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))
    pi = 0
    sinal = 1
    n = 0

    while n <= k:
        pi += sinal * 4 / (2*n +1)
        sinal *= -1
        n += 1
    print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))
    pi = 0
    sinal = 1
    if not k % 2 == 0:
        sinal *= -1
    while k >= 0:
        pi += sinal * 4 / (2*k +1)
        sinal *= -1
        k -= 1
    print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
    k = int(input("Digite o valor de k: "))
    pi = 0
    sinal = 1
    if not k % 2 == 0:
        sinal *= -1
    while k >= 0:
        pi += sinal * 4 / (2*k +1)
        sinal *= -1
        k -= 1
    print("O valor de pi é aproximadamente {}".format(pi))
```

Na prática, com a precisão padrão de python (64 bits), esse caso não é de fato um problema

Um pouco mais de história

commons.wikimedia.org/wiki/File:Recurve_samick_a-z_hg.jpg

commons.wikimedia.org/wiki/File:Crossbow_MET_DP293488.jpg

commons.wikimedia.org/wiki/File:Crossbow_TenPoint_Stealth_SS.png

Evolução (mais ou menos) contínua

(a mesma ideia refinada)

commons.wikimedia.org/wiki/File:Nummen_kirkko,_kynttelikko,_oikea-trimmed.jpg

commons.wikimedia.org/wiki/File:Gluehlampe_01_KMJ.jpg

commons.wikimedia.org/wiki/File:Leuchtstofflampen-chtaube050409.jpg

commons.wikimedia.org/wiki/File:LED-E27-Light-Bulb-1134.jpg

Evolução com saltos qualitativos

(ideias diferentes)

Evolução com saltos qualitativos

(ideias diferentes)

mas as ideias anteriores podem continuar sendo parcialmente usadas ou influenciar a ideia nova

Computadores eletrônicos

- **Os computadores eletrônicos não são apenas um refinamento das calculadoras e outros dispositivos mecânicos**

Computadores eletrônicos

- Os computadores eletrônicos não são apenas um refinamento das calculadoras e outros dispositivos mecânicos
- Três ideias "novas" estão na base dessa diferença:

Computadores eletrônicos

- **Os computadores eletrônicos não são apenas um refinamento das calculadoras e outros dispositivos mecânicos**
- **Três ideias "novas" estão na base dessa diferença:**
  - Aritmética de base 2 (operações numéricas)

Computadores eletrônicos

- **Os computadores eletrônicos não são apenas um refinamento das calculadoras e outros dispositivos mecânicos**
- **Três ideias "novas" estão na base dessa diferença:**
  - Aritmética de base 2 (operações numéricas)
  - Álgebra booleana (operações lógicas)

Computadores eletrônicos

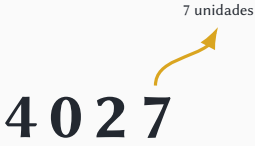
- **Os computadores eletrônicos não são apenas um refinamento das calculadoras e outros dispositivos mecânicos**
- **Três ideias "novas" estão na base dessa diferença:**
  - Aritmética de base 2 (operações numéricas)
  - Álgebra booleana (operações lógicas)
  - Corrente elétrica

Computadores eletrônicos

- **Os computadores eletrônicos não são apenas um refinamento das calculadoras e outros dispositivos mecânicos**
- **Três ideias "novas" estão na base dessa diferença:**
  - Aritmética de base 2 (operações numéricas)
  - Álgebra booleana (operações lógicas)
  - Corrente elétrica

4027

Números de base 10



Números de base 10

Números de base 10

Números de base 10

Números de base 10

Números de base 10

- **Dez símbolos**
  - 0 1 2 3 4 5 6 7 8 9

Sistemas de numeração

- **O que o número dez tem de especial?**

Sistemas de numeração

- O que o número dez tem de especial?
- nada

Sistemas de numeração

- **O que o número dez tem de especial?**
- **nada**
  - (ok, temos dez dedos nas mãos)

Sistemas de numeração

- **O que o número dez tem de especial?**
- **nada**
  - (ok, temos dez dedos nas mãos)
- **Outros sistemas já foram usados**

Sistemas de numeração

- **O que o número dez tem de especial?**
- **nada**
  - (ok, temos dez dedos nas mãos)
- **Outros sistemas já foram usados**
  - Sexagesimal (base 60)

Sistemas de numeração

- **O que o número dez tem de especial?**
- **nada**
  - (ok, temos dez dedos nas mãos)
- **Outros sistemas já foram usados**
  - Sexagesimal (base 60)
    - » *Horas, minutos, graus...*

Sistemas de numeração

- **O que o número dez tem de especial?**
- **nada**
  - (ok, temos dez dedos nas mãos)
- **Outros sistemas já foram usados**
  - Sexagesimal (base 60)
    - » *Horas, minutos, graus...*
  - Duodecimal (base 12)

Sistemas de numeração

- **O que o número dez tem de especial?**
- **nada**
  - (ok, temos dez dedos nas mãos)
- **Outros sistemas já foram usados**
  - Sexagesimal (base 60)
    - » *Horas, minutos, graus...*
  - Duodecimal (base 12)
    - » *Dúzias, grosas...*

Sistemas de numeração

- **O que o número dez tem de especial?**
- **nada**
  - (ok, temos dez dedos nas mãos)
- **Outros sistemas já foram usados**
  - Sexagesimal (base 60)
    - » *Horas, minutos, graus...*
  - Duodecimal (base 12)
    - » *Dúzias, grosas...*
- **O "menor" sistema possível usa base dois**

Sistemas de numeração

- **O que o número dez tem de especial?**
- **nada**
  - (ok, temos dez dedos nas mãos)
- **Outros sistemas já foram usados**
  - Sexagesimal (base 60)
    - » *Horas, minutos, graus...*
  - Duodecimal (base 12)
    - » *Dúzias, grosas...*
- **O "menor" sistema possível usa base dois**
  - Não dá para ter menos de dois símbolos!

Sistemas de numeração

- **O que o número dez tem de especial?**
- **nada**
  - (ok, temos dez dedos nas mãos)
- **Outros sistemas já foram usados**
  - Sexagesimal (base 60)
    - » *Horas, minutos, graus...*
  - Duodecimal (base 12)
    - » *Dúzias, grosas...*
- **O "menor" sistema possível usa base dois**
  - Não dá para ter menos de dois símbolos!
    - » *Mesmo "contando palitinhos" (**IIIII**...), ainda temos a presença ou a ausência deles*

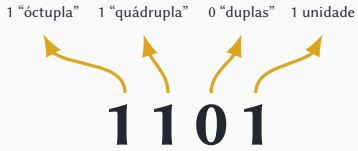
1101

Números de base 2

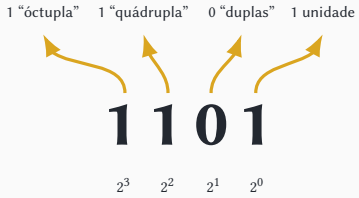
Números de base 2

Números de base 2

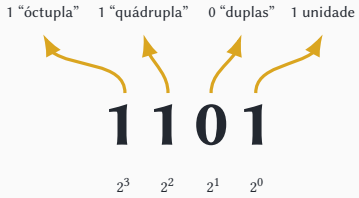
Números de base 2



Números de base 2

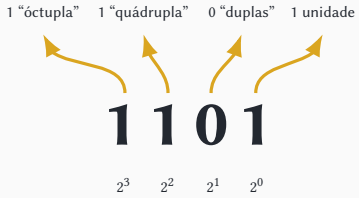


Números de base 2



(em decimal: 13)

Números de base 2



(em decimal: 13)

- **Dois símbolos**
  - 0 1

$$4 + 5$$

Números de base 2

$$4 + 5$$

$$0100$$

Números de base 2

$$4 + 5$$

$$\begin{array}{r} 0100 \\ +\,0101 \end{array}$$

Números de base 2

$$4 + 5$$

$$\begin{array}{r} 0100 \\ +\,0101 \\ \hline \end{array}$$

Números de base 2

$$4 + 5$$

$$\begin{array}{r} 0100 \\ +\,0101 \\ \hline 1 \end{array}$$

Números de base 2

$$4 + 5$$

$$\begin{array}{r} 0100 \\ +\,0101 \\ \hline 01 \end{array}$$

Números de base 2

$$4 + 5$$

$$\begin{array}{r} 0100 \\ +\,0101 \\ \hline 001 \end{array}$$

Números de base 2

$$4 + 5$$

$$\begin{array}{r} {\scriptstyle 1} \\ 0100 \\ +\,0101 \\ \hline 001 \end{array}$$

Números de base 2

$$4 + 5$$

$$\begin{array}{r} \overset{1}{} \\ 0100 \\ +\,0101 \\ \hline 1001 \end{array}$$

Números de base 2

$$4 + 5$$

$$\begin{array}{r} {\scriptstyle 1} \\ 0100 \\ +\,0101 \\ \hline 1001 \end{array}$$

$$9$$

Números de base 2

$$4 + 5$$

$$\begin{array}{r} {\scriptstyle 1} \\ 0100 \\ +\,0101 \\ \hline 1001 \end{array}$$

$$9$$

Intermezzo

WTF “bit”, “byte”, “hexadecimal”?

Bits e bytes

- bit: binary digit

Bits e bytes

- bit: **b**inary digi**t**
- “1011” é o número onze ou o número dois seguido do número três?

Bits e bytes

- **bit: binary digit**
- **"1011" é o número onze ou o número dois seguido do número três?**
  - Na linguagem escrita, sabemos onde começam e terminam as palavras ou números porque usamos espaços

Bits e bytes

- **bit: binary digit**
- **"1011" é o número onze ou o número dois seguido do número três?**
  - Na linguagem escrita, sabemos onde começam e terminam as palavras ou números porque usamos espaços
  - No computador, em geral usamos um tamanho fixo para cada número/letra ⟶ byte

Bits e bytes

- **bit: binary digit**
- **"1011" é o número onze ou o número dois seguido do número três?**
  - Na linguagem escrita, sabemos onde começam e terminam as palavras ou números porque usamos espaços
  - No computador, em geral usamos um tamanho fixo para cada número/letra ⟶ byte
  - Historicamente, diversos tamanhos; em particular, 6 bits foi bastante usado

Bits e bytes

- **bit: binary digit**
- **"1011" é o número onze ou o número dois seguido do número três?**
  - Na linguagem escrita, sabemos onde começam e terminam as palavras ou números porque usamos espaços
  - No computador, em geral usamos um tamanho fixo para cada número/letra ⟶ byte
  - Historicamente, diversos tamanhos; em particular, 6 bits foi bastante usado
  - Hoje em dia é "padrão" usar 8 bits

Bits e bytes

- **bit: binary digit**
- **"1011" é o número onze ou o número dois seguido do número três?**
  - Na linguagem escrita, sabemos onde começam e terminam as palavras ou números porque usamos espaços
  - No computador, em geral usamos um tamanho fixo para cada número/letra ⟶ byte
  - Historicamente, diversos tamanhos; em particular, 6 bits foi bastante usado
  - Hoje em dia é "padrão" usar 8 bits
    - » *Números 0–255*

Bits e bytes

- **bit: binary digit**
- **"1011" é o número onze ou o número dois seguido do número três?**
  - Na linguagem escrita, sabemos onde começam e terminam as palavras ou números porque usamos espaços
  - No computador, em geral usamos um tamanho fixo para cada número/letra ⟶ byte
  - Historicamente, diversos tamanhos; em particular, 6 bits foi bastante usado
  - Hoje em dia é "padrão" usar 8 bits
    - » *Números 0–255*
    - » *Todas as letras latinas e espaço adicional para outros caracteres dependendo da língua*

Hexadecimal

- **Normalmente pensamos em números decimais e deixamos para o computador a tarefa de traduzir binário ↔ decimal**

Hexadecimal

- Normalmente pensamos em números decimais e deixamos para o computador a tarefa de traduzir binário ↔ decimal
- Às vezes precisamos "lembrar" dos números binários...

Hexadecimal

- Normalmente pensamos em números decimais e deixamos para o computador a tarefa de traduzir binário ↔ decimal
- Às vezes precisamos "lembrar" dos números binários...
- Mas é muito ruim manipular números como 10110111! (183)

Hexadecimal

- Normalmente pensamos em números decimais e deixamos para o computador a tarefa de traduzir binário ↔ decimal
- Às vezes precisamos "lembrar" dos números binários...
- Mas é muito ruim manipular números como 10110111! (183)

Hexadecimal

- Normalmente pensamos em números decimais e deixamos para o computador a tarefa de traduzir binário ↔ decimal
- Às vezes precisamos "lembrar" dos números binários...
- Mas é muito ruim manipular números como 10110111! (183)

- 8 bits ⟶ 4 bits (números 0–15) + 4 bits (números 0–15)

Hexadecimal

- Normalmente pensamos em números decimais e deixamos para o computador a tarefa de traduzir binário ↔ decimal
- Às vezes precisamos "lembrar" dos números binários...
- Mas é muito ruim manipular números como 10110111! (183)

- 8 bits ⟶ 4 bits (números 0–15) + 4 bits (números 0–15)
- Podemos representar um byte como 2 dígitos na base 16!

