MAC 113 — Introdução à Ciência da Computação

Aula 29

Nelson Lago

1°/2025

Previously on MAC113...

Busca binária

Quando procuramos em um dicionário:

- **Abrimos o dicionário "em algum lugar perto do meio"**
  - Ih, é antes ⟶ abre "em algum lugar do meio" entre o começo e a página atual
  - Ih, é depois ⟶ abre "em algum lugar do meio" entre a página atual e o fim
- **No caso do dicionário, sabemos o mínimo ("a") e o máximo ("z") da lista, então não abrimos "no meio", mas tentamos "chutar" uma página próxima**
- **Numa lista de números genérica, não temos essa "dica", então o melhor é olhar o "meio" mesmo**

Busca binária

Vamos procurar pelo número 23

Busca binária

Vamos procurar pelo número 23

1	3	7	14	21	22	23	26	27	30	31

Busca binária

Vamos procurar pelo número 23

Busca binária

Vamos procurar pelo número 23

Busca binária

Vamos procurar pelo número 23

Busca binária

Vamos procurar pelo número 23

Busca binária

Vamos procurar pelo número 23

Busca binária

Vamos procurar pelo número 23

Busca binária

Escreva uma função que recebe um vector ordenado de números vec e um número n e informa se o número está ou não no vector

```
pertence <- function(vec, n) {
    i <- 1
    j <- length(vec)
    while (i <= j) {
        meio <- (i + j) %/% 2
        if (n == vec[meio]) { return (TRUE) }
        else if (n > vec[meio]) { i <- meio + 1 }
        else { j <- meio - 1 }
    }
    return (FALSE)
}
```

Busca binária

Escreva uma função que recebe um vector ordenado de números vec e um número n e informa se o número está ou não no vector

```
pertence <- function(vec, n) {
    while (length(vec) > 1) {
        meio <- length(vec) %/% 2
        if (n == vec[meio]) { return (TRUE) }
        if (n < vec[meio]) { vec <- vec[seq_len(meio - 1)] }
        else { vec <- vec[-seq_len(meio)] }
    }
    if (length(vec) == 1) { return (n == vec[1]) }
    return (FALSE)
}
```

Divisão e conquista

Divisão e conquista

Você sabe somar mais de dois números de uma vez?

$$1 + 2 + 3 + 4 = (1 + 2) + 3 + 4 = 3 + 3 + 4 = (3 + 3) + 4 = 6 + 4 = 10$$

Imagine que o computador só é capaz de fazer uma soma de cada vez; faça uma função que soma uma lista de números

```
soma <- function(v) {
    s <- 0
    for (n in v) { s <- s + n }
    return (s)
}
```

Pega um por um e soma

(como no exemplo $1 + 2 + 3 + 4$ acima)

Divisão e conquista

Mas podemos pensar um pouco diferente:

```
somatório <- function(l) {
    if (length(l) == 0) { return (0) }
    if (length(l) == 1) { return (l[1]) }
    if (length(l) == 2) { return (l[1] + l[2]) }
    if (length(l) >= 3) { return (l[1] + somatório(l[-1])) }
}
```

$$1 + 2 + 3 + 4 = 1 + (2 + 3 + 4) = 1 + (2 + (3 + 4)) = 1 + (2 + 7) = 1 + 9 = 10$$

A soma de n números é a soma do primeiro com a soma do restante

Divisão e conquista

```
somatório <- function(l) {
    if (length(l) == 0) { return (0) }
    return (l[1] + somatório(l[-1]))
}
```

```
somatório <- function (l) {
    return (l[1] + somatório(l[-1]))
}
```

- **Não dá para fazer chamadas recursivas para sempre!**
  - É preciso haver ao menos um caso em que **não** fazemos a chamada recursiva
  - É preciso garantir que **sempre** vamos chegar nesse caso

Ordenação

Escreva uma função que recebe um vector de números
e informa se ele está ordenado

```
ordenado <- function(vec) {
    for (i in seq_len(length(vec) - 1)) {
        if (vec[i] > vec[i + 1]) { return (FALSE) }
    }
    return (TRUE)
}
```

Ordenação — ideias (bubblesort)

Se a lista está fora de ordem, existe algum elemento de índice i tal que `elemento[i]` > `elemento[i+1]`. Se "consertarmos" todos esses casos, a lista fica ordenada.

```
bubblesort <- function(vec) {
    while (! ordenado(vec)) {
        for (i in seq_len(length(vec) - 1)) {
            if (vec[i] > vec[i + 1]) {
                tmp <- vec[i]
                vec[i] <- vec[i + 1]
                vec[i + 1] <- tmp
            }
        }
    }
}
```

Complexidade computacional

- **Complexidade computacional**
  - O "tamanho do esforço" que o computador deve fazer para resolver um problema de tamanho n
  - Geralmente, estamos preocupados com o tempo de execução, mas às vezes o "esforço" é o consumo de memória ou algum outro aspecto
  - A ideia intuitiva é "mais ou menos proporcional a"
- **Busca linear**
  - O "tamanho do esforço" em termos do tempo de execução é "mais ou menos proporcional" ao tamanho da lista de entrada
    - » *Dizemos que o algoritmo é* **O(n)**
- **Busca binária**
  - O "tamanho do esforço" em termos do tempo de execução é "mais ou menos proporcional" ao número de vezes que podemos dividir a lista de entrada pela metade $\longrightarrow \log_2 n$
    - » *Dizemos que o algoritmo é* **O(log(n))**

Complexidade computacional

- **Em computadores modernos, na maior parte do tempo, a complexidade não importa**
  - Algoritmos com complexidade "ruim" só se tornam perceptivelmente lentos com valores "grandes" de n
  - Nos problemas complexos que envolvem valores grandes de n, você provavelmente vai usar uma biblioteca pronta que implementa uma boa solução para o problema
- **MAS!**
- **Às vezes é preciso levar a complexidade em conta**
  - E é bom saber que existem problemas para os quais não se conhecem soluções com complexidade "boa"