Hexadecimal

- **Normalmente pensamos em números decimais e deixamos para o computador a tarefa de traduzir binário ↔ decimal**
- **Às vezes precisamos "lembrar" dos números binários...**
- **Mas é muito ruim manipular números como 10110111! (183)**

- **8 bits ⟶ 4 bits (números 0–15) + 4 bits (números 0–15)**
- **Podemos representar um byte como 2 dígitos na base 16!**
  - Um sistema numérico com base 16 (hexadecimal) precisa de 16 símbolos

Hexadecimal

- **Normalmente pensamos em números decimais e deixamos para o computador a tarefa de traduzir binário ↔ decimal**
- **Às vezes precisamos "lembrar" dos números binários...**
- **Mas é muito ruim manipular números como 10110111! (183)**

- **8 bits ⟶ 4 bits (números 0–15) + 4 bits (números 0–15)**
- **Podemos representar um byte como 2 dígitos na base 16!**
  - Um sistema numérico com base 16 (hexadecimal) precisa de 16 símbolos
    - » *Que tal usar 0–9 e juntar A–F?*

Hexadecimal

- **Normalmente pensamos em números decimais e deixamos para o computador a tarefa de traduzir binário ↔ decimal**
- **Às vezes precisamos "lembrar" dos números binários...**
- **Mas é muito ruim manipular números como 10110111! (183)**

- **8 bits ⟶ 4 bits (números 0–15) + 4 bits (números 0–15)**
- **Podemos representar um byte como 2 dígitos na base 16!**
  - Um sistema numérico com base 16 (hexadecimal) precisa de 16 símbolos
    - » *Que tal usar 0–9 e juntar A–F?*
  - 10110111 ⟶ 1011 0111 ⟶ B7 ⟶ $11 * 16^1 + 7 * 16^0$ ⟶ 183

Hexadecimal

- **Normalmente pensamos em números decimais e deixamos para o computador a tarefa de traduzir binário ↔ decimal**
- **Às vezes precisamos "lembrar" dos números binários...**
- **Mas é muito ruim manipular números como 10110111! (183)**

- **8 bits ⟶ 4 bits (números 0–15) + 4 bits (números 0–15)**
- **Podemos representar um byte como 2 dígitos na base 16!**
  - Um sistema numérico com base 16 (hexadecimal) precisa de 16 símbolos
    - » *Que tal usar 0–9 e juntar A–F?*
  - 10110111 ⟶ 1011 0111 ⟶ B7 ⟶ $11 * 16^1 + 7 * 16^0$ ⟶ 183
  - Com um pouco de prática, não é muito difícil fazer as conversões "de cabeça"

Números de ponto flutuante e representação binária

- Os números de ponto flutuante também são armazenados no sistema binário, ou seja, $\text{mantissa}_2 \times 2^{\text{expoente}}$

Números de ponto flutuante e representação binária

- **Os números de ponto flutuante também são armazenados no sistema binário, ou seja, $\text{mantissa}_2 \times 2^{\text{expoente}}$**
  - Em python, sempre 64 bits: 1 para o sinal, 11 para o expoente e 52 para a mantissa

Números de ponto flutuante e representação binária

- **Os números de ponto flutuante também são armazenados no sistema binário, ou seja, $\text{mantissa}_2 \times 2^{\text{expoente}}$**
  - Em python, sempre 64 bits: 1 para o sinal, 11 para o expoente e 52 para a mantissa
  - Em várias outras linguagens, é possível escolher o tamanho da memória utilizada: 16, 32, 64 etc. bits (padrão IEEE 754)

Números de ponto flutuante e representação binária

- **Os números de ponto flutuante também são armazenados no sistema binário, ou seja, $\text{mantissa}_2 \times 2^{\text{expoente}}$**
  - Em python, sempre 64 bits: 1 para o sinal, 11 para o expoente e 52 para a mantissa
  - Em várias outras linguagens, é possível escolher o tamanho da memória utilizada: 16, 32, 64 etc. bits (padrão IEEE 754)
    - » *A biblioteca NumPy traz esse recurso para python*

Números de ponto flutuante e representação binária

- **Os números de ponto flutuante também são armazenados no sistema binário, ou seja, $\text{mantissa}_2 \times 2^{\text{expoente}}$**
  - Em python, sempre 64 bits: 1 para o sinal, 11 para o expoente e 52 para a mantissa
  - Em várias outras linguagens, é possível escolher o tamanho da memória utilizada: 16, 32, 64 etc. bits (padrão IEEE 754)
    - » *A biblioteca NumPy traz esse recurso para python*
- **Isso importa porque**

Números de ponto flutuante e representação binária

- **Os números de ponto flutuante também são armazenados no sistema binário, ou seja, $\text{mantissa}_2 \times 2^{\text{expoente}}$**
  - Em python, sempre 64 bits: 1 para o sinal, 11 para o expoente e 52 para a mantissa
  - Em várias outras linguagens, é possível escolher o tamanho da memória utilizada: 16, 32, 64 etc. bits (padrão IEEE 754)
    - » *A biblioteca NumPy traz esse recurso para python*
- **Isso importa porque**
  - Toda dízima periódica é também um "binário periódico"

Números de ponto flutuante e representação binária

- **Os números de ponto flutuante também são armazenados no sistema binário, ou seja, $\text{mantissa}_2 \times 2^{\text{expoente}}$**
  - Em python, sempre 64 bits: 1 para o sinal, 11 para o expoente e 52 para a mantissa
  - Em várias outras linguagens, é possível escolher o tamanho da memória utilizada: 16, 32, 64 etc. bits (padrão IEEE 754)
    - » *A biblioteca NumPy traz esse recurso para python*
- **Isso importa porque**
  - Toda dízima periódica é também um "binário periódico"
  - Existem "binários periódicos" que não são dízimas periódicas

Números de ponto flutuante e representação binária

- **Os números de ponto flutuante também são armazenados no sistema binário, ou seja, $\text{mantissa}_2 \times 2^{\text{expoente}}$**
  - Em python, sempre 64 bits: 1 para o sinal, 11 para o expoente e 52 para a mantissa
  - Em várias outras linguagens, é possível escolher o tamanho da memória utilizada: 16, 32, 64 etc. bits (padrão IEEE 754)
    - » *A biblioteca NumPy traz esse recurso para python*
- **Isso importa porque**
  - Toda dízima periódica é também um "binário periódico"
  - Existem "binários periódicos" que não são dízimas periódicas

```
» print(0.1 + 0.1 + 0.1)
  0.30000000000000004
```

Números de ponto flutuante e representação binária

- **Dado um racional expresso como $\frac{i}{j}$ em sua forma irredutível, esse racional será periódico na base b se algum dos fatores primos que compõem o denominador j não for fator primo da base b**

Números de ponto flutuante e representação binária

- **Dado um racional expresso como $\frac{i}{j}$ em sua forma irredutível, esse racional será periódico na base b se algum dos fatores primos que compõem o denominador j não for fator primo da base b**
  - Ou seja, frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 e 5 são dízimas periódicas

Números de ponto flutuante e representação binária

- **Dado um racional expresso como $\frac{i}{j}$ em sua forma irredutível, esse racional será periódico na base b se algum dos fatores primos que compõem o denominador j não for fator primo da base b**
  - Ou seja, frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 e 5 são dízimas periódicas
  - Frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 são "binários periódicos"

Números de ponto flutuante e representação binária

- **Dado um racional expresso como $\frac{i}{j}$ em sua forma irredutível, esse racional será periódico na base b se algum dos fatores primos que compõem o denominador j não for fator primo da base b**
  - Ou seja, frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 e 5 são dízimas periódicas
  - Frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 são "binários periódicos"
- $\frac{2}{3}$:

Números de ponto flutuante e representação binária

- **Dado um racional expresso como $\frac{i}{j}$ em sua forma irredutível, esse racional será periódico na base b se algum dos fatores primos que compõem o denominador j não for fator primo da base b**
  - Ou seja, frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 e 5 são dízimas periódicas
  - Frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 são "binários periódicos"
- **$\frac{2}{3}$: 10 não é múltiplo de 3 $\longrightarrow$ é dízima periódica**

Números de ponto flutuante e representação binária

- **Dado um racional expresso como $\frac{i}{j}$ em sua forma irredutível, esse racional será periódico na base b se algum dos fatores primos que compõem o denominador j não for fator primo da base b**
  - Ou seja, frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 e 5 são dízimas periódicas
  - Frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 são "binários periódicos"
- $\frac{2}{3}$ **: 10 não é múltiplo de 3** $\longrightarrow$ **é dízima periódica**
- $\frac{1}{10} = \frac{1}{2 \times 5}$ **:**

Números de ponto flutuante e representação binária

- **Dado um racional expresso como $\frac{i}{j}$ em sua forma irredutível, esse racional será periódico na base b se algum dos fatores primos que compõem o denominador j não for fator primo da base b**
  - Ou seja, frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 e 5 são dízimas periódicas
  - Frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 são "binários periódicos"
- **$\frac{2}{3}$: 10 não é múltiplo de 3 $\longrightarrow$ é dízima periódica**
- **$\frac{1}{10} = \frac{1}{2\times5}$: 10 é múltiplo de 2 e de 5 $\longrightarrow$ não é dízima periódica**

Números de ponto flutuante e representação binária

- **Dado um racional expresso como $\frac{i}{j}$ em sua forma irredutível, esse racional será periódico na base b se algum dos fatores primos que compõem o denominador j não for fator primo da base b**
  - Ou seja, frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 e 5 são dízimas periódicas
  - Frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 são "binários periódicos"
- **$\frac{2}{3}$: 10 não é múltiplo de 3 $\longrightarrow$ é dízima periódica**
- **$\frac{1}{10} = \frac{1}{2\times5}$: 10 é múltiplo de 2 e de 5 $\longrightarrow$ não é dízima periódica**
- **MAS!**

Números de ponto flutuante e representação binária

- **Dado um racional expresso como $\frac{i}{j}$ em sua forma irredutível, esse racional será periódico na base b se algum dos fatores primos que compõem o denominador j não for fator primo da base b**
  - Ou seja, frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 e 5 são dízimas periódicas
  - Frações irredutíveis em que o denominador é múltiplo de um primo diferente de 2 são "binários periódicos"
- **$\frac{2}{3}$: 10 não é múltiplo de 3 $\longrightarrow$ é dízima periódica**
- **$\frac{1}{10} = \frac{1}{2\times5}$: 10 é múltiplo de 2 e de 5 $\longrightarrow$ não é dízima periódica**
- **MAS!**
- **2 não é múltiplo de 5 $\longrightarrow$ $\frac{1}{10}$ é "binário periódico"**

Computadores eletrônicos

- **Os computadores eletrônicos não são apenas um refinamento das calculadoras e outros dispositivos mecânicos**
- **Três ideias "novas" estão na base dessa diferença:**
  - Aritmética de base 2 (operações numéricas)
  - Álgebra booleana (operações lógicas)
  - Corrente elétrica

Álgebra booleana

- **Apenas dois valores**
  - True/1
  - False/0

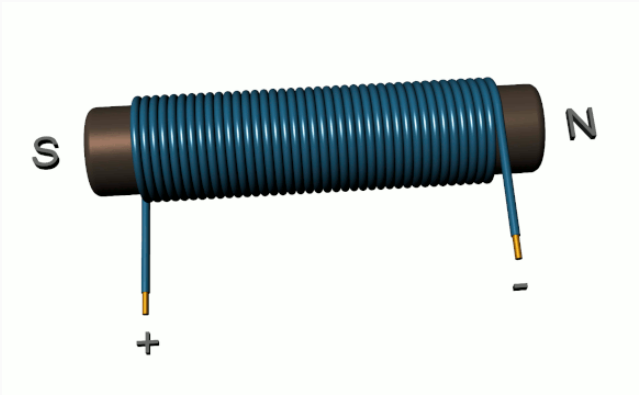
Álgebra booleana

- **Apenas dois valores**
  - True/1
  - False/0
- **Operações lógicas**
  - Conjunção (`AND`)
  - Disjunção (`OR`)
  - Negação (`NOT`)

Os matemático pira! 😜

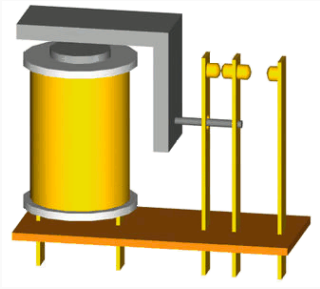
Computadores eletrônicos

- **Os computadores eletrônicos não são apenas um refinamento das calculadoras e outros dispositivos mecânicos**
- **Três ideias "novas" estão na base dessa diferença:**
  - Aritmética de base 2 (operações numéricas)
  - Álgebra booleana (operações lógicas)
  - Corrente elétrica

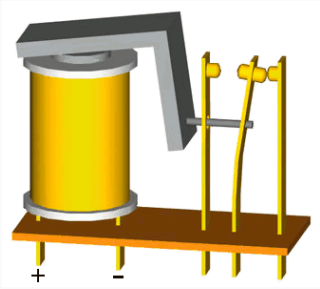


commons.wikimedia.org/wiki/File:Simple_electromagnet2.gif

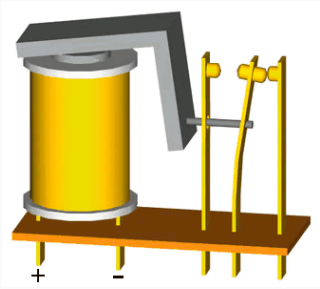
commons.wikimedia.org/wiki/File:Light_Switch.jpg

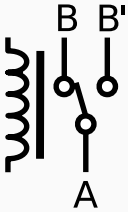


commons.wikimedia.org/wiki/File:Relay_principle_horizontal_new.gif



commons.wikimedia.org/wiki/File:Relay_principle_horizontal_new.gif





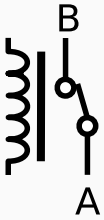
commons.wikimedia.org/wiki/File:Relay_principle_horizontal_new.gif

Operações lógicas com relês

NOT:

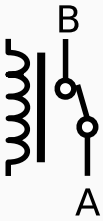
Operações lógicas com relês

NOT:



Operações lógicas com relês

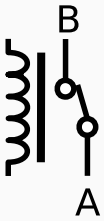
NOT:



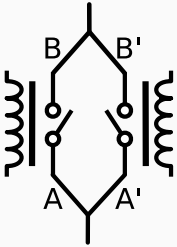
OR:

Operações lógicas com relês

NOT:

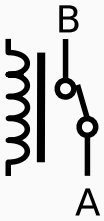


OR:

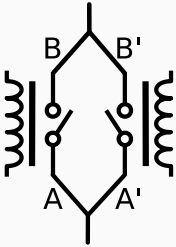


Operações lógicas com relês

NOT:



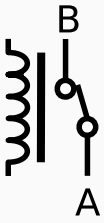
OR:



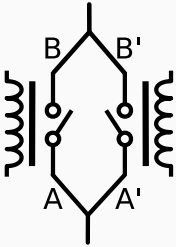
AND:

Operações lógicas com relês

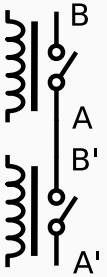
NOT:



OR:



AND:



Portas lógicas

NOT:

Portas lógicas

NOT: 

Portas lógicas

NOT: 

OR:

Portas lógicas

NOT: 

OR: 

Portas lógicas

NOT: 

OR: 

AND:

Portas lógicas

NOT: 

OR: 

AND: 

Portas lógicas

NOT: 

OR: 

AND: 

XOR:

Portas lógicas

NOT: 

OR: 

AND: 

XOR: 

Portas lógicas

NOT: 

OR: 

AND: 

XOR: 

NOR:

Portas lógicas

NOT: 

OR: 

AND: 

XOR: 

NOR: 

Portas lógicas

NOT: 

OR: 

AND: 

XOR: 

NOR: 

NAND:

Portas lógicas

NOT: 

OR: 

AND: 

XOR: 

NOR: 

NAND: 

Portas lógicas

- **Combinando portas lógicas, podemos implementar operações lógicas complexas**

Portas lógicas

- Combinando portas lógicas, podemos implementar operações lógicas complexas
- Podemos também implementar as operações numéricas (tomando cuidado especial com o “vai um”)

Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0$

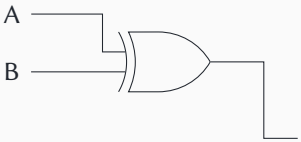
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

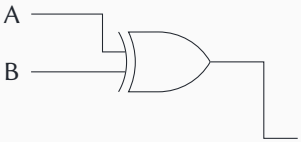
$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0$



Somando dígitos na base 2 com portas lógicas



$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0 \longrightarrow$ “vai um”

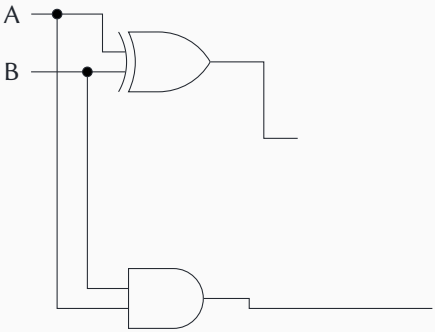
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0 \longrightarrow$ “vai um”



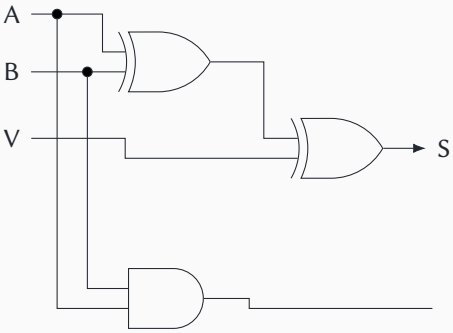
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0 \longrightarrow$ “vai um”



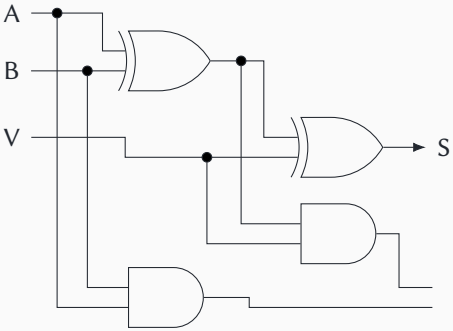
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0 \longrightarrow$ “vai um”



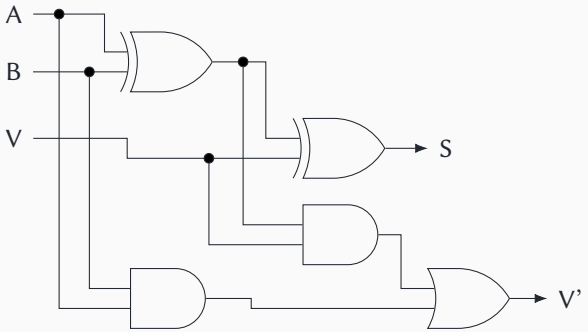
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0 \longrightarrow$ “vai um”



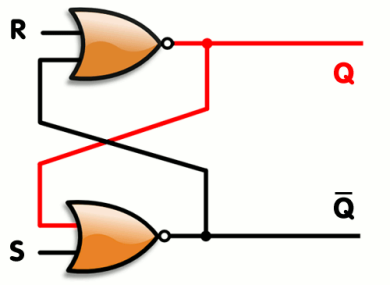
Portas lógicas

- Combinando portas lógicas, podemos implementar operações lógicas complexas
- Podemos também implementar as operações numéricas (tomando cuidado especial com o “vai um”)

Portas lógicas

- Combinando portas lógicas, podemos implementar operações lógicas complexas
- Podemos também implementar as operações numéricas (tomando cuidado especial com o “vai um”)
- E também podemos implementar memórias

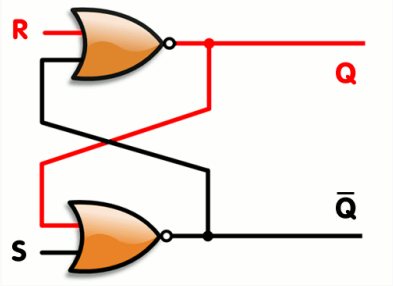
flip-flop



Estável — 1/True

commons.wikimedia.org/wiki/File:R-S_mk2.gif

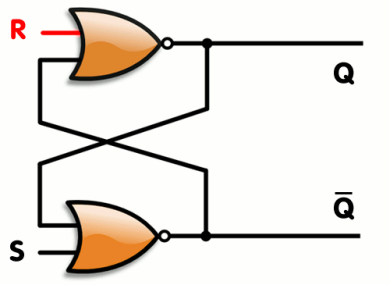
flip-flop



Reset — passando de 1/True para 0/False

commons.wikimedia.org/wiki/File:R-S_mk2.gif

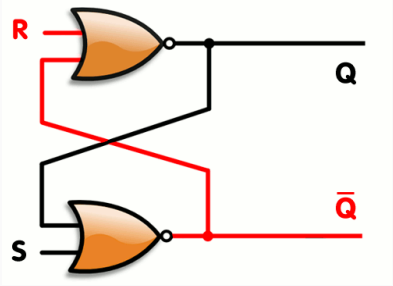
flip-flop



Reset — passando de 1/True para 0/False

commons.wikimedia.org/wiki/File:R-S_mk2.gif

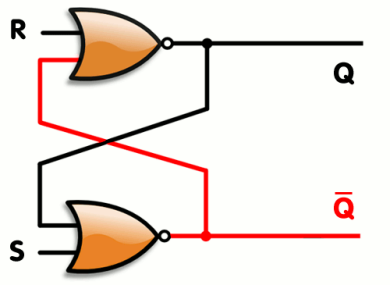
flip-flop



Reset — passando de 1/True para 0/False

commons.wikimedia.org/wiki/File:R-S_mk2.gif

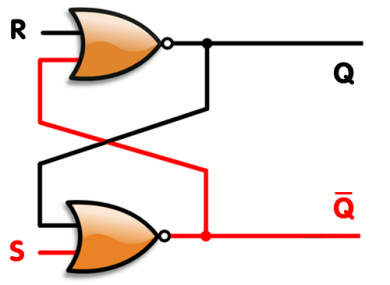
flip-flop



Estável — 0/False

commons.wikimedia.org/wiki/File:R-S_mk2.gif

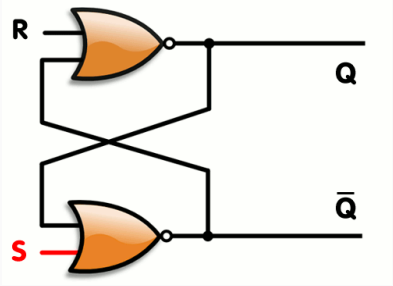
flip-flop



Set — passando de 0/False para 1/True

commons.wikimedia.org/wiki/File:R-S_mk2.gif

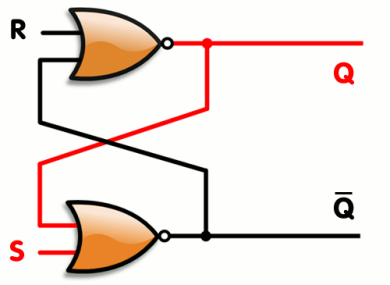
flip-flop



Set — passando de 0/False para 1/True

commons.wikimedia.org/wiki/File:R-S_mk2.gif

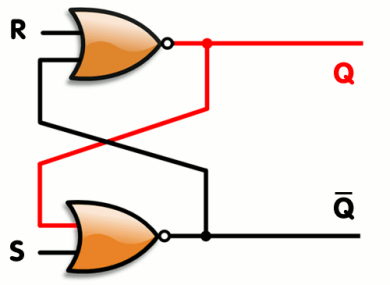
flip-flop



Set — passando de 0/False para 1/True

commons.wikimedia.org/wiki/File:R-S_mk2.gif

flip-flop



Estável — 1/True

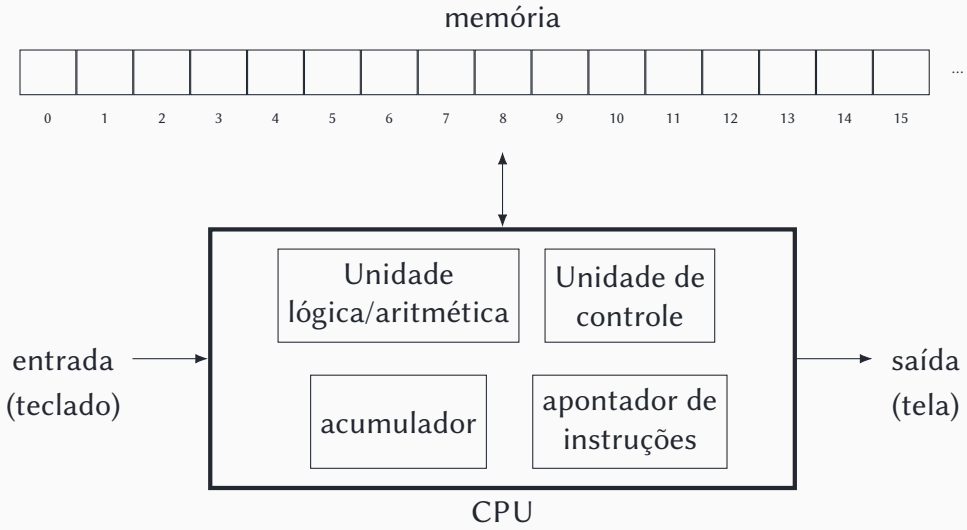
commons.wikimedia.org/wiki/File:R-S_mk2.gif

Mínimo

Recursos essenciais do computador que vimos até agora:

- **Sequência de instruções (programa) definida pelo usuário**
- **Leitura e escrita de dados (teclado e tela)**
- **Operações matemáticas (soma, divisão etc.)**
- **Números inteiros**
- **Referências a dados na memória**
- **Condicionais e laços (caso especial de condicionais)**

O computador HIPO



O computador HIPO

11	LDA	Load from memory to accumulator
12	STA	Store in memory from accumulator
21	ADD	Add memory and accumulator (result is left in accumulator)
22	SUB	Same, but subtraction
23	MUL	Same, but multiplication
24	DIV	Same, but division
25	REM	Same, but remainder
29	REV	Revert sign (+/-) of the accumulator
31	INN	Load from keyboard to memory
41	PRN	Show from memory to screen

50	NOP	Do nothing
51	JMP	Jump to given address
55	JEQ	Same, but only if the accumulator is zero
53	JDZ	Same, if not zero
57	JGE	Same, if zero or more
54	JGT	Same, if greater than zero
52	JLE	Same, if zero or less
56	JLT	Same, if less than zero
70	STP	Stop

Instruções reconhecidas pelo computador HIPO

Computadores eletromecânicos

- **No início do século XX, alguns dispositivos baseados em relês foram de fato construídos**

Computadores eletromecânicos

- **No início do século XX, alguns dispositivos baseados em relês foram de fato construídos**
  - O Z3 de Konrad Zuse usava o sistema binário, era programável com cartões perfurados e era “quase Turing-completo”, mas não foi valorizado durante a guerra

Computadores eletromecânicos

- **No início do século XX, alguns dispositivos baseados em relês foram de fato construídos**
  - O Z3 de Konrad Zuse usava o sistema binário, era programável com cartões perfurados e era “quase Turing-completo”, mas não foi valorizado durante a guerra
  - Harvard Mark I, similar e influenciado por Babbage, mas efetivamente usado para o cálculo de tabelas matemáticas e cálculos relacionados à bomba atômica

Computadores eletromecânicos

- **No início do século XX, alguns dispositivos baseados em relês foram de fato construídos**
  - O Z3 de Konrad Zuse usava o sistema binário, era programável com cartões perfurados e era “quase Turing-completo”, mas não foi valorizado durante a guerra
  - Harvard Mark I, similar e influenciado por Babbage, mas efetivamente usado para o cálculo de tabelas matemáticas e cálculos relacionados à bomba atômica
- **Nenhum deles era de fato Turing-completo**

Computadores eletromecânicos

- **No início do século XX, alguns dispositivos baseados em relês foram de fato construídos**
  - O Z3 de Konrad Zuse usava o sistema binário, era programável com cartões perfurados e era "quase Turing-completo", mas não foi valorizado durante a guerra
  - Harvard Mark I, similar e influenciado por Babbage, mas efetivamente usado para o cálculo de tabelas matemáticas e cálculos relacionados à bomba atômica
- **Nenhum deles era de fato Turing-completo**
- **Os computadores eletrônicos (Colossus, ENIAC) tornaram esses sistemas obsoletos quase imediatamente**

Passo

Sistemas analógicos

- **Em uma balança, relógio ou régua de cálculo, o movimento das partes vai indicando valores cada vez mais próximos do resultado correto**

Sistemas analógicos

- **Em uma balança, relógio ou régua de cálculo, o movimento das partes vai indicando valores cada vez mais próximos do resultado correto**
  - Escala contínua de valores ⟶ sistemas analógicos

Sistemas analógicos

- **Em uma balança, relógio ou régua de cálculo, o movimento das partes vai indicando valores cada vez mais próximos do resultado correto**
  - Escala contínua de valores $\longrightarrow$ sistemas analógicos
- **Em um ábaco não!**

Sistemas analógicos

- **Em uma balança, relógio ou régua de cálculo, o movimento das partes vai indicando valores cada vez mais próximos do resultado correto**
  - Escala contínua de valores ⟶ sistemas analógicos
- **Em um ábaco não!**
  - Durante o movimento de cada peça, o estado do ábaco não faz sentido (a peça não está nem de um lado nem do outro)

Sistemas analógicos

- **Em uma balança, relógio ou régua de cálculo, o movimento das partes vai indicando valores cada vez mais próximos do resultado correto**
  - Escala contínua de valores ⟶ sistemas analógicos
- **Em um ábaco não!**
  - Durante o movimento de cada peça, o estado do ábaco não faz sentido (a peça não está nem de um lado nem do outro)
- **Ao fazer uma soma no papel também não!**

Sistemas analógicos

- **Em uma balança, relógio ou régua de cálculo, o movimento das partes vai indicando valores cada vez mais próximos do resultado correto**
  - Escala contínua de valores ⟶ sistemas analógicos
- **Em um ábaco não!**
  - Durante o movimento de cada peça, o estado do ábaco não faz sentido (a peça não está nem de um lado nem do outro)
- **Ao fazer uma soma no papel também não!**
  - Enquanto os dígitos estão sendo desenhados, o estado da expressão no papel não faz sentido

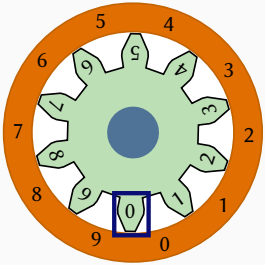
Sistemas analógicos

- **Em uma balança, relógio ou régua de cálculo, o movimento das partes vai indicando valores cada vez mais próximos do resultado correto**
  - Escala contínua de valores ⟶ sistemas analógicos
- **Em um ábaco não!**
  - Durante o movimento de cada peça, o estado do ábaco não faz sentido (a peça não está nem de um lado nem do outro)
- **Ao fazer uma soma no papel também não!**
  - Enquanto os dígitos estão sendo desenhados, o estado da expressão no papel não faz sentido
- **O processamento é feito em passos descontínuos**

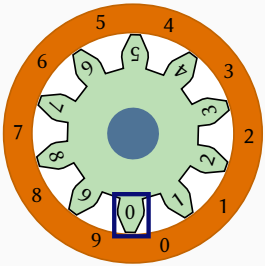
Sistemas analógicos

- **Em uma balança, relógio ou régua de cálculo, o movimento das partes vai indicando valores cada vez mais próximos do resultado correto**
  - Escala contínua de valores ⟶ sistemas analógicos
- **Em um ábaco não!**
  - Durante o movimento de cada peça, o estado do ábaco não faz sentido (a peça não está nem de um lado nem do outro)
- **Ao fazer uma soma no papel também não!**
  - Enquanto os dígitos estão sendo desenhados, o estado da expressão no papel não faz sentido
- **O processamento é feito em passos descontínuos**
  - sistemas digitais

Pascalina passo a passo

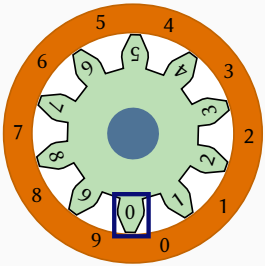


Pascalina passo a passo



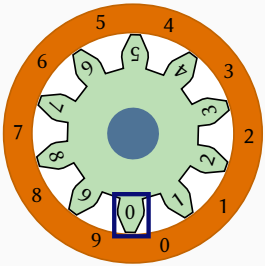
- Cada vez que movemos a engrenagem, realizamos um passo

Pascalina passo a passo



- **Cada vez que movemos a engrenagem, realizamos um passo**
  - Enquanto a engrenagem está em movimento (ou seja, durante a execução do passo), o estado da pascalina não faz sentido

Pascalina passo a passo



- **Cada vez que movemos a engrenagem, realizamos um passo**
  - Enquanto a engrenagem está em movimento (ou seja, durante a execução do passo), o estado da pascalina não faz sentido
- **Cada passo $\longrightarrow$ uma operação de um dígito**

Passo em calculadoras mecânicas

- **Calculadoras mecânicas posteriores (e a máquina analítica) processavam todos os dígitos em um único passo**

Passo em calculadoras mecânicas

- **Calculadoras mecânicas posteriores (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Geralmente, um giro completo de uma manivela

Passo em calculadoras mecânicas

- **Calculadoras mecânicas posteriores (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Geralmente, um giro completo de uma manivela
  - Evidentemente, muito mais rápido

Passo em calculadoras mecânicas

- **Calculadoras mecânicas posteriores (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Geralmente, um giro completo de uma manivela
  - Evidentemente, muito mais rápido
- **Existe uma velocidade máxima para a execução dos passos**

Passo em calculadoras mecânicas

- **Calculadoras mecânicas posteriores (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Geralmente, um giro completo de uma manivela
  - Evidentemente, muito mais rápido
- **Existe uma velocidade máxima para a execução dos passos**
  - Potência necessária para girar a manivela

Passo em calculadoras mecânicas

- **Calculadoras mecânicas posteriores (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Geralmente, um giro completo de uma manivela
  - Evidentemente, muito mais rápido
- **Existe uma velocidade máxima para a execução dos passos**
  - Potência necessária para girar a manivela
  - Resistência das engrenagens ao esforço mecânico

Passo em calculadoras mecânicas

- **Calculadoras mecânicas posteriores (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Geralmente, um giro completo de uma manivela
  - Evidentemente, muito mais rápido
- **Existe uma velocidade máxima para a execução dos passos**
  - Potência necessária para girar a manivela
  - Resistência das engrenagens ao esforço mecânico
  - ...