Complexidade dos algoritmos de ordenação

bozosort: NEM IDEIA! 😂

- Não é determinístico (pode não terminar nunca)
- Eu chuto que, para uma dada probabilidade de encontrar uma solução (por exemplo, 99%), a complexidade deve ser mais ou menos proporcional a O(n!)
  - » *Com dez elementos, tende a demorar vários segundos!*

Complexidade dos algoritmos de ordenação

bubblesort: $O(n^2)$

1. Troca a posição de todos os elementos fora do lugar (proporcional a n)
2. Repete até todos os elementos estarem no lugar certo (proporcional a n)

selection sort: $O(n^2)$

1. Passa pela lista procurando o menor elemento (proporcional a n)
2. Repete para todos os elementos da lista (proporcional a n)

mergesort: $O(n \times \log(n))$

1. Divide a lista na metade (proporcional a $\log n$)
2. Para cada par de "fatias", mescla (proporcional a n)

Complexidade dos algoritmos de ordenação

- **É possível demonstrar que não existe algoritmo de ordenação genérico com complexidade melhor que O(n×log(n))**
  - Na verdade, isso se aplica a algoritmos baseados em comparações diretas entre os elementos
  - Existem alguns algoritmos com complexidade O(n), mas eles não fazem comparações diretas entre os elementos; para isso ser possível, eles dependem de limitações específicas, como um número máximo de dígitos em cada número

Números de ponto flutuante e suas pegadinhas

Números de ponto flutuante

- **Em python (e outras poucas linguagens), números inteiros podem ser tão grandes quanto necessário**
  - (Até o limite da memória disponível)
    - » *A menos que você esteja lidando com números muito grandes, cada um deles ocupa pouca memória*
- **Mas, na maioria das linguagens (incluindo R), geralmente há um tamanho máximo**
  - Em R, números até ~2 bilhões, positivo ou negativo (32 bits)
    - » *Pouco! Uma das razões pelas quais inteiros não são muito usados em R*
  - Em geral, números até ~9 quintilhões, positivo ou negativo (64 bits)

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 4.30×10^{-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 e/ou o número máximo possível fica reduzido
  - 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 $\times 10^{\text{expoente}}$**
  - É possível representar números em uma faixa muito maior (em R e na maioria das linguagens, de $\sim 1.7 \times 10^{-308}$ a $\sim 1.7 \times 10^{308}$, positivo ou negativo)
  - A precisão dos números é independente de sua grandeza
    - » *depende apenas do tamanho da mantissa (na maioria das linguagens, 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}**
  - O maior número representável seria 9.999×10^{5}, ou seja, 999.900
- **Não seria possível representar o número 34.567!**
  - Seria preciso usar 3.457×10^{4}, ou seja, 34.570

E como lidar com isso?

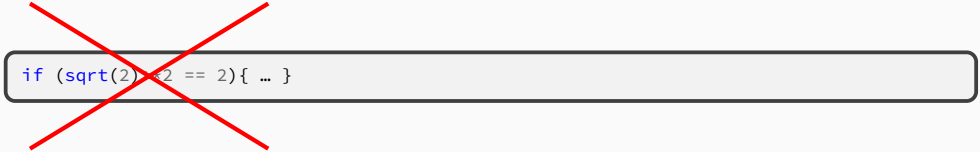
Imprecisões com ponto flutuante

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

```
if (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`)



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`)

```
if (sqrt(2)**2 == 2){ … }
```

```
epsilon <- 1e-15 # É preciso escolher um valor "razoável"
if (abs(sqrt(2)**2 - 2) < epsilon) { … }
```

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**
  - Somas/subtrações com números de grandezas muito diferentes
    - » `cat(format(2e10 + 4e-10, digits=22), "\n")`
      `2e+10` 😱
  - Subtrações com números muito próximos
    - » *O resultado da diferença vai ser predominantemente composto pelos dígitos menos significativos, onde o erro é proporcionalmente maior*
      `cat(format(0.1000000000001, digits=22), "\n")`
      `0.10000000000000999950123` *(erros no dígito 18)*
      `cat(format(0.1000000000001 -0.1, digits=22), "\n")`
      (diferença no dígito 13)
      `9.998946115530316092l9e-14` *(erro no dígito 4)* 😳
- **Somatórios podem fazer as imprecisões se combinarem, tornando erros irrelevantes em catastróficos**
  - Para evitar isso, procure processar do “pequeno” para o “grande”

E como lidar com isso?

Na maior parte do tempo,
não é preciso se preocupar

Dataframes

Modifique este programa para imprimir a lista de alunos em ordem alfabética (usando o algoritmo bubblesort)

```
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
cat("Boletim de notas:\n")
for (i in seq_len(nrow(df))) {
  line <- df[i,]
  cat(unlist(line[c("nome", "nota")]), "\n")
}
```

```
Boletim de notas:
Zé 7
Fê 8.5
Má 8.3
Lú 9.1
```

Dataframes

Modifique este programa para imprimir a lista de alunos em ordem alfabética (usando o algoritmo bubblesort)

```
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
cat("Boletim de notas:\n")
for (i in seq_len(nrow(df))) {
  line <- df[i,]
  cat(unlist(line[c("nome", "nota")]), "\n")
}
```

Dataframes

Modifique este programa para imprimir a lista de alunos em ordem alfabética (usando o algoritmo bubblesort)

```
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- bubblesort(df)
cat("Boletim de notas:\n")
for (i in seq_len(nrow(df))) {
  line <- df[i,]
  cat(unlist(line[c("nome", "nota")]), "\n")
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {









}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {









    return(df)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {




}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {



    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df)    )) {

    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {

    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {
        if (                                    ) {                }
    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {
        if (                                    ) { return (FALSE) }
    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {
        if (df                                  ) { return (FALSE) }
    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {
        if (df[[        ]]                    ) { return (FALSE) }
    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {
        if (df[[i       ]]                      ) { return (FALSE) }
    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {
        if (df[[i,"nome"]]                    ) { return (FALSE) }
    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {