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também precisam de passos**

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também precisam de passos**
  - O estado do sistema não faz sentido enquanto algum relê está mudando de estado

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também precisam de passos**
  - O estado do sistema não faz sentido enquanto algum relê está mudando de estado
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica se move muito rapidamente, mas não é instantânea)*

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também precisam de passos**
  - O estado do sistema não faz sentido enquanto algum relê está mudando de estado
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica se move muito rapidamente, mas não é instantânea)*
  - Só faz sentido "ler" um sinal de entrada depois que o sinal anterior foi processado até o fim (pois o resultado pode fazer parte da entrada do próximo passo)

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também precisam de passos**
  - O estado do sistema não faz sentido enquanto algum relê está mudando de estado
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica se move muito rapidamente, mas não é instantânea)*
  - Só faz sentido "ler" um sinal de entrada depois que o sinal anterior foi processado até o fim (pois o resultado pode fazer parte da entrada do próximo passo)
  - Passos mais rápidos consomem mais energia e geram mais calor

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também precisam de passos**
  - O estado do sistema não faz sentido enquanto algum relê está mudando de estado
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica se move muito rapidamente, mas não é instantânea)*
  - Só faz sentido "ler" um sinal de entrada depois que o sinal anterior foi processado até o fim (pois o resultado pode fazer parte da entrada do próximo passo)
  - Passos mais rápidos consomem mais energia e geram mais calor
- **existe um "metrônomo" (*clock*) que controla a velocidade dos passos**

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também precisam de passos**
  - O estado do sistema não faz sentido enquanto algum relê está mudando de estado
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica se move muito rapidamente, mas não é instantânea)*
  - Só faz sentido "ler" um sinal de entrada depois que o sinal anterior foi processado até o fim (pois o resultado pode fazer parte da entrada do próximo passo)
  - Passos mais rápidos consomem mais energia e geram mais calor
- **existe um "metrônomo" (*clock*) que controla a velocidade dos passos**
  - "Meu computador é de 3.2GHz"

Acumulador

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível
  - Podemos escolher a ordem em que vamos realizar os passos

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível
  - Podemos escolher a ordem em que vamos realizar os passos
    - » *Porque a expressão completa está sempre visível*

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível
  - Podemos escolher a ordem em que vamos realizar os passos
    - » *Porque a expressão completa está sempre visível*

$$\frac{2 \times (3 + 4) - 5 + 6 - 7}{8}$$

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível
  - Podemos escolher a ordem em que vamos realizar os passos
    - » *Porque a expressão completa está sempre visível*

$$\frac{2 \times (3 + 4) - 5 + 6 - 7}{8}$$

$$\frac{2 \times (3 + 4) + 1 - 7}{8}$$

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível
  - Podemos escolher a ordem em que vamos realizar os passos
    - » *Porque a expressão completa está sempre visível*

$$\frac{2 \times (3 + 4) - 5 + 6 - 7}{8}$$

$$\frac{2 \times (3 + 4) + 1 - 7}{8}$$

$$\frac{2 \times 7 + 1 - 7}{8}$$

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível
  - Podemos escolher a ordem em que vamos realizar os passos
    - » *Porque a expressão completa está sempre visível*

$$\frac{2 \times (3 + 4) - 5 + 6 - 7}{8}$$

$$\frac{2 \times (3 + 4) + 1 - 7}{8}$$

$$\frac{2 \times 7 + 1 - 7}{8}$$

$$\frac{2 \times 7 - 6}{8}$$

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível
  - Podemos escolher a ordem em que vamos realizar os passos
    - » *Porque a expressão completa está sempre visível*

$$\frac{2 \times (3 + 4) - 5 + 6 - 7}{8}$$

$$\frac{2 \times (3 + 4) + 1 - 7}{8}$$

$$\frac{2 \times 7 + 1 - 7}{8}$$

$$\frac{2 \times 7 - 6}{8}$$

$$\frac{14 - 6}{8}$$

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível
  - Podemos escolher a ordem em que vamos realizar os passos
    - » *Porque a expressão completa está sempre visível*

$$\frac{2 \times (3 + 4) - 5 + 6 - 7}{8}$$

$$\frac{2 \times (3 + 4) + 1 - 7}{8}$$

$$\frac{2 \times 7 + 1 - 7}{8}$$

$$\frac{2 \times 7 - 6}{8}$$

$$\frac{14 - 6}{8}$$

$$\frac{7 - 3}{4}$$

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível
  - Podemos escolher a ordem em que vamos realizar os passos
    - » *Porque a expressão completa está sempre visível*

$$\frac{2 \times (3 + 4) - 5 + 6 - 7}{8}$$

$$\frac{2 \times (3 + 4) + 1 - 7}{8}$$

$$\frac{2 \times 7 + 1 - 7}{8}$$

$$\frac{2 \times 7 - 6}{8}$$

$$\frac{14 - 6}{8}$$

$$\frac{7 - 3}{4}$$

$$\frac{4}{4}$$

O acumulador

- **Para resolver uma conta no papel, fazemos "afirmações", como** $1 + 2 - 3 + 4$
- **Reescrevemos a afirmação conforme vamos realizando as diversas operações**
  - Ou seja, não é necessário usar borracha
  - O "histórico" dos passos é visível
  - Podemos escolher a ordem em que vamos realizar os passos
    - » *Porque a expressão completa está sempre visível*

$$\frac{2 \times (3 + 4) - 5 + 6 - 7}{8}$$

$$\frac{2 \times (3 + 4) + 1 - 7}{8}$$

$$\frac{2 \times 7 + 1 - 7}{8}$$

$$\frac{2 \times 7 - 6}{8}$$

$$\frac{14 - 6}{8}$$

$$\frac{7 - 3}{4}$$

$$\frac{4}{4}$$

$$1$$

O acumulador

- Mas não é assim que fazemos contas "de cabeça"

O acumulador

- **Mas não é assim que fazemos contas "de cabeça"**
  - Executamos os passos sequencialmente

O acumulador

- **Mas não é assim que fazemos contas "de cabeça"**
  - Executamos os passos sequencialmente
  - A cada passo, guardamos o resultado na memória e processamos esse resultado com o próximo operando

O acumulador

- **Mas não é assim que fazemos contas "de cabeça"**
  - Executamos os passos sequencialmente
  - A cada passo, guardamos o resultado na memória e processamos esse resultado com o próximo operando
    - » *Ou seja, temos sempre um resultado intermediário*

O acumulador

- **Mas não é assim que fazemos contas "de cabeça"**
  - Executamos os passos sequencialmente
  - A cada passo, guardamos o resultado na memória e processamos esse resultado com o próximo operando
    - » *Ou seja, temos sempre um resultado intermediário*
    - » *Esse resultado intermediário é um dos operandos da próxima operação*

O acumulador

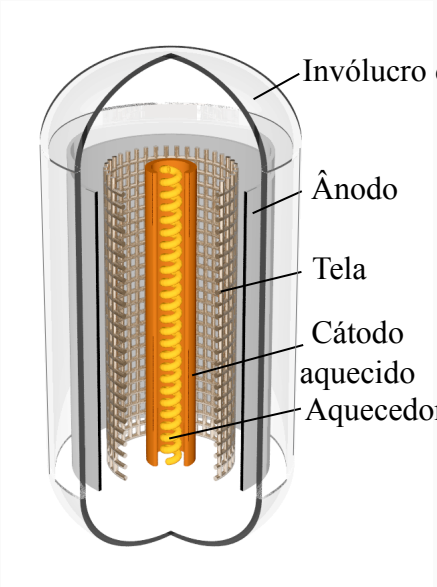
- **Mas não é assim que fazemos contas "de cabeça"**
  - Executamos os passos sequencialmente
  - A cada passo, guardamos o resultado na memória e processamos esse resultado com o próximo operando
    - » *Ou seja, temos sempre um resultado intermediário*
    - » *Esse resultado intermediário é um dos operandos da próxima operação*
    - » *Ele é "jogado fora" e substituído por um novo resultado intermediário a cada passo (no papel, usaríamos a borracha)*

O acumulador

- **Mas não é assim que fazemos contas "de cabeça"**
  - Executamos os passos sequencialmente
  - A cada passo, guardamos o resultado na memória e processamos esse resultado com o próximo operando
    - » *Ou seja, temos sempre um resultado intermediário*
    - » *Esse resultado intermediário é um dos operandos da próxima operação*
    - » *Ele é "jogado fora" e substituído por um novo resultado intermediário a cada passo (no papel, usaríamos a borracha)*
    - » *Após o último passo, ele é o resultado final*

O acumulador

- **Mas não é assim que fazemos contas "de cabeça"**
  - Executamos os passos sequencialmente
  - A cada passo, guardamos o resultado na memória e processamos esse resultado com o próximo operando
    - » *Ou seja, temos sempre um resultado intermediário*
    - » *Esse resultado intermediário é um dos operandos da próxima operação*
    - » *Ele é "jogado fora" e substituído por um novo resultado intermediário a cada passo (no papel, usaríamos a borracha)*
    - » *Após o último passo, ele é o resultado final*
- **O mesmo acontece com ábacos, calculadoras de bolso... e o computador!**



commons.wikimedia.org/wiki/File:Triode-english-text.svg

Computadores eletrônicos — válvulas

- Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis

Computadores eletrônicos — válvulas

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Mais rápidas que relês

Computadores eletrônicos — válvulas

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Mais rápidas que relês
  - Consumo muito grande de energia

Computadores eletrônicos — válvulas

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Mais rápidas que relês
  - Consumo muito grande de energia
  - Frágeis e “queimam” com facilidade

Computadores eletrônicos — válvulas

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Mais rápidas que relês
  - Consumo muito grande de energia
  - Frágeis e “queimam” com facilidade
- **ENIAC, o primeiro computador Turing-completo (1946)**

Computadores eletrônicos — válvulas

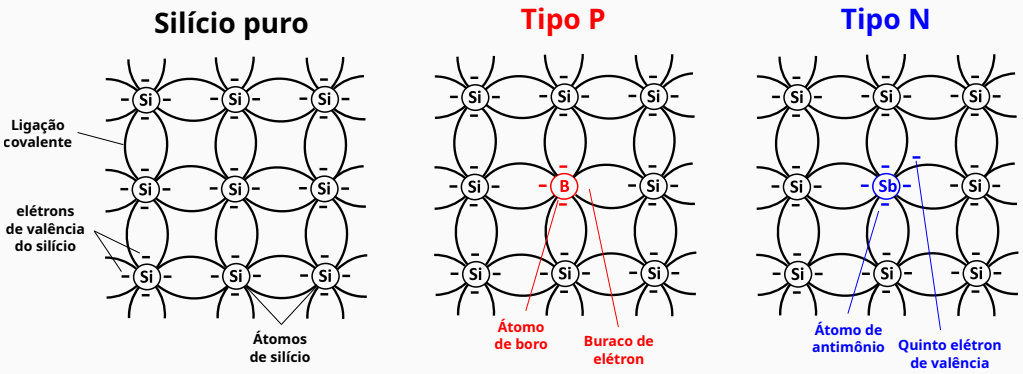
- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Mais rápidas que relês
  - Consumo muito grande de energia
  - Frágeis e “queimam” com facilidade
- **ENIAC, o primeiro computador Turing-completo (1946)**
  - “Programá-lo” envolvia alterar as conexões elétricas do circuito

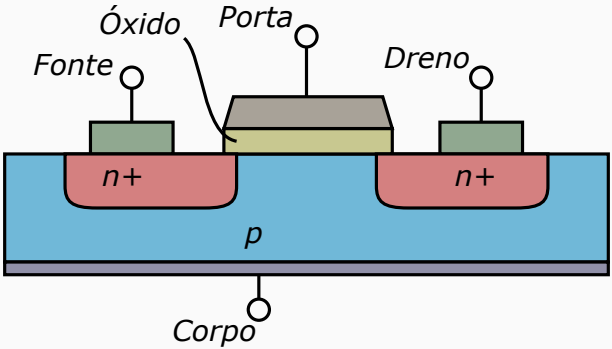
Computadores eletrônicos — válvulas

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Mais rápidas que relês
  - Consumo muito grande de energia
  - Frágeis e "queimam" com facilidade
- **ENIAC, o primeiro computador Turing-completo (1946)**
  - "Programá-lo" envolvia alterar as conexões elétricas do circuito
- **Ferranti Mark 1 (1951, Reino Unido), com 4.050 válvulas, pesava 4,5 toneladas e consumia 26kW(?)**

Computadores eletrônicos — válvulas

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Mais rápidas que relês
  - Consumo muito grande de energia
  - Frágeis e "queimam" com facilidade
- **ENIAC, o primeiro computador Turing-completo (1946)**
  - "Programá-lo" envolvia alterar as conexões elétricas do circuito
- **Ferranti Mark 1 (1951, Reino Unido), com 4.050 válvulas, pesava 4,5 toneladas e consumia 26kW(?)**
- **IBM 650 (1954), pesava 2,4–2,8 toneladas e consumia 23–50kW, foi um "sucesso comercial" (quase 2.000 unidades vendidas em dez anos)**


Silício puro
Tipo P
Tipo N
Ligação covalente
elétrons de valência do silício
Átomos de silício
Átomo de boro
Buraco de elétron
Átomo de antimônio
Quinto elétron de valência
Si
B
Sb


Óxido
Porta
Fonte
Dreno
n+
n+
p
Corpo

Computadores eletrônicos — transístores e CIs

- O transístor (1947) também pode realizar as mesmas operações, mas é ainda mais rápido, consome muito menos energia, é bem mais durável e muito menor

Computadores eletrônicos — transístores e CIs

- **O transístor (1947) também pode realizar as mesmas operações, mas é ainda mais rápido, consome muito menos energia, é bem mais durável e muito menor**
  - IBM 7070 (1958, pesava 10 toneladas), NCR 315 (1962, pesava 601Kg) e outros

Computadores eletrônicos — transístores e CIs

- **O transístor (1947) também pode realizar as mesmas operações, mas é ainda mais rápido, consome muito menos energia, é bem mais durável e muito menor**
  - IBM 7070 (1958, pesava 10 toneladas), NCR 315 (1962, pesava 601Kg) e outros
- **Fotolitografia $\longrightarrow$ é possível construir circuitos com vários transístores de uma vez**

Computadores eletrônicos — transístores e CIs

- **O transístor (1947) também pode realizar as mesmas operações, mas é ainda mais rápido, consome muito menos energia, é bem mais durável e muito menor**
  - IBM 7070 (1958, pesava 10 toneladas), NCR 315 (1962, pesava 601Kg) e outros
- **Fotolitografia $\longrightarrow$ é possível construir circuitos com vários transístores de uma vez**
  - Conjuntos de circuitos integrados (PDP-11, 1970)

Computadores eletrônicos — transístores e CIs

- **O transístor (1947) também pode realizar as mesmas operações, mas é ainda mais rápido, consome muito menos energia, é bem mais durável e muito menor**
  - IBM 7070 (1958, pesava 10 toneladas), NCR 315 (1962, pesava 601Kg) e outros
- **Fotolitografia $\longrightarrow$ é possível construir circuitos com vários transístores de uma vez**
  - Conjuntos de circuitos integrados (PDP-11, 1970)
  - Microprocessadores (Intel 4004, 1971)

Computadores eletrônicos — transístores e CIs

- **O transístor (1947) também pode realizar as mesmas operações, mas é ainda mais rápido, consome muito menos energia, é bem mais durável e muito menor**
  - IBM 7070 (1958, pesava 10 toneladas), NCR 315 (1962, pesava 601Kg) e outros
- **Fotolitografia → é possível construir circuitos com vários transístores de uma vez**
  - Conjuntos de circuitos integrados (PDP-11, 1970)
  - Microprocessadores (Intel 4004, 1971)
    - » *Início dos "computadores pessoais" e videogames*

Computadores eletrônicos — transístores e CIs

- **O transístor (1947) também pode realizar as mesmas operações, mas é ainda mais rápido, consome muito menos energia, é bem mais durável e muito menor**
  - IBM 7070 (1958, pesava 10 toneladas), NCR 315 (1962, pesava 601Kg) e outros
- **Fotolitografia ⟶ é possível construir circuitos com vários transístores de uma vez**
  - Conjuntos de circuitos integrados (PDP-11, 1970)
  - Microprocessadores (Intel 4004, 1971)
    - » *Início dos "computadores pessoais" e videogames*
    - » *Estamos aqui* 🙂

Intermezzo

Como fazer computadores mais rápidos?

- Como fazer computadores mais rápidos?

Como fazer computadores mais rápidos?

- **Como fazer computadores mais rápidos?**
  1. Fazer operações complexas em um único passo

Como fazer computadores mais rápidos?

- **Como fazer computadores mais rápidos?**
  1. Fazer operações complexas em um único passo
  2. Aumentar a velocidade dos passos (*clock*)

Como fazer computadores mais rápidos?

- **Como fazer computadores mais rápidos?**
  1. Fazer operações complexas em um único passo
  2. Aumentar a velocidade dos passos (*clock*)
  3. Fazer várias coisas ao mesmo tempo

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...
  - Podemos realizar multiplicações como uma série de adições...

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...
  - Podemos realizar multiplicações como uma série de adições...
  - Ou podemos criar um conjunto de portas lógicas que executem multiplicações em um único passo

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...
  - Podemos realizar multiplicações como uma série de adições...
  - Ou podemos criar um conjunto de portas lógicas que executem multiplicações em um único passo
  - Idem para funções de ponto flutuante, processamento de sinais (FFT), criptografia, codificação/decodificação de mídia...

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...
  - Podemos realizar multiplicações como uma série de adições...
  - Ou podemos criar um conjunto de portas lógicas que executem multiplicações em um único passo
  - Idem para funções de ponto flutuante, processamento de sinais (FFT), criptografia, codificação/decodificação de mídia...
  - Processadores especializados

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...
  - Podemos realizar multiplicações como uma série de adições...
  - Ou podemos criar um conjunto de portas lógicas que executem multiplicações em um único passo
  - Idem para funções de ponto flutuante, processamento de sinais (FFT), criptografia, codificação/decodificação de mídia...
  - Processadores especializados
    - » *Yamaha DX7*

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...
  - Podemos realizar multiplicações como uma série de adições...
  - Ou podemos criar um conjunto de portas lógicas que executem multiplicações em um único passo
  - Idem para funções de ponto flutuante, processamento de sinais (FFT), criptografia, codificação/decodificação de mídia...
  - Processadores especializados
    - » *Yamaha DX7*
  - Coprocessadores dedicados

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...
  - Podemos realizar multiplicações como uma série de adições...
  - Ou podemos criar um conjunto de portas lógicas que executem multiplicações em um único passo
  - Idem para funções de ponto flutuante, processamento de sinais (FFT), criptografia, codificação/decodificação de mídia...
  - Processadores especializados
    - » *Yamaha DX7*
  - Coprocessadores dedicados
    - » *Digidesign Pro Tools*

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...
  - Podemos realizar multiplicações como uma série de adições...
  - Ou podemos criar um conjunto de portas lógicas que executem multiplicações em um único passo
  - Idem para funções de ponto flutuante, processamento de sinais (FFT), criptografia, codificação/decodificação de mídia...
  - Processadores especializados
    - » *Yamaha DX7*
  - Coprocessadores dedicados
    - » *Digidesign Pro Tools*
    - » *GPUs (SIMD)*

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...
  - Podemos realizar multiplicações como uma série de adições...
  - Ou podemos criar um conjunto de portas lógicas que executem multiplicações em um único passo
  - Idem para funções de ponto flutuante, processamento de sinais (FFT), criptografia, codificação/decodificação de mídia...
  - Processadores especializados
    - » *Yamaha DX7*
  - Coprocessadores dedicados
    - » *Digidesign Pro Tools*
    - » *GPUs (SIMD)*
    - » *Apple Silicon (M1/M2/M3)*

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 8 bits, 16 bits, 32 bits, 64 bits...
  - Podemos realizar multiplicações como uma série de adições...
  - Ou podemos criar um conjunto de portas lógicas que executem multiplicações em um único passo
  - Idem para funções de ponto flutuante, processamento de sinais (FFT), criptografia, codificação/decodificação de mídia...
  - Processadores especializados
    - » *Yamaha DX7*
  - Coprocessadores dedicados
    - » *Digidesign Pro Tools*
    - » *GPUs (SIMD)*
    - » *Apple Silicon (M1/M2/M3)*
    - » *FPGAs ("hardware programável")*

O processador 8088 (1979), usado nos primeiros PCs da IBM, tinha 29.000 transístores

O processador 8088 (1979), usado nos primeiros PCs da IBM, tinha 29.000 transístores

O Athlon 64 (2007) tinha 122 milhões

O processador 8088 (1979), usado nos primeiros PCs da IBM, tinha 29.000 transístores

O Athlon 64 (2007) tinha 122 milhões

Processadores atuais têm bilhões

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos (*clock*)**

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos (*clock*)**
  - Um tanto óbvio 😛

Como fazer computadores mais rápidos?

- **Fazer várias coisas ao mesmo tempo**

Como fazer computadores mais rápidos?

- **Fazer várias coisas ao mesmo tempo**
  - Vários processadores (genéricos ou especializados)

Como fazer computadores mais rápidos?

- **Fazer várias coisas ao mesmo tempo**
  - Vários processadores (genéricos ou especializados)
    - » *Em servidores, podem atender vários usuários*

Como fazer computadores mais rápidos?

- **Fazer várias coisas ao mesmo tempo**
  - Vários processadores (genéricos ou especializados)
    - » *Em servidores, podem atender vários usuários*
    - » *Em computadores pessoais, divisão de tarefas; por exemplo, cada uma das imagens em uma página web pode ser renderizada por um processador*

Como fazer computadores mais rápidos?

- **Fazer várias coisas ao mesmo tempo**
  - Vários processadores (genéricos ou especializados)
    - » *Em servidores, podem atender vários usuários*
    - » *Em computadores pessoais, divisão de tarefas; por exemplo, cada uma das imagens em uma página web pode ser renderizada por um processador*
    - » *GPUs para cálculos vetoriais (SIMD)*

Como fazer computadores mais rápidos?

- **Fazer várias coisas ao mesmo tempo**
  - Vários processadores (genéricos ou especializados)
    - » *Em servidores, podem atender vários usuários*
    - » *Em computadores pessoais, divisão de tarefas; por exemplo, cada uma das imagens em uma página web pode ser renderizada por um processador*
    - » *GPUs para cálculos vetoriais (SIMD)*
  - "Trenzinho" (*pipelining*)

Como fazer computadores mais rápidos?

- **Fazer várias coisas ao mesmo tempo**
  - Vários processadores (genéricos ou especializados)
    - » *Em servidores, podem atender vários usuários*
    - » *Em computadores pessoais, divisão de tarefas; por exemplo, cada uma das imagens em uma página web pode ser renderizada por um processador*
    - » *GPUs para cálculos vetoriais (SIMD)*
  - "Trenzinho" (*pipelining*)
  - Execução especulativa e fora de ordem

“Trenzinho” (*Pipelining*)

1. **Lê instrução da memória**

“Trenzinho” (*Pipelining*)

1. Lê instrução da memória
2. Lê operandos da memória

“Trenzinho” (*Pipelining*)

1. Lê instrução da memória
2. Lê operandos da memória
3. Executa instrução e coloca o resultado no acumulador

“Trenzinho” (*Pipelining*)

1. Lê instrução da memória
2. Lê operandos da memória
3. Executa instrução e coloca o resultado no acumulador
4. Grava o conteúdo do acumulador na memória

“Trenzinho” (*Pipelining*)

1. Lê instrução da memória
2. Lê operandos da memória
3. Executa instrução e coloca o resultado no acumulador
4. Grava o conteúdo do acumulador na memória

Os circuitos reponsáveis por cada etapa são diferentes, então eles ficam ociosos 75% do tempo!

“Trenzinho” (*Pipelining*)

1. Lê instrução da memória
2. Lê operandos da memória
3. Executa instrução e coloca o resultado no acumulador
4. Grava o conteúdo do acumulador na memória

Os circuitos reponsáveis por cada etapa são diferentes, então eles ficam ociosos 75% do tempo!

- Ao ler os operandos, lê também a próxima instrução

“Trenzinho” (*Pipelining*)

1. Lê instrução da memória
2. Lê operandos da memória
3. Executa instrução e coloca o resultado no acumulador
4. Grava o conteúdo do acumulador na memória

Os circuitos reponsáveis por cada etapa são diferentes, então eles ficam ociosos 75% do tempo!

- Ao ler os operandos, lê também a próxima instrução
- Ao executar a instrução, também lê os operandos da próxima E também lê a instrução seguinte

“Trenzinho” (*Pipelining*)

1. Lê instrução da memória
2. Lê operandos da memória
3. Executa instrução e coloca o resultado no acumulador
4. Grava o conteúdo do acumulador na memória

Os circuitos reponsáveis por cada etapa são diferentes, então eles ficam ociosos 75% do tempo!

- Ao ler os operandos, lê também a próxima instrução
- Ao executar a instrução, também lê os operandos da próxima E também lê a instrução seguinte
- ... e assim por diante

“Trenzinho” (*Pipelining*)

1. Lê instrução da memória
2. Lê operandos da memória
3. Executa instrução e coloca o resultado no acumulador
4. Grava o conteúdo do acumulador na memória

Os circuitos reponsáveis por cada etapa são diferentes, então eles ficam ociosos 75% do tempo!

- Ao ler os operandos, lê também a próxima instrução
- Ao executar a instrução, também lê os operandos da próxima E também lê a instrução seguinte
- ... e assim por diante
- Alguns processadores têm *pipelines* com mais de 20 etapas

Execução especulativa e fora de ordem

- E se tiver `if`/`else`?

Execução especulativa e fora de ordem

- **E se tiver `if`/`else`?**
  - tenta adivinhar qual “lado” provavelmente vai ser executado

Execução especulativa e fora de ordem

- **E se tiver `if`/`else`?**
  - tenta adivinhar qual "lado" provavelmente vai ser executado
    - » *se der certo, ótimo; se não, joga o resultado fora*

Execução especulativa e fora de ordem

- **E se tiver `if`/`else`?**
  - tenta adivinhar qual "lado" provavelmente vai ser executado
    - » *se der certo, ótimo; se não, joga o resultado fora*
- **A instrução *n* é uma soma**

Execução especulativa e fora de ordem

- **E se tiver `if`/`else`?**
  - tenta adivinhar qual "lado" provavelmente vai ser executado
    - » *se der certo, ótimo; se não, joga o resultado fora*
- **A instrução n é uma soma**
- **A instrução $n + 3$ é uma multiplicação**

Execução especulativa e fora de ordem

- **E se tiver `if`/`else`?**
  - tenta adivinhar qual "lado" provavelmente vai ser executado
    - » *se der certo, ótimo; se não, joga o resultado fora*
- **A instrução n é uma soma**
- **A instrução $n + 3$ é uma multiplicação**
- **O circuito que processa cada uma dessas instruções é diferente**

Execução especulativa e fora de ordem

- **E se tiver `if`/`else`?**
  - tenta adivinhar qual "lado" provavelmente vai ser executado
    - » *se der certo, ótimo; se não, joga o resultado fora*
- **A instrução n é uma soma**
- **A instrução $n + 3$ é uma multiplicação**
- **O circuito que processa cada uma dessas instruções é diferente**
- **Processa os dois ao mesmo tempo e guarda o segundo resultado para a hora certa**

Execução especulativa e fora de ordem

- **E se tiver `if`/`else`?**
  - tenta adivinhar qual "lado" provavelmente vai ser executado
    - » *se der certo, ótimo; se não, joga o resultado fora*
- **A instrução n é uma soma**
- **A instrução $n + 3$ é uma multiplicação**
- **O circuito que processa cada uma dessas instruções é diferente**
- **Processa os dois ao mesmo tempo e guarda o segundo resultado para a hora certa**
  - Novamente, se tiver **`if`/`else`** o resultado pode ser inútil

Execução especulativa e fora de ordem

- **E se tiver `if`/`else`?**
  - tenta adivinhar qual "lado" provavelmente vai ser executado
    - » *se der certo, ótimo; se não, joga o resultado fora*
- **A instrução n é uma soma**
- **A instrução $n + 3$ é uma multiplicação**
- **O circuito que processa cada uma dessas instruções é diferente**
- **Processa os dois ao mesmo tempo e guarda o segundo resultado para a hora certa**
  - Novamente, se tiver `if`/`else` o resultado pode ser inútil
  - Não funciona quando a maioria das instruções depende do resultado de uma instrução anterior

Mas não é tão simples...🙄

Como fazer computadores mais rápidos?