    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {
        if (df[[i,"nome"]] > df[[i + 1, "nome"]]) { return (FALSE) }
    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {
        for (i in seq_len(nrow(df) - 1)) {









        }
    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {
        if (df[[i,"nome"]] > df[[i + 1, "nome"]]) { return (FALSE) }
    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {
        for (i in seq_len(nrow(df) - 1)) {
            if (df[[i,"nome"]] > df[[i + 1, "nome"]]) {









            }
        }
    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {
        if (df[[i,"nome"]] > df[[i + 1, "nome"]]) { return (FALSE) }
    }
    return (TRUE)
}
```

Dataframes

Bubblesort: encontra todos os pares fora de ordem e troca; repete até o conjunto estar ordenado

```
library(tibble)
bubblesort <- function(df) {
    while (! ordenado(df)) {
        for (i in seq_len(nrow(df) - 1)) {
            if (df[[i,"nome"]] > df[[i + 1, "nome"]]) {
                tmp <- df[i,]
                df[i,] <- df[i + 1,]
                df[i + 1,] <- tmp
            }
        }
    }
    return(df)
}
ordenado <- function(df) {
    for (i in seq_len(nrow(df) - 1)) {
        if (df[[i,"nome"]] > df[[i + 1, "nome"]]) { return (FALSE) }
    }
    return (TRUE)
}
```

Dataframes

```r
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))

cat("Boletim de notas:\n")
for (i in seq_len(nrow(df))) {
  line <- df[i,]
  cat(unlist(line[c("nome", "nota")]), "\n")
}
```

Dataframes

```r
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- df[order(df$nome),]
cat("Boletim de notas:\n")
for (i in seq_len(nrow(df))) {
  line <- df[i,]
  cat(unlist(line[c("nome", "nota")]), "\n")
}
```

Dataframes

```
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- df[order(df$nome),]
cat("Boletim de notas:\n")
for (i in seq_len(nrow(df))) {
  line <- df[i,]
  cat(unlist(line[c("nome", "nota")]), "\n")
}
```

`order()` devolve um vector com os índices dos elementos do vector dado em ordem

Dataframes

```
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- df[order(df$nome),]
cat("Boletim de notas:\n")
for (i in seq_len(nrow(df))) {
  line <- df[i,]
  cat(unlist(line[c("nome", "nota")]), "\n")
}
```

`order()` devolve um vector com os índices dos elementos do vector dado em ordem

```
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
cat(order(df$nome), "\n")
```

Dataframes

```
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- df[order(df$nome),]
cat("Boletim de notas:\n")
for (i in seq_len(nrow(df))) {
  line <- df[i,]
  cat(unlist(line[c("nome", "nota")]), "\n")
}
```

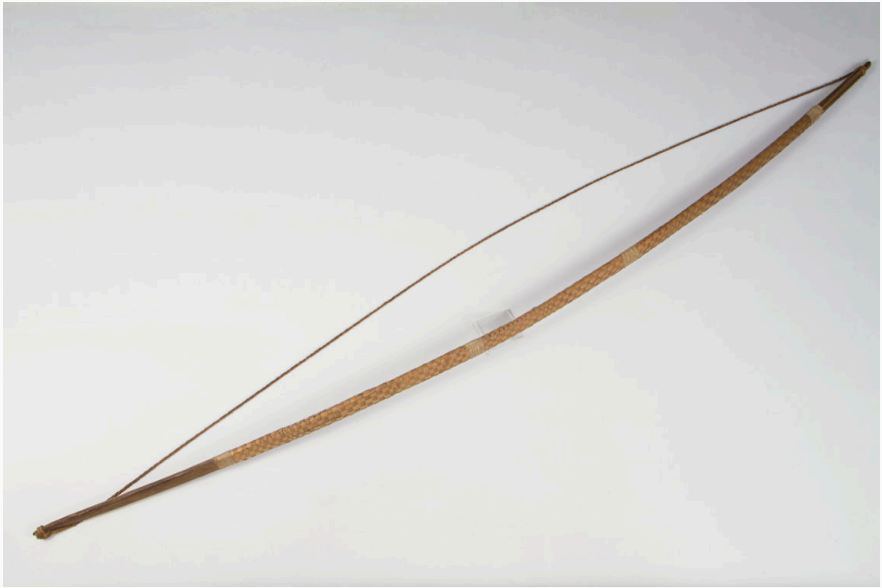
`order()` devolve um vector com os índices dos elementos do vector dado em ordem

```
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
cat(order(df$nome), "\n")
```