- Circuitos mais complexos são maiores e gastam mais energia

Como fazer computadores mais rápidos?

- Circuitos mais complexos são maiores e gastam mais energia
- *Clock* mais rápido também gasta mais energia

Como fazer computadores mais rápidos?

- Circuitos mais complexos são maiores e gastam mais energia
- *Clock* mais rápido também gasta mais energia
- Mais energia $\longrightarrow$ mais calor

Como fazer computadores mais rápidos?

- **Circuitos mais complexos são maiores e gastam mais energia**
- ***Clock* mais rápido também gasta mais energia**
- **Mais energia $\longrightarrow$ mais calor**
  - existe um limite de temperatura viável!

Como fazer computadores mais rápidos?

- **Circuitos mais complexos são maiores e gastam mais energia**
- ***Clock* mais rápido também gasta mais energia**
- **Mais energia → mais calor**
  - existe um limite de temperatura viável!
- **A miniaturização permite reduzir a voltagem do circuito → circuitos mais complexos e com *clock* maior sem aumentar o tamanho e o gasto energético**

Como fazer computadores mais rápidos?

- **Circuitos mais complexos são maiores e gastam mais energia**
- ***Clock* mais rápido também gasta mais energia**
- **Mais energia → mais calor**
  - existe um limite de temperatura viável!
- **A miniaturização permite reduzir a voltagem do circuito → circuitos mais complexos e com *clock* maior sem aumentar o tamanho e o gasto energético**
  - Mas já chegamos ao limite mínimo da voltagem por volta de 2005 😭

Como fazer computadores mais rápidos?

- **Circuitos mais complexos são maiores e gastam mais energia**
- ***Clock* mais rápido também gasta mais energia**
- **Mais energia → mais calor**
  - existe um limite de temperatura viável!
- **A miniaturização permite reduzir a voltagem do circuito → circuitos mais complexos e com *clock* maior sem aumentar o tamanho e o gasto energético**
  - Mas já chegamos ao limite mínimo da voltagem por volta de 2005 😭
  - Na fotolitografia, é preciso usar luz ultravioleta porque o tamanho dos elementos no circuito é menor que o comprimento de onda da luz visível

Como fazer computadores mais rápidos?

- **Pipelining, execução especulativa e fora de ordem já são amplamente usados**

Como fazer computadores mais rápidos?

- **Pipelining, execução especulativa e fora de ordem já são amplamente usados**
- **Processamento (explicitamente) paralelo**

Como fazer computadores mais rápidos?

- **Pipelining, execução especulativa e fora de ordem já são amplamente usados**
- **Processamento (explicitamente) paralelo**
  - Nem tudo pode ser paralelizado

Como fazer computadores mais rápidos?

- **Pipelining, execução especulativa e fora de ordem já são amplamente usados**
- **Processamento (explicitamente) paralelo**
  - Nem tudo pode ser paralelizado
  - Quanto mais aumentamos o paralelismo, menor o ganho

Como fazer computadores mais rápidos?

- **Pipelining, execução especulativa e fora de ordem já são amplamente usados**
- **Processamento (explicitamente) paralelo**
  - Nem tudo pode ser paralelizado
  - Quanto mais aumentamos o paralelismo, menor o ganho
  - Quase sempre, o programa precisa ser criado com paralelismo em mente

Como fazer computadores mais rápidos?

- **Pipelining, execução especulativa e fora de ordem já são amplamente usados**
- **Processamento (explicitamente) paralelo**
  - Nem tudo pode ser paralelizado
  - Quanto mais aumentamos o paralelismo, menor o ganho
  - Quase sempre, o programa precisa ser criado com paralelismo em mente
    - » *E programar aplicações paralelas é significativamente mais complexo*

E o software?

Software — Linguagens de Programação

- **Linguagens de programação**

Software — Linguagens de Programação

- **Linguagens de programação**
  - FORTRAN (FORmula TRANslator, para processamento científico, de 1957)

Software — Linguagens de Programação

- **Linguagens de programação**
  - FORTRAN (FORmula TRANslator, para processamento científico, de 1957)
  - LISP (primeira linguagem funcional, 1958)

Software — Linguagens de Programação

- **Linguagens de programação**
  - FORTRAN (FORmula TRANslator, para processamento científico, de 1957)
  - LISP (primeira linguagem funcional, 1958)
  - ALGOL (primeira linguagem estruturada, com escopo de funções etc., 1960)

Software — Linguagens de Programação

- **Linguagens de programação**
  - FORTRAN (FORmula TRANslator, para processamento científico, de 1957)
  - LISP (primeira linguagem funcional, 1958)
  - ALGOL (primeira linguagem estruturada, com escopo de funções etc., 1960)
  - Simula 67 (primeira linguagem orientada a objetos)

Software — Linguagens de Programação

- **Linguagens de programação**
  - FORTRAN (FORmula TRANslator, para processamento científico, de 1957)
  - LISP (primeira linguagem funcional, 1958)
  - ALGOL (primeira linguagem estruturada, com escopo de funções etc., 1960)
  - Simula 67 (primeira linguagem orientada a objetos)
  - C (até hoje uma das mais importantes linguagens, 1973)

Software — Linguagens de Programação

- **Linguagens de programação**
  - FORTRAN (FORmula TRANslator, para processamento científico, de 1957)
  - LISP (primeira linguagem funcional, 1958)
  - ALGOL (primeira linguagem estruturada, com escopo de funções etc., 1960)
  - Simula 67 (primeira linguagem orientada a objetos)
  - C (até hoje uma das mais importantes linguagens, 1973)
- **Atualmente, existem centenas de linguagens de programação em uso**

Software — Linguagens de Programação

- **Linguagens de programação**
  - FORTRAN (FORmula TRANslator, para processamento científico, de 1957)
  - LISP (primeira linguagem funcional, 1958)
  - ALGOL (primeira linguagem estruturada, com escopo de funções etc., 1960)
  - Simula 67 (primeira linguagem orientada a objetos)
  - C (até hoje uma das mais importantes linguagens, 1973)
- **Atualmente, existem centenas de linguagens de programação em uso**
  - Linguagens com tipagem estática e dinâmica

Software — Linguagens de Programação

- **Linguagens de programação**
  - FORTRAN (FORmula TRANslator, para processamento científico, de 1957)
  - LISP (primeira linguagem funcional, 1958)
  - ALGOL (primeira linguagem estruturada, com escopo de funções etc., 1960)
  - Simula 67 (primeira linguagem orientada a objetos)
  - C (até hoje uma das mais importantes linguagens, 1973)
- **Atualmente, existem centenas de linguagens de programação em uso**
  - Linguagens com tipagem estática e dinâmica
  - Linguagens imperativas/orientadas a objetos, funcionais e lógicas

Software — Linguagens de Programação

- **Linguagens de programação**
  - FORTRAN (FORmula TRANslator, para processamento científico, de 1957)
  - LISP (primeira linguagem funcional, 1958)
  - ALGOL (primeira linguagem estruturada, com escopo de funções etc., 1960)
  - Simula 67 (primeira linguagem orientada a objetos)
  - C (até hoje uma das mais importantes linguagens, 1973)
- **Atualmente, existem centenas de linguagens de programação em uso**
  - Linguagens com tipagem estática e dinâmica
  - Linguagens imperativas/orientadas a objetos, funcionais e lógicas
  - Linguagens interpretadas e compiladas (além de técnicas "híbridas")

Software — Linguagens de Programação

- **Linguagens de programação**
  - FORTRAN (FORmula TRANslator, para processamento científico, de 1957)
  - LISP (primeira linguagem funcional, 1958)
  - ALGOL (primeira linguagem estruturada, com escopo de funções etc., 1960)
  - Simula 67 (primeira linguagem orientada a objetos)
  - C (até hoje uma das mais importantes linguagens, 1973)
- **Atualmente, existem centenas de linguagens de programação em uso**
  - Linguagens com tipagem estática e dinâmica
  - Linguagens imperativas/orientadas a objetos, funcionais e lógicas
  - Linguagens interpretadas e compiladas (além de técnicas "híbridas")
  - ...

Software — Sistemas Operacionais

- **Sistemas Operacionais**

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia parar o computador e carregar o novo programa

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia parar o computador e carregar o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia parar o computador e carregar o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia parar o computador e carregar o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Sistemas de arquivos (mecanismo de acesso padronizado aos dados)

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia parar o computador e carregar o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Sistemas de arquivos (mecanismo de acesso padronizado aos dados)
  - Sistema para executar vários programas ao mesmo tempo (*time sharing*)

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia parar o computador e carregar o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Sistemas de arquivos (mecanismo de acesso padronizado aos dados)
  - Sistema para executar vários programas ao mesmo tempo (*time sharing*)
    - » *O sistema vai alternando entre os programas*

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia parar o computador e carregar o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Sistemas de arquivos (mecanismo de acesso padronizado aos dados)
  - Sistema para executar vários programas ao mesmo tempo (*time sharing*)
    - » *O sistema vai alternando entre os programas*
    - » *Enquanto um programa espera os dados que estão sendo lidos da fita/disco ou do usuário, outro programa pode ser executado* ⟶ *sistemas multiusuário*

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia parar o computador e carregar o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Sistemas de arquivos (mecanismo de acesso padronizado aos dados)
  - Sistema para executar vários programas ao mesmo tempo (*time sharing*)
    - » *O sistema vai alternando entre os programas*
    - » *Enquanto um programa espera os dados que estão sendo lidos da fita/disco ou do usuário, outro programa pode ser executado* ⟶ *sistemas multiusuário*
  - Ambiente de programação padronizado

Software — Sistemas Operacionais

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia parar o computador e carregar o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Sistemas de arquivos (mecanismo de acesso padronizado aos dados)
  - Sistema para executar vários programas ao mesmo tempo (*time sharing*)
    - » *O sistema vai alternando entre os programas*
    - » *Enquanto um programa espera os dados que estão sendo lidos da fita/disco ou do usuário, outro programa pode ser executado* ⟶ *sistemas multiusuário*
  - Ambiente de programação padronizado
  - Interface com o usuário padronizada

- **Como produzir programas complexos com qualidade e em tempo razoável?**

Software — Técnicas e Gestão

- **Como produzir programas complexos com qualidade e em tempo razoável?**
- **Engenharia de Software**

- **Como produzir programas complexos com qualidade e em tempo razoável?**
- **Engenharia de Software**
  - Ferramentas de apoio

- **Como produzir programas complexos com qualidade e em tempo razoável?**
- **Engenharia de Software**
  - Ferramentas de apoio
  - Técnicas de programação

Software — Técnicas e Gestão

- **Como produzir programas complexos com qualidade e em tempo razoável?**
- **Engenharia de Software**
  - Ferramentas de apoio
  - Técnicas de programação
    - » *Algoritmos, padrões de projeto...*

Software — Técnicas e Gestão

- **Como produzir programas complexos com qualidade e em tempo razoável?**
- **Engenharia de Software**
  - Ferramentas de apoio
  - Técnicas de programação
    - » *Algoritmos, padrões de projeto...*
  - Metodologias de gestão

Software — Técnicas e Gestão

- **Como produzir programas complexos com qualidade e em tempo razoável?**
- **Engenharia de Software**
  - Ferramentas de apoio
  - Técnicas de programação
    - » *Algoritmos, padrões de projeto...*
  - Metodologias de gestão
    - » *Hierarquia*

Software — Técnicas e Gestão

- **Como produzir programas complexos com qualidade e em tempo razoável?**
- **Engenharia de Software**
  - Ferramentas de apoio
  - Técnicas de programação
    - » *Algoritmos, padrões de projeto...*
  - Metodologias de gestão
    - » *Hierarquia*
    - » *Papéis (arquiteto, gerente de produto, programador...)*

Popularização e o final da pré-história

Software como produto

- **No início, o desafio era desenvolver o hardware**

Software como produto

- **No início, o desafio era desenvolver o hardware**
  - A programação era um “mero detalhe”

Software como produto

- **No início, o desafio era desenvolver o hardware**
  - A programação era um “mero detalhe”
    - » *Geralmente, mulheres*

Software como produto

- **No início, o desafio era desenvolver o hardware**
  - A programação era um “mero detalhe”
    - » *Geralmente, mulheres*
    - » *Não era vendido como um produto, mas sim consultoria*

Software como produto

- **No início, o desafio era desenvolver o hardware**
  - A programação era um “mero detalhe”
    - » *Geralmente, mulheres*
    - » *Não era vendido como um produto, mas sim consultoria*
- **Com o tempo acabou se tornando um produto**

Software como produto

- **No início, o desafio era desenvolver o hardware**
  - A programação era um “mero detalhe”
    - » *Geralmente, mulheres*
    - » *Não era vendido como um produto, mas sim consultoria*
- **Com o tempo acabou se tornando um produto**
  - Discussões sobre proteção por patentes ou *copyright* nos anos 1960

Software como produto

- **No início, o desafio era desenvolver o hardware**
  - A programação era um “mero detalhe”
    - » *Geralmente, mulheres*
    - » *Não era vendido como um produto, mas sim consultoria*
- **Com o tempo acabou se tornando um produto**
  - Discussões sobre proteção por patentes ou *copyright* nos anos 1960
    - » *Optou-se pelo copyright*

Software como produto

- **No início, o desafio era desenvolver o hardware**
  - A programação era um “mero detalhe”
    - » *Geralmente, mulheres*
    - » *Não era vendido como um produto, mas sim consultoria*
- **Com o tempo acabou se tornando um produto**
  - Discussões sobre proteção por patentes ou *copyright* nos anos 1960
    - » *Optou-se pelo copyright*
- **Crescimento com o avanço dos PCs no final dos anos 1970**

Software como produto

- **No início, o desafio era desenvolver o hardware**
  - A programação era um “mero detalhe”
    - » *Geralmente, mulheres*
    - » *Não era vendido como um produto, mas sim consultoria*
- **Com o tempo acabou se tornando um produto**
  - Discussões sobre proteção por patentes ou *copyright* nos anos 1960
    - » *Optou-se pelo copyright*
- **Crescimento com o avanço dos PCs no final dos anos 1970**
  - “Carta aos hobbyistas” de Bill Gates (1976)

Software como produto

- **No início, o desafio era desenvolver o hardware**
  - A programação era um "mero detalhe"
    - » *Geralmente, mulheres*
    - » *Não era vendido como um produto, mas sim consultoria*
- **Com o tempo acabou se tornando um produto**
  - Discussões sobre proteção por patentes ou *copyright* nos anos 1960
    - » *Optou-se pelo copyright*
- **Crescimento com o avanço dos PCs no final dos anos 1970**
  - "Carta aos hobbyistas" de Bill Gates (1976)
  - *Free Software Foundation* e o projeto GNU de Richard Stallman (1983)

O BASIC da Microsoft

- **Em 1975, foi lançado o Altair 8080 — um kit de componentes para hobbyistas**

O BASIC da Microsoft

- **Em 1975, foi lançado o Altair 8080 — um kit de componentes para hobbyistas**
  - Não tinha teclado ou tela, apenas alguns botões e leds

O BASIC da Microsoft

- **Em 1975, foi lançado o Altair 8080 — um kit de componentes para hobbyistas**
  - Não tinha teclado ou tela, apenas alguns botões e leds