```
2 4 3 1
```

And now for something completely different

Um pouco mais de história



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


OSRAM FH 14W/840
Recyclable
Italy
Radium NL 18 W/840
Spectralux Plus
Cool White
Recyclable Germany


Sunbeam

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

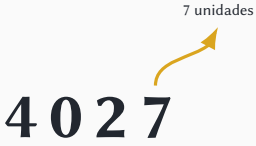
- **Os computadores eletrônicos não são apenas um refinamento das calculadoras e outros dispositivos mecânicos**
- **Três ideias "novas" estão na base dessa diferença:**
  - Aritmética de base 2 (operações numéricas)
  - Álgebra booleana (operações lógicas)
  - Corrente elétrica

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*

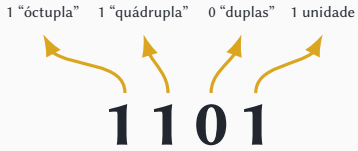
1101

Números de base 2

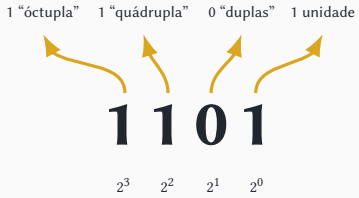
Números de base 2

Números de base 2

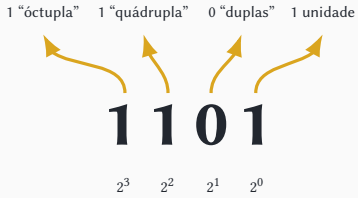
Números de base 2



Números de base 2

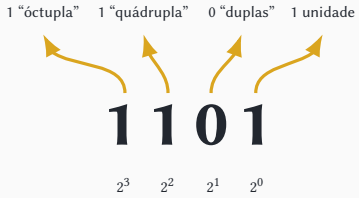


Números de base 2



(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

$$4 + 5$$

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

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 $\longrightarrow$ 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 $\longrightarrow$ 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

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
    - » *Normalmente representamos por* `0xB7`

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
    - » *Normalmente representamos por* `0xB7`
  - Com um pouco de prática, não é difícil fazer as conversões "de cabeça"

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
    - » *Normalmente representamos por* `0xB7`
  - Com um pouco de prática, não é difícil fazer as conversões "de cabeça"
  - É fácil identificar quantos bytes há em um número como `0x41A0CF28`

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
    - » *Normalmente representamos por* `0xB7`
  - Com um pouco de prática, não é difícil fazer as conversões "de cabeça"
  - É fácil identificar quantos bytes há em um número como `0x41A0CF28` (`41 A0 CF 28` ⟶ 4 bytes)

Ponto flutuante no sistema binário

Números de ponto flutuante e representação binária

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

Números de ponto flutuante e representação binária

- **Os números de ponto flutuante também são armazenados no sistema binário, ou seja, $\textbf{mantissa}_2 \times 2^{\textbf{expoente}}$**
  - Em R (e 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 R (e 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 R (e 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)
- **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 R (e 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)
- **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 R (e 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)
- **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 R (e 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)
- **Isso importa porque**
  - Toda dízima periódica é também um "binário periódico"
  - Existem "binários periódicos" que não são dízimas periódicas

```
» cat(format(0.1, digits=22), "\n")
  0.1000000000000000055511
```

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**
- $\frac{1}{50} = \frac{1}{2 \times 5 \times 5}$ **:**

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**
- **$\frac{1}{50} = \frac{1}{2\times5\times5}$: 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**
- **$\frac{1}{50} = \frac{1}{2\times5\times5}$: 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**
- **$\frac{1}{50} = \frac{1}{2\times5\times5}$: 10 é múltiplo de 2 e de 5 $\longrightarrow$ não é dízima periódica**
- **MAS!**
- **2 não é múltiplo de 5 $\longrightarrow$ $\frac{1}{50}$ 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 (George Boole)

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

Álgebra booleana (George Boole)

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

Álgebra booleana (George Boole)

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

Álgebra booleana (George Boole)

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

Álgebra booleana (George Boole)

- **Apenas dois valores**
  - True/1
  - False/0
- **Operações lógicas básicas**
  - Conjunção (`AND`)
  - Disjunção (`OR`)
  - Negação (`NOT`)
- **Leis**
  - Leis de De Morgan
  - Associatividade, comutabilidade, transitividade...

Álgebra booleana (George Boole)

- **Apenas dois valores**
  - True/1
  - False/0
- **Operações lógicas básicas**
  - Conjunção (`AND`)
  - Disjunção (`OR`)
  - Negação (`NOT`)
- **Leis**
  - Leis de De Morgan
  - Associatividade, comutabilidade, transitividade...
- **Lógica proposicional**

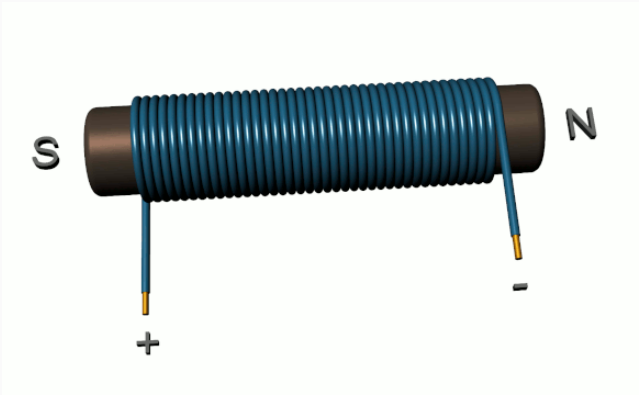
Álgebra booleana (George Boole)

- **Apenas dois valores**
  - True/1
  - False/0
- **Operações lógicas básicas**
  - Conjunção (`AND`)
  - Disjunção (`OR`)
  - Negação (`NOT`)
- **Leis**
  - Leis de De Morgan
  - Associatividade, comutabilidade, transitividade...
- **Lógica proposicional**
- **Variáveis e funções $\longrightarrow$ lógicas de primeira ordem e de ordem superior**

Os matemático pira! 😜

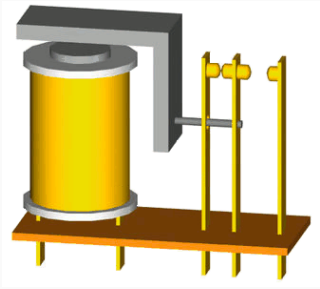
Computadores eletrônicos

- **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

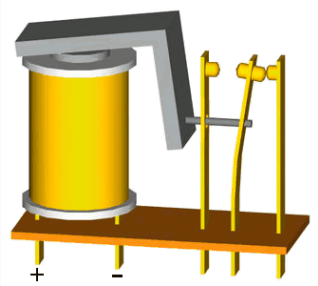


commons.wikimedia.org/wiki/File:Simple_electromagnet2.gif

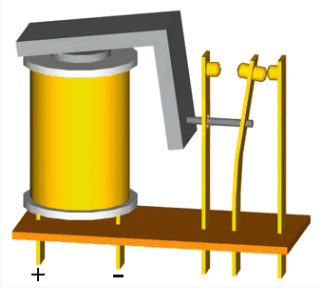
commons.wikimedia.org/wiki/File:Light_Switch.jpg

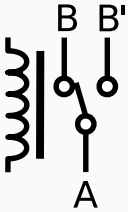


commons.wikimedia.org/wiki/File:Relay_principle_horizontal_new.gif



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





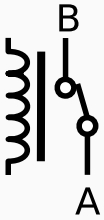
commons.wikimedia.org/wiki/File:Relay_principle_horizontal_new.gif

Operações lógicas com relês

NOT:

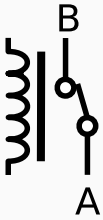
Operações lógicas com relês

NOT:



Operações lógicas com relês

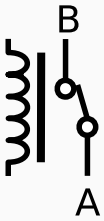
NOT:



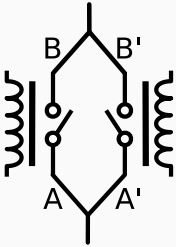
OR:

Operações lógicas com relês

NOT:

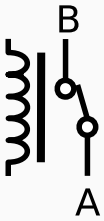


OR:

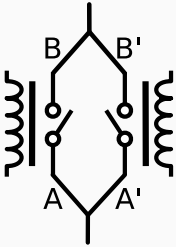


Operações lógicas com relês

NOT:



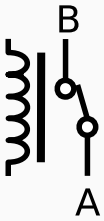
OR:



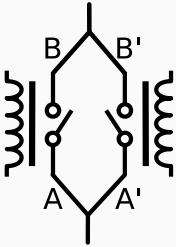
AND:

Operações lógicas com relês

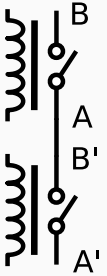
NOT:



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 (Claude Shannon)**

Portas lógicas

- Combinando portas lógicas, podemos implementar operações lógicas complexas (Claude Shannon)
- 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

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0$

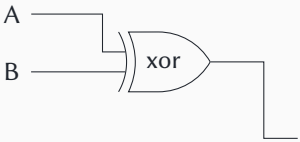
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0$



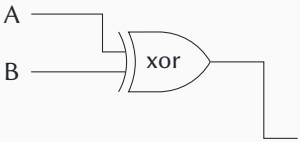
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0 \longrightarrow$ “vai um”



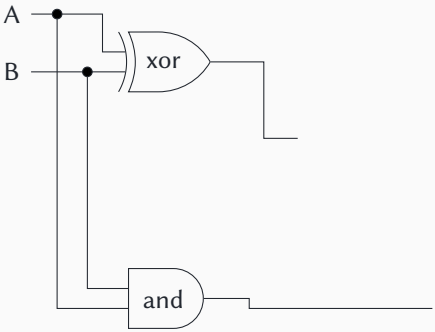
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0 \longrightarrow$ “vai um”



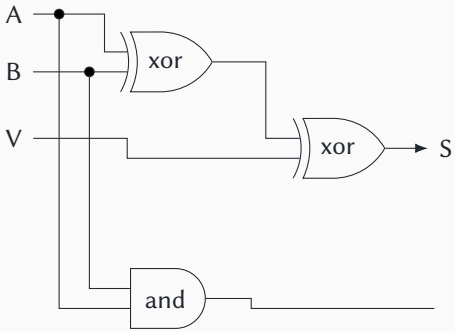
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0 \longrightarrow$ “vai um”