  - Era possível ler e gravar dados gravados em fitas cassete

O BASIC da Microsoft

- **Em 1975, foi lançado o Altair 8080 — um kit de componentes para hobbyistas**
  - Não tinha teclado ou tela, apenas alguns botões e leds
  - Era possível ler e gravar dados gravados em fitas cassete
- **Paul Allen e Bill Gates criaram um interpretador BASIC para o Altair 8080 e fundaram a Microsoft**

O BASIC da Microsoft

- **Em 1977, três computadores muito influentes foram lançados:**

O BASIC da Microsoft

- **Em 1977, três computadores muito influentes foram lançados:**
  - Tandy TRS-80 — processador Z80, 4–16kB de memória e grande rede de varejo (Radio Shack) oferecendo upgrades, suporte e treinamento

O BASIC da Microsoft

- **Em 1977, três computadores muito influentes foram lançados:**
  - Tandy TRS-80 — processador Z80, 4–16kB de memória e grande rede de varejo (Radio Shack) oferecendo upgrades, suporte e treinamento
  - Commodore PET — processador 6502, 4–8kB de memória

O BASIC da Microsoft

- **Em 1977, três computadores muito influentes foram lançados:**
  - Tandy TRS-80 — processador Z80, 4–16kB de memória e grande rede de varejo (Radio Shack) oferecendo upgrades, suporte e treinamento
  - Commodore PET — processador 6502, 4–8kB de memória
  - Apple][— processador 6502, 4–48kB de memória, cores e saída para TV

O BASIC da Microsoft

- **Em 1977, três computadores muito influentes foram lançados:**
  - Tandy TRS-80 — processador Z80, 4–16kB de memória e grande rede de varejo (Radio Shack) oferecendo upgrades, suporte e treinamento
  - Commodore PET — processador 6502, 4–8kB de memória
  - Apple][— processador 6502, 4–48kB de memória, cores e saída para TV
- **Os três incluíam o BASIC da Microsoft, que acabou se tornando "padrão"**

O BASIC da Microsoft

- **Em 1977, três computadores muito influentes foram lançados:**
  - Tandy TRS-80 — processador Z80, 4–16kB de memória e grande rede de varejo (Radio Shack) oferecendo upgrades, suporte e treinamento
  - Commodore PET — processador 6502, 4–8kB de memória
  - Apple][— processador 6502, 4–48kB de memória, cores e saída para TV
- **Os três incluíam o BASIC da Microsoft, que acabou se tornando "padrão"**
- **Vários computadores posteriores incluíam também o sistema operacional CP/M que, de maneira similar, acabou se tornando "padrão"**

Visicalc

- **Computadores pessoais eram “divertidos”, mas pouco úteis, especialmente para empresas**

Visicalc

- Computadores pessoais eram “divertidos”, mas pouco úteis, especialmente para empresas
- Em 1979, foi lançado o VisiCalc, primeira planilha eletrônica, para o Apple][

Visicalc

- Computadores pessoais eram “divertidos”, mas pouco úteis, especialmente para empresas
- Em 1979, foi lançado o VisiCalc, primeira planilha eletrônica, para o Apple][
- “Killer App”

Visicalc

- **Computadores pessoais eram “divertidos”, mas pouco úteis, especialmente para empresas**
- **Em 1979, foi lançado o VisiCalc, primeira planilha eletrônica, para o Apple][**
- **“Killer App”**
  - Com seu enorme sucesso, alavancou as vendas do Apple][

Visicalc

- **Computadores pessoais eram “divertidos”, mas pouco úteis, especialmente para empresas**
- **Em 1979, foi lançado o VisiCalc, primeira planilha eletrônica, para o Apple][**
- **“Killer App”**
  - Com seu enorme sucesso, alavancou as vendas do Apple][
- **Foi disponibilizado para outras plataformas cerca de um ano depois**

Visicalc

- **Computadores pessoais eram “divertidos”, mas pouco úteis, especialmente para empresas**
- **Em 1979, foi lançado o VisiCalc, primeira planilha eletrônica, para o Apple][**
- **“Killer App”**
  - Com seu enorme sucesso, alavancou as vendas do Apple][
- **Foi disponibilizado para outras plataformas cerca de um ano depois**
  - Mas a posição do Apple][já estava consolidada

O PC da IBM, a Intel e a Microsoft

- No final dos anos 1970, o computador pessoal começou a se tornar “popular”

O PC da IBM, a Intel e a Microsoft

- No final dos anos 1970, o computador pessoal começou a se tornar "popular"
- Para a IBM, esse era um mercado irrelevante

O PC da IBM, a Intel e a Microsoft

- **No final dos anos 1970, o computador pessoal começou a se tornar "popular"**
- **Para a IBM, esse era um mercado irrelevante**
  - No entanto, sendo a maior empresa de computação da época, era importante oferecer um produto nessa categoria

O PC da IBM, a Intel e a Microsoft

- **No final dos anos 1970, o computador pessoal começou a se tornar "popular"**
- **Para a IBM, esse era um mercado irrelevante**
  - No entanto, sendo a maior empresa de computação da época, era importante oferecer um produto nessa categoria
- **A IBM normalmente desenvolvia o projeto de seus computadores "do zero"**

O PC da IBM, a Intel e a Microsoft

- **No final dos anos 1970, o computador pessoal começou a se tornar "popular"**
- **Para a IBM, esse era um mercado irrelevante**
  - No entanto, sendo a maior empresa de computação da época, era importante oferecer um produto nessa categoria
- **A IBM normalmente desenvolvia o projeto de seus computadores "do zero"**
  - Mas, como o PC era um projeto de menor relevância, ele foi projetado usando componentes prontos disponíveis no mercado (como o processador Intel 8088)

O PC da IBM, a Intel e a Microsoft

- Por ameaças de processos antitruste, a IBM não desenvolvia mais o software para seus sistemas

O PC da IBM, a Intel e a Microsoft

- Por ameaças de processos antitruste, a IBM não desenvolvia mais o software para seus sistemas
- Ela procurou os fornecedores “tradicionais” de computadores pessoais

O PC da IBM, a Intel e a Microsoft

- **Por ameaças de processos antitruste, a IBM não desenvolvia mais o software para seus sistemas**
- **Ela procurou os fornecedores "tradicionais" de computadores pessoais**
  - BASIC $\longrightarrow$ Microsoft

O PC da IBM, a Intel e a Microsoft

- **Por ameaças de processos antitruste, a IBM não desenvolvia mais o software para seus sistemas**
- **Ela procurou os fornecedores "tradicionais" de computadores pessoais**
  - BASIC $\longrightarrow$ Microsoft
  - CP/M $\longrightarrow$ Digital Research

O PC da IBM, a Intel e a Microsoft

- **Por ameaças de processos antitruste, a IBM não desenvolvia mais o software para seus sistemas**
- **Ela procurou os fornecedores "tradicionais" de computadores pessoais**
  - BASIC ⟶ Microsoft
  - CP/M ⟶ Digital Research
    - » *A negociação com a Digital Research falhou, então a Microsoft foi contratada para fazer também o sistema operacional (DOS)*

O PC da IBM, a Intel e a Microsoft

- **Por ameaças de processos antitruste, a IBM não desenvolvia mais o software para seus sistemas**
- **Ela procurou os fornecedores "tradicionais" de computadores pessoais**
  - BASIC ⟶ Microsoft
  - CP/M ⟶ Digital Research
    - » *A negociação com a Digital Research falhou, então a Microsoft foi contratada para fazer também o sistema operacional (DOS)*
    - » *A Microsoft comprou (por US$ 50.000) o QDOS (Quick and Dirty Operating System) e o adaptou*

O PC da IBM, a Intel e a Microsoft

- **Por ameaças de processos antitruste, a IBM não desenvolvia mais o software para seus sistemas**
- **Ela procurou os fornecedores "tradicionais" de computadores pessoais**
  - BASIC ⟶ Microsoft
  - CP/M ⟶ Digital Research
    - » *A negociação com a Digital Research falhou, então a Microsoft foi contratada para fazer também o sistema operacional (DOS)*
    - » *A Microsoft comprou (por US$ 50.000) o QDOS (Quick and Dirty Operating System) e o adaptou*
- **Ao invés de vender o DOS para a IBM, a Microsoft fez um acordo de licenciamento**

O PC da IBM, a Intel e a Microsoft

- **Por ameaças de processos antitruste, a IBM não desenvolvia mais o software para seus sistemas**
- **Ela procurou os fornecedores "tradicionais" de computadores pessoais**
  - BASIC ⟶ Microsoft
  - CP/M ⟶ Digital Research
    - » *A negociação com a Digital Research falhou, então a Microsoft foi contratada para fazer também o sistema operacional (DOS)*
    - » *A Microsoft comprou (por US$ 50.000) o QDOS (Quick and Dirty Operating System) e o adaptou*
- **Ao invés de vender o DOS para a IBM, a Microsoft fez um acordo de licenciamento**
  - Para a IBM, o software tinha pouco valor

O PC da IBM, a Intel e a Microsoft

- **Por ameaças de processos antitruste, a IBM não desenvolvia mais o software para seus sistemas**
- **Ela procurou os fornecedores "tradicionais" de computadores pessoais**
  - BASIC ⟶ Microsoft
  - CP/M ⟶ Digital Research
    - » *A negociação com a Digital Research falhou, então a Microsoft foi contratada para fazer também o sistema operacional (DOS)*
    - » *A Microsoft comprou (por US$ 50.000) o QDOS (Quick and Dirty Operating System) e o adaptou*
- **Ao invés de vender o DOS para a IBM, a Microsoft fez um acordo de licenciamento**
  - Para a IBM, o software tinha pouco valor
  - Esse foi o "momento da virada" para a Microsoft

Como todo o hardware e software era produzido por terceiros, rapidamente surgiram PCs compatíveis de outros fabricantes

Como todo o hardware e software era produzido por terceiros, rapidamente surgiram PCs compatíveis de outros fabricantes

Microsoft e Intel eram essenciais nesse novo mercado

GUIs, Xerox e o Macintosh

- Xerox em 1969: “O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos”

GUIs, Xerox e o Macintosh

- Xerox em 1969: “O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos” 🤣

GUIs, Xerox e o Macintosh

- **Xerox em 1969: "O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos"** 🤣
  - Começamos a reduzir o uso de papel há poucos anos

GUIs, Xerox e o Macintosh

- **Xerox em 1969: "O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos"** 🤣
  - Começamos a reduzir o uso de papel há poucos anos
- **Xerox Palo Alto Research Center (PARC) — 1970**

GUIs, Xerox e o Macintosh

- **Xerox em 1969: "O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos"** 🤣
  - Começamos a reduzir o uso de papel há poucos anos
- **Xerox Palo Alto Research Center (PARC) — 1970**
  - Ethernet

GUIs, Xerox e o Macintosh

- **Xerox em 1969: “O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos”** 🤣
  - Começamos a reduzir o uso de papel há poucos anos
- **Xerox Palo Alto Research Center (PARC) — 1970**
  - Ethernet
  - Smalltalk (“a” linguagem orientada a objetos)

GUIs, Xerox e o Macintosh

- **Xerox em 1969: "O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos"** 🤣
  - Começamos a reduzir o uso de papel há poucos anos
- **Xerox Palo Alto Research Center (PARC) — 1970**
  - Ethernet
  - Smalltalk ("a" linguagem orientada a objetos)
  - Impressora a laser

GUIs, Xerox e o Macintosh

- **Xerox em 1969: "O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos"** 🤣
  - Começamos a reduzir o uso de papel há poucos anos
- **Xerox Palo Alto Research Center (PARC) — 1970**
  - Ethernet
  - Smalltalk ("a" linguagem orientada a objetos)
  - Impressora a laser
  - Interpress (precursor do PostScript e PDF)

GUIs, Xerox e o Macintosh

- **Xerox em 1969: "O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos"** 🤣
  - Começamos a reduzir o uso de papel há poucos anos
- **Xerox Palo Alto Research Center (PARC) — 1970**
  - Ethernet
  - Smalltalk ("a" linguagem orientada a objetos)
  - Impressora a laser
  - Interpress (precursor do PostScript e PDF)
- **Xerox Alto (1972?), primeiro computador com GUI e mouse**

GUIs, Xerox e o Macintosh

- **Xerox em 1969: "O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos"** 🤣
  - Começamos a reduzir o uso de papel há poucos anos
- **Xerox Palo Alto Research Center (PARC) — 1970**
  - Ethernet
  - Smalltalk ("a" linguagem orientada a objetos)
  - Impressora a laser
  - Interpress (precursor do PostScript e PDF)
- **Xerox Alto (1972?), primeiro computador com GUI e mouse**
  - Baseado nas ideias de Douglas Engelbart (The Mother of All Demos) — 1968

GUIs, Xerox e o Macintosh

- **Xerox em 1969: "O computador vai acabar com o papel nas empresas e nós vamos nos tornar obsoletos"** 🤣
  - Começamos a reduzir o uso de papel há poucos anos
- **Xerox Palo Alto Research Center (PARC) — 1970**
  - Ethernet
  - Smalltalk ("a" linguagem orientada a objetos)
  - Impressora a laser
  - Interpress (precursor do PostScript e PDF)
- **Xerox Alto (1972?), primeiro computador com GUI e mouse**
  - Baseado nas ideias de Douglas Engelbart (The Mother of All Demos) — 1968
- **Gestores não deram valor**

GUIs, Xerox e o Macintosh

- Em 1979, Steve Jobs visitou o PARC, o que acabou dando origem ao Macintosh (e ao Windows), lançado em 1984

GUIs, Xerox e o Macintosh

- Em 1979, Steve Jobs visitou o PARC, o que acabou dando origem ao Macintosh (e ao Windows), lançado em 1984
- Em 1982, dois desenvolvedores do Interpress criaram a Adobe e a linguagem PostScript

GUIs, Xerox e o Macintosh

- Em 1979, Steve Jobs visitou o PARC, o que acabou dando origem ao Macintosh (e ao Windows), lançado em 1984
- Em 1982, dois desenvolvedores do Interpress criaram a Adobe e a linguagem PostScript
- Em 1985, Adobe e Apple fizeram um acordo para a produção de impressoras laser com suporte à linguagem PostScript (a linha *Apple LaserWriter*)

GUIs, Xerox e o Macintosh

- **Em 1979, Steve Jobs visitou o PARC, o que acabou dando origem ao Macintosh (e ao Windows), lançado em 1984**
- **Em 1982, dois desenvolvedores do Interpress criaram a Adobe e a linguagem PostScript**
- **Em 1985, Adobe e Apple fizeram um acordo para a produção de impressoras laser com suporte à linguagem PostScript (a linha *Apple LaserWriter*)**
  - Gráficos sofisticados, vários tipos de letra e boa qualidade de impressão

GUIs, Xerox e o Macintosh

- **Em 1979, Steve Jobs visitou o PARC, o que acabou dando origem ao Macintosh (e ao Windows), lançado em 1984**
- **Em 1982, dois desenvolvedores do Interpress criaram a Adobe e a linguagem PostScript**
- **Em 1985, Adobe e Apple fizeram um acordo para a produção de impressoras laser com suporte à linguagem PostScript (a linha *Apple LaserWriter*)**
  - Gráficos sofisticados, vários tipos de letra e boa qualidade de impressão
- **O Macintosh se tornou a plataforma "padrão" para produção gráfica**

GUIs, Xerox e o Macintosh

- **Em 1979, Steve Jobs visitou o PARC, o que acabou dando origem ao Macintosh (e ao Windows), lançado em 1984**
- **Em 1982, dois desenvolvedores do Interpress criaram a Adobe e a linguagem PostScript**
- **Em 1985, Adobe e Apple fizeram um acordo para a produção de impressoras laser com suporte à linguagem PostScript (a linha *Apple LaserWriter*)**
  - Gráficos sofisticados, vários tipos de letra e boa qualidade de impressão
- **O Macintosh se tornou a plataforma "padrão" para produção gráfica**
- **A linguagem PostScript se tornou um "padrão" adotado por muitas outras impressoras (e deu origem ao PDF)**

A Internet

- A Internet surgiu como um sistema experimental em 1971

A Internet

- A Internet surgiu como um sistema experimental em 1971
- A partir de 1994, o acesso começou a se popularizar

A Internet

- **A Internet surgiu como um sistema experimental em 1971**
- **A partir de 1994, o acesso começou a se popularizar**
  - Mesma época em que a WWW se popularizou

A Internet

- **A Internet surgiu como um sistema experimental em 1971**
- **A partir de 1994, o acesso começou a se popularizar**
  - Mesma época em que a WWW se popularizou
- **Crescimento trouxe grandes investimentos de risco**

A Internet

- **A Internet surgiu como um sistema experimental em 1971**
- **A partir de 1994, o acesso começou a se popularizar**
  - Mesma época em que a WWW se popularizou
- **Crescimento trouxe grandes investimentos de risco**
  - "Estouro da bolha" em 2000

Como “monetizar” a Internet?