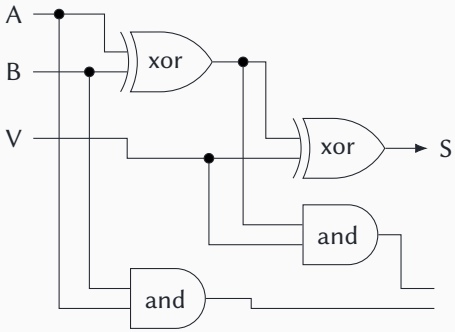
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0 \longrightarrow$ “vai um”



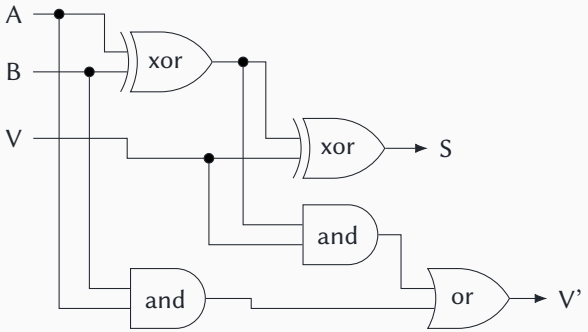
Somando dígitos na base 2 com portas lógicas

$0 + 0 = 0$

$1 + 0 = 1$

$0 + 1 = 1$

$1 + 1 = 0 \longrightarrow$ “vai um”



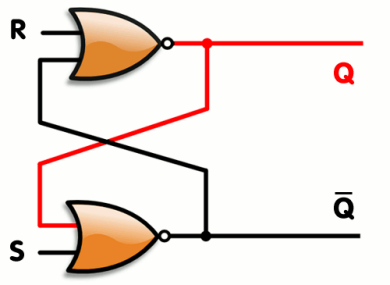
Portas lógicas

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

Portas lógicas

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

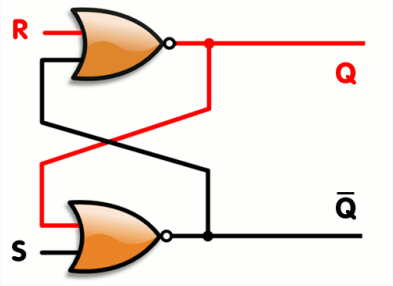
flip-flop



Estável — 1/True

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

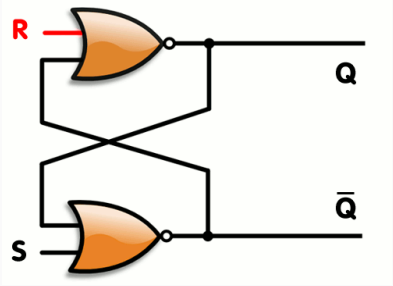
flip-flop



Reset — passando de 1/True para 0/False

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

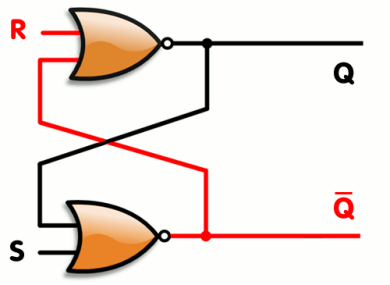
flip-flop



Reset — passando de 1/True para 0/False

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

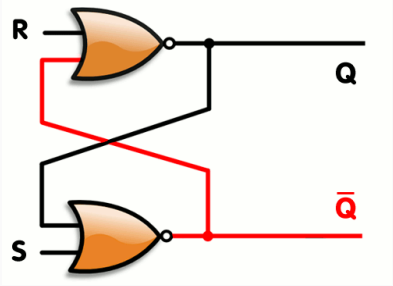
flip-flop



Reset — passando de 1/True para 0/False

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

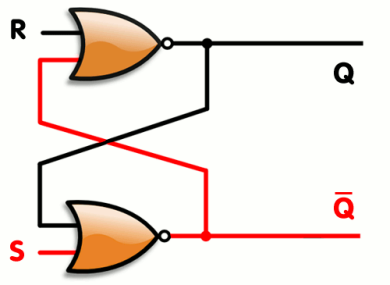
flip-flop



Estável — 0/False

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

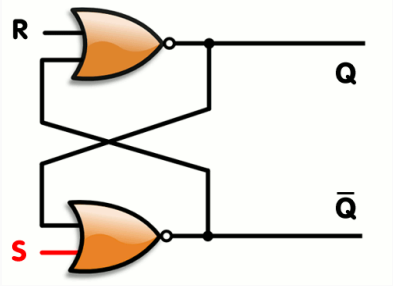
flip-flop



Set — passando de 0/False para 1/True

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

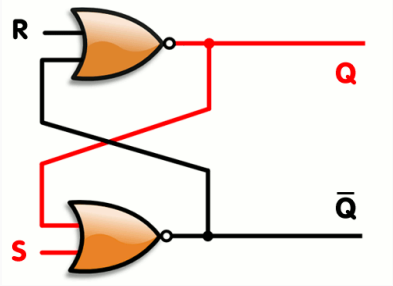
flip-flop



Set — passando de 0/False para 1/True

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

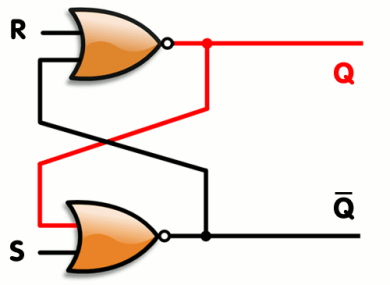
flip-flop



Set — passando de 0/False para 1/True

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

flip-flop



Estável — 1/True

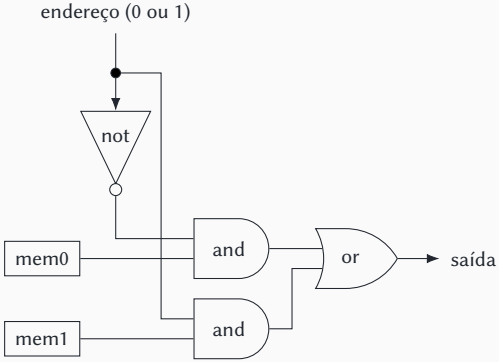
commons.wikimedia.org/wiki/File:R-S_mk2.gif

Endereços de memória

- Também é possível ler um número binário de entrada e reproduzir na saída o conteúdo da memória no endereço correspondente

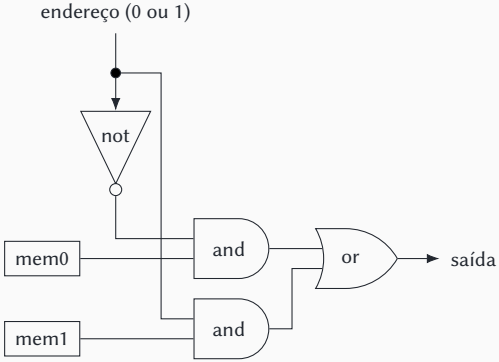
Endereços de memória

- **Também é possível ler um número binário de entrada e reproduzir na saída o conteúdo da memória no endereço correspondente**



Endereços de memória

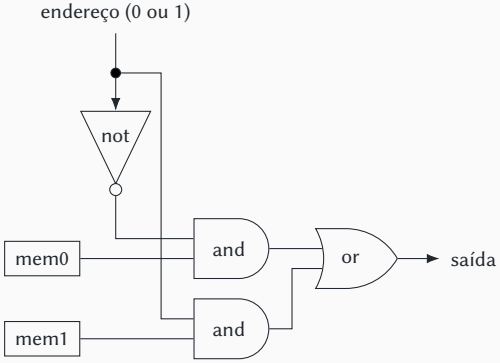
- **Também é possível ler um número binário de entrada e reproduzir na saída o conteúdo da memória no endereço correspondente**



- Ou seja, é possível ler e gravar dados da memória

Endereços de memória

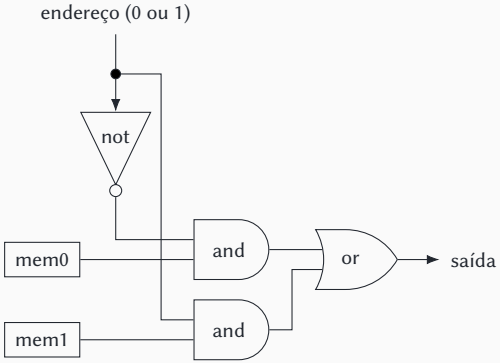
- **Também é possível ler um número binário de entrada e reproduzir na saída o conteúdo da memória no endereço correspondente**



- Ou seja, é possível ler e gravar dados da memória
- Com 8 desses $\longrightarrow$ 2 posições de 1 byte cada

Endereços de memória

- **Também é possível ler um número binário de entrada e reproduzir na saída o conteúdo da memória no endereço correspondente**



- Ou seja, é possível ler e gravar dados da memória
- Com 8 desses ⟶ 2 posições de 1 byte cada
- Com um circuito mais complexo ⟶ mais posições de memória

Entrada e saída

- **Uma impressora mecânica (similar a uma máquina de escrever) pode ser acessada por um endereço de memória**

Entrada e saída

- **Uma impressora mecânica (similar a uma máquina de escrever) pode ser acessada por um endereço de memória**
  - Quando um um número binário é escrito naquele endereço, ela imprime o caractere correspondente no papel

Entrada e saída

- **Uma impressora mecânica (similar a uma máquina de escrever) pode ser acessada por um endereço de memória**
  - Quando um um número binário é escrito naquele endereço, ela imprime o caractere correspondente no papel
- **Uma leitora de cartões perfurados também pode ser acessada por um endereço de memória**

Entrada e saída

- **Uma impressora mecânica (similar a uma máquina de escrever) pode ser acessada por um endereço de memória**
  - Quando um um número binário é escrito naquele endereço, ela imprime o caractere correspondente no papel
- **Uma leitora de cartões perfurados também pode ser acessada por um endereço de memória**
  - O número binário representado pelos bits registrados no cartão pode ser lido naquele endereço

Entrada e saída

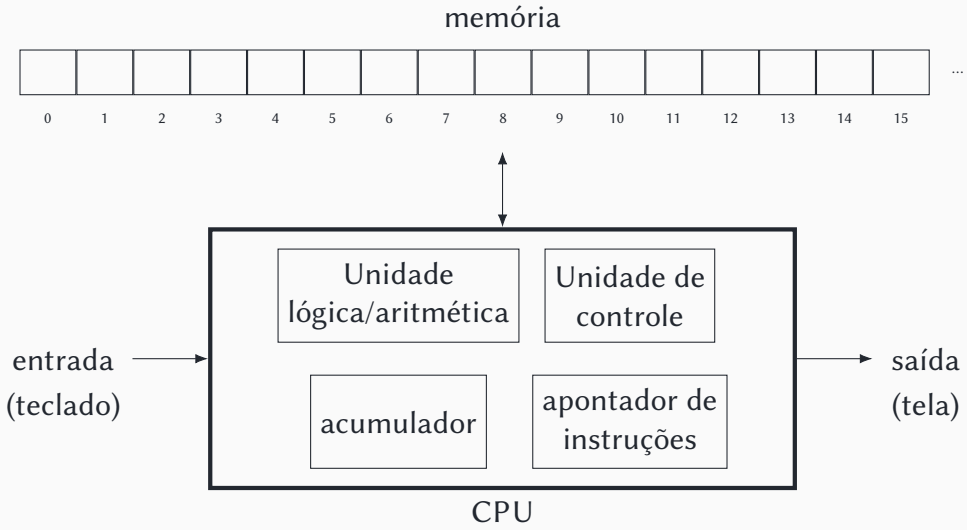
- **Uma impressora mecânica (similar a uma máquina de escrever) pode ser acessada por um endereço de memória**
  - Quando um um número binário é escrito naquele endereço, ela imprime o caractere correspondente no papel
- **Uma leitora de cartões perfurados também pode ser acessada por um endereço de memória**
  - O número binário representado pelos bits registrados no cartão pode ser lido naquele endereço
- **Teclado, tela, mouse, scanner etc. são extensões dessa ideia**

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.) + acumulador**
- **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 com relês

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

Computadores eletromecânicos com relês

- **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 com relês

- **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 fortemente influenciado por Babbage, mas efetivamente usado para o cálculo de tabelas matemáticas e cálculos relacionados à bomba atômica

Computadores eletromecânicos com relês

- **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 fortemente influenciado por Babbage, mas efetivamente usado para o cálculo de tabelas matemáticas e cálculos relacionados à bomba atômica
    - » *Pesava 4,3 toneladas e consumia ~3.7kW*

Computadores eletromecânicos com relês

- **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 fortemente influenciado por Babbage, mas efetivamente usado para o cálculo de tabelas matemáticas e cálculos relacionados à bomba atômica
    - » *Pesava 4,3 toneladas e consumia ~3.7kW*
    - » *Parceria entre a universidade e a IBM*

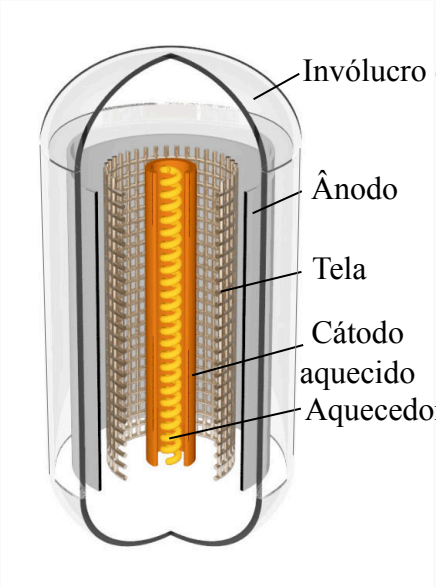
Computadores eletromecânicos com relês

- **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 fortemente influenciado por Babbage, mas efetivamente usado para o cálculo de tabelas matemáticas e cálculos relacionados à bomba atômica
    - » *Pesava 4,3 toneladas e consumia ~3.7kW*
    - » *Parceria entre a universidade e a IBM*
    - » *Repetições eram feitas prendendo o começo da fita de papel com os comandos ao seu fim* ⟶ *"loop" (laço)*

Nenhum deles era de fato Turing-completo

Nenhum deles era de fato Turing-completo

Os computadores eletrônicos (ABC, Colossus, ENIAC e sucessores) tornaram esses sistemas obsoletos quase imediatamente



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

Computadores eletrônicos — válvulas 1/3

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

Computadores eletrônicos — válvulas 1/3

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

Computadores eletrônicos — válvulas 1/3

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

Computadores eletrônicos — válvulas 1/3

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

Computadores eletrônicos — válvulas 1/3

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Muito mais rápidas que relês
  - Consumo muito grande de energia
  - Frágeis e “queimam” com facilidade
- **Atanasoff-Berry Computer (“ABC”, 1942)**

Computadores eletrônicos — válvulas 1/3

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Muito mais rápidas que relês
  - Consumo muito grande de energia
  - Frágeis e "queimam" com facilidade
- **Atanasoff-Berry Computer ("ABC", 1942)**
  - números binários, apenas solucionava sistemas de equações lineares

Computadores eletrônicos — válvulas 1/3

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Muito mais rápidas que relês
  - Consumo muito grande de energia
  - Frágeis e "queimam" com facilidade
- **Atanasoff-Berry Computer ("ABC", 1942)**
  - números binários, apenas solucionava sistemas de equações lineares
- **Colossus Mark I e II (1943/44, reino unido)**

Computadores eletrônicos — válvulas 1/3

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Muito mais rápidas que relês