Como “monetizar” a Internet?

- **Na imensa maioria dos casos, a resposta foi “propaganda segmentada”**

Como “monetizar” a Internet?

- **Na imensa maioria dos casos, a resposta foi “propaganda segmentada”**
  - Anúncios relevantes para o conteúdo sendo acessado pelo usuário

Como “monetizar” a Internet?

- **Na imensa maioria dos casos, a resposta foi “propaganda segmentada”**
  - Anúncios relevantes para o conteúdo sendo acessado pelo usuário
    - » *Google adsense, anúncios junto com os resultados das buscas*

Como “monetizar” a Internet?

- **Na imensa maioria dos casos, a resposta foi “propaganda segmentada”**
  - Anúncios relevantes para o conteúdo sendo acessado pelo usuário
    - » *Google adsense, anúncios junto com os resultados das buscas*
  - Anúncios relevantes para o perfil do usuário

Como “monetizar” a Internet?

- **Na imensa maioria dos casos, a resposta foi “propaganda segmentada”**
  - Anúncios relevantes para o conteúdo sendo acessado pelo usuário
    - » *Google adsense, anúncios junto com os resultados das buscas*
  - Anúncios relevantes para o perfil do usuário
    - » *gmail, redes sociais*

Como “monetizar” a Internet?

- **Na imensa maioria dos casos, a resposta foi “propaganda segmentada”**
  - Anúncios relevantes para o conteúdo sendo acessado pelo usuário
    - » *Google adsense, anúncios junto com os resultados das buscas*
  - Anúncios relevantes para o perfil do usuário
    - » *gmail, redes sociais*
- **Diversos problemas**

Como “monetizar” a Internet?

- **Na imensa maioria dos casos, a resposta foi “propaganda segmentada”**
  - Anúncios relevantes para o conteúdo sendo acessado pelo usuário
    - » *Google adsense, anúncios junto com os resultados das buscas*
  - Anúncios relevantes para o perfil do usuário
    - » *gmail, redes sociais*
- **Diversos problemas**
  - Privacidade, conteúdo enviesado, discurso de ódio, polarização política...

Como “monetizar” a Internet?

- **Na imensa maioria dos casos, a resposta foi “propaganda segmentada”**
  - Anúncios relevantes para o conteúdo sendo acessado pelo usuário
    - » *Google adsense, anúncios junto com os resultados das buscas*
  - Anúncios relevantes para o perfil do usuário
    - » *gmail, redes sociais*
- **Diversos problemas**
  - Privacidade, conteúdo enviesado, discurso de ódio, polarização política...
- ***Streaming* e outros serviços de assinatura**

Como “monetizar” a Internet?

- **Na imensa maioria dos casos, a resposta foi “propaganda segmentada”**
  - Anúncios relevantes para o conteúdo sendo acessado pelo usuário
    - » *Google adsense, anúncios junto com os resultados das buscas*
  - Anúncios relevantes para o perfil do usuário
    - » *gmail, redes sociais*
- **Diversos problemas**
  - Privacidade, conteúdo enviesado, discurso de ódio, polarização política...
- ***Streaming* e outros serviços de assinatura**
  - Também fazem coleta de dados

Como “monetizar” a Internet?

- **Na imensa maioria dos casos, a resposta foi “propaganda segmentada”**
  - Anúncios relevantes para o conteúdo sendo acessado pelo usuário
    - » *Google adsense, anúncios junto com os resultados das buscas*
  - Anúncios relevantes para o perfil do usuário
    - » *gmail, redes sociais*
- **Diversos problemas**
  - Privacidade, conteúdo enviesado, discurso de ódio, polarização política...
- ***Streaming* e outros serviços de assinatura**
  - Também fazem coleta de dados
  - Usam DRM

A corrida pelo smartphone

- Em 1996, a Palm lançou seu PDA, o PalmPilot

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador “auxiliar” portátil, usado de maneira integrada com computadores desktop

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
- **Em 2002/2003, primeiros smartphones e telefones com câmera**

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
- **Em 2002/2003, primeiros smartphones e telefones com câmera**
  - Teclado físico + tela sensível (mão ou caneta)

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
- **Em 2002/2003, primeiros smartphones e telefones com câmera**
  - Teclado físico + tela sensível (mão ou caneta)
- **Em 2005–2007, a Nokia lançou diversos *tablets* com o sistema Maemo, com interface usando tela sensível ao toque**

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
- **Em 2002/2003, primeiros smartphones e telefones com câmera**
  - Teclado físico + tela sensível (mão ou caneta)
- **Em 2005–2007, a Nokia lançou diversos *tablets* com o sistema Maemo, com interface usando tela sensível ao toque**
- **Em 2007, a Apple lançou o iPhone**

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
- **Em 2002/2003, primeiros smartphones e telefones com câmera**
  - Teclado físico + tela sensível (mão ou caneta)
- **Em 2005–2007, a Nokia lançou diversos *tablets* com o sistema Maemo, com interface usando tela sensível ao toque**
- **Em 2007, a Apple lançou o iPhone**
  - Excelente combinação de recursos que já existiam, mas não em um único dispositivo

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
- **Em 2002/2003, primeiros smartphones e telefones com câmera**
  - Teclado físico + tela sensível (mão ou caneta)
- **Em 2005–2007, a Nokia lançou diversos *tablets* com o sistema Maemo, com interface usando tela sensível ao toque**
- **Em 2007, a Apple lançou o iPhone**
  - Excelente combinação de recursos que já existiam, mas não em um único dispositivo
  - Comercialização em parceria com pacotes de dados das operadoras de telefonia, aumentando muito a utilidade do dispositivo

Fim da pré-história!

Tendências

- **Software livre**

Tendências

- **Software livre**
- **Computação paralela**

Tendências

- **Software livre**
- **Computação paralela**
- **Computadores portáteis e vestíveis**

Tendências

- **Software livre**
- **Computação paralela**
- **Computadores portáteis e vestíveis**
- **Computação ubíqua**

Tendências

- Software livre
- Computação paralela
- Computadores portáteis e vestíveis
- Computação ubíqua
- Computação verde

Tendências

- Software livre
- Computação paralela
- Computadores portáteis e vestíveis
- Computação ubíqua
- Computação verde
- Inteligência Artificial

Tendências

- Software livre
- Computação paralela
- Computadores portáteis e vestíveis
- Computação ubíqua
- Computação verde
- Inteligência Artificial
- Criptomoedas

Software livre e patentes

- Atualmente, o software livre é essencial na computação

Software livre e patentes

- **Atualmente, o software livre é essencial na computação**
  - Mas por razões econômicas

Software livre e patentes

- **Atualmente, o software livre é essencial na computação**
  - Mas por razões econômicas
  - As questões éticas da FSF continuam tão prementes quanto antes

Software livre e patentes

- **Atualmente, o software livre é essencial na computação**
  - Mas por razões econômicas
  - As questões éticas da FSF continuam tão prementes quanto antes
- **O escopo das diversas formas de proteção à "propriedade intelectual" vem crescendo em todas as áreas**

Software livre e patentes

- **Atualmente, o software livre é essencial na computação**
  - Mas por razões econômicas
  - As questões éticas da FSF continuam tão prementes quanto antes
- **O escopo das diversas formas de proteção à "propriedade intelectual" vem crescendo em todas as áreas**
  - No caso do software, patentes se tornaram comuns, mesmo que legalmente "não existam"

Tecnologias “disruptivas” e energia

- **Criptomoedas surgiram para se contrapor a instituições financeiras tradicionais**

Tecnologias “disruptivas” e energia

- **Criptomoedas surgiram para se contrapor a instituições financeiras tradicionais**
  - Tornaram-se ativos negociados dentro das instituições financeiras tradicionais

Tecnologias “disruptivas” e energia

- **Criptomoedas surgiram para se contrapor a instituições financeiras tradicionais**
  - Tornaram-se ativos negociados dentro das instituições financeiras tradicionais
  - São sorvedouros de energia com benefícios públicos questionáveis

Tecnologias “disruptivas” e energia

- **Criptomoedas surgiram para se contrapor a instituições financeiras tradicionais**
  - Tornaram-se ativos negociados dentro das instituições financeiras tradicionais
  - São sorvedouros de energia com benefícios públicos questionáveis
- **Inteligência Artificial traz diversas promessas**

Tecnologias “disruptivas” e energia

- **Criptomoedas surgiram para se contrapor a instituições financeiras tradicionais**
  - Tornaram-se ativos negociados dentro das instituições financeiras tradicionais
  - São sorvedouros de energia com benefícios públicos questionáveis
- **Inteligência Artificial traz diversas promessas**
  - Mas pode vir a servir apenas para reduzir custos e empregos

Tecnologias “disruptivas” e energia

- **Criptomoedas surgiram para se contrapor a instituições financeiras tradicionais**
  - Tornaram-se ativos negociados dentro das instituições financeiras tradicionais
  - São sorvedouros de energia com benefícios públicos questionáveis
- **Inteligência Artificial traz diversas promessas**
  - Mas pode vir a servir apenas para reduzir custos e empregos
  - Também é um sorvedouro de energia

Privacidade, segurança, entretenimento e controle

- O computador hoje media grande parte de nossas interações

Privacidade, segurança, entretenimento e controle

- **O computador hoje media grande parte de nossas interações**
  - Finanças

Privacidade, segurança, entretenimento e controle

- **O computador hoje media grande parte de nossas interações**
  - Finanças
  - Informação e notícias

Privacidade, segurança, entretenimento e controle

- **O computador hoje media grande parte de nossas interações**
  - Finanças
  - Informação e notícias
  - Trabalho

Privacidade, segurança, entretenimento e controle

- **O computador hoje media grande parte de nossas interações**
  - Finanças
  - Informação e notícias
  - Trabalho
  - Escola e aprendizado

Privacidade, segurança, entretenimento e controle

- **O computador hoje media grande parte de nossas interações**
  - Finanças
  - Informação e notícias
  - Trabalho
  - Escola e aprendizado
  - Relacionamentos pessoais

Privacidade, segurança, entretenimento e controle

- **O computador hoje media grande parte de nossas interações**
  - Finanças
  - Informação e notícias
  - Trabalho
  - Escola e aprendizado
  - Relacionamentos pessoais
  - Governo e política

Privacidade, segurança, entretenimento e controle

- **O computador hoje media grande parte de nossas interações**
  - Finanças
  - Informação e notícias
  - Trabalho
  - Escola e aprendizado
  - Relacionamentos pessoais
  - Governo e política
  - Entretenimento

Privacidade, segurança, entretenimento e controle

- **O computador hoje media grande parte de nossas interações**
  - Finanças
  - Informação e notícias
  - Trabalho
  - Escola e aprendizado
  - Relacionamentos pessoais
  - Governo e política
  - Entretenimento
  - ...

Privacidade, segurança, entretenimento e controle

- **Falhas de segurança podem ter impactos significativos**

Privacidade, segurança, entretenimento e controle

- **Falhas de segurança podem ter impactos significativos**
- **O acesso indiscriminado a grandes volumes de dados (*big data*) — como localização, buscas, relacionamentos etc. — pode trazer grandes benefícios, mas também problemas**

Privacidade, segurança, entretenimento e controle

- **Falhas de segurança podem ter impactos significativos**
- **O acesso indiscriminado a grandes volumes de dados (*big data*) — como localização, buscas, relacionamentos etc. — pode trazer grandes benefícios, mas também problemas**
  - Privacidade individual

Privacidade, segurança, entretenimento e controle

- **Falhas de segurança podem ter impactos significativos**
- **O acesso indiscriminado a grandes volumes de dados (*big data*) — como localização, buscas, relacionamentos etc. — pode trazer grandes benefícios, mas também problemas**
  - Privacidade individual
  - Manipulação ideológica

Privacidade, segurança, entretenimento e controle

- **Falhas de segurança podem ter impactos significativos**
- **O acesso indiscriminado a grandes volumes de dados (*big data*) — como localização, buscas, relacionamentos etc. — pode trazer grandes benefícios, mas também problemas**
  - Privacidade individual
  - Manipulação ideológica
  - ...

Privacidade, segurança, entretenimento e controle

- **Falhas de segurança podem ter impactos significativos**
- **O acesso indiscriminado a grandes volumes de dados (*big data*) — como localização, buscas, relacionamentos etc. — pode trazer grandes benefícios, mas também problemas**
  - Privacidade individual
  - Manipulação ideológica
  - ...
  - *Weapons of Math Destruction* (Cathy O'Neil)

Privacidade, segurança, entretenimento e controle

- **Falhas de segurança podem ter impactos significativos**
- **O acesso indiscriminado a grandes volumes de dados (*big data*) — como localização, buscas, relacionamentos etc. — pode trazer grandes benefícios, mas também problemas**
  - Privacidade individual
  - Manipulação ideológica
  - ...
  - *Weapons of Math Destruction* (Cathy O'Neil)
- **A computação em nuvem promove uma grande concentração de dados e infraestrutura (com consequências econômicas) e também traz problemas com privacidade**

Privacidade, segurança, entretenimento e controle

- **Falhas de segurança podem ter impactos significativos**
- **O acesso indiscriminado a grandes volumes de dados (*big data*) — como localização, buscas, relacionamentos etc. — pode trazer grandes benefícios, mas também problemas**
  - Privacidade individual
  - Manipulação ideológica
  - ...
  - *Weapons of Math Destruction* (Cathy O'Neil)
- **A computação em nuvem promove uma grande concentração de dados e infraestrutura (com consequências econômicas) e também traz problemas com privacidade**
- **O acesso ao "conteúdo" através do computador tem sido viabilizado por tecnologias de DRM**

Computer science is no more about computers than astronomy is about telescopes

— Edsger Dijkstra