  - Consumo muito grande de energia
  - Frágeis e "queimam" com facilidade
- **Atanasoff-Berry Computer ("ABC", 1942)**
  - números binários, apenas solucionava sistemas de equações lineares
- **Colossus Mark I e II (1943/44, reino unido)**
  - "programável" (alterando o circuito) mas não Turing-completo, usado para decriptografar mensagens, só foi tornado público nos anos 70

Computadores eletrônicos — válvulas 1/3

- **Válvulas podem realizar as mesmas operações que os relês, mas sem partes móveis**
  - Muito mais rápidas que relês
  - Consumo muito grande de energia
  - Frágeis e "queimam" com facilidade
- **Atanasoff-Berry Computer ("ABC", 1942)**
  - números binários, apenas solucionava sistemas de equações lineares
- **Colossus Mark I e II (1943/44, reino unido)**
  - "programável" (alterando o circuito) mas não Turing-completo, usado para decriptografar mensagens, só foi tornado público nos anos 70
  - 1600 válvulas, consumo de 8,5kW

Computadores eletrônicos — válvulas 2/3

- **ENIAC, o primeiro computador Turing-completo (1945/46)**

Computadores eletrônicos — válvulas 2/3

- **ENIAC, o primeiro computador Turing-completo (1945/46)**
  - ~17.000 válvulas, 30 toneladas, consumo de 174kW

Computadores eletrônicos — válvulas 2/3

- **ENIAC, o primeiro computador Turing-completo (1945/46)**
  - ~17.000 válvulas, 30 toneladas, consumo de 174kW
  - "Programá-lo" envolvia alterar as conexões elétricas do circuito

Computadores eletrônicos — válvulas 2/3

- **ENIAC, o primeiro computador Turing-completo (1945/46)**
  - ~17.000 válvulas, 30 toneladas, consumo de 174kW
  - "Programá-lo" envolvia alterar as conexões elétricas do circuito
    - » *Sistema similar às máquinas Hollerith: um "painel" com o circuito correspondente à operação desejada*

Computadores eletrônicos — válvulas 2/3

- **ENIAC, o primeiro computador Turing-completo (1945/46)**
  - ~17.000 válvulas, 30 toneladas, consumo de 174kW
  - "Programá-lo" envolvia alterar as conexões elétricas do circuito
    - » *Sistema similar às máquinas Hollerith: um "painel" com o circuito correspondente à operação desejada*
    - » *programável pero no mucho!*

Computadores eletrônicos — válvulas 2/3

- **ENIAC, o primeiro computador Turing-completo (1945/46)**
  - ~17.000 válvulas, 30 toneladas, consumo de 174kW
  - "Programá-lo" envolvia alterar as conexões elétricas do circuito
    - » *Sistema similar às máquinas Hollerith: um "painel" com o circuito correspondente à operação desejada*
    - » *programável pero no mucho!*
  - Pouco tempo depois, arquitetura de von Neumann (o programa é armazenado na mesma memória e da mesma forma que os dados)

Computadores eletrônicos — válvulas 2/3

- **ENIAC, o primeiro computador Turing-completo (1945/46)**
  - ~17.000 válvulas, 30 toneladas, consumo de 174kW
  - "Programá-lo" envolvia alterar as conexões elétricas do circuito
    - » *Sistema similar às máquinas Hollerith: um "painel" com o circuito correspondente à operação desejada*
    - » *programável pero no mucho!*
  - Pouco tempo depois, arquitetura de von Neumann (o programa é armazenado na mesma memória e da mesma forma que os dados)
    - » *Programar o sistema envolve apenas ler o código de uma fita ou cartões (ou hoje em dia, do disco)*

Computadores eletrônicos — válvulas 2/3

- **ENIAC, o primeiro computador Turing-completo (1945/46)**
  - ~17.000 válvulas, 30 toneladas, consumo de 174kW
  - "Programá-lo" envolvia alterar as conexões elétricas do circuito
    - » *Sistema similar às máquinas Hollerith: um "painel" com o circuito correspondente à operação desejada*
    - » *programável pero no mucho!*
  - Pouco tempo depois, arquitetura de von Neumann (o programa é armazenado na mesma memória e da mesma forma que os dados)
    - » *Programar o sistema envolve apenas ler o código de uma fita ou cartões (ou hoje em dia, do disco)*
  - Os criadores fundaram uma empresa, que foi comprada pela Remington Rand (fabricante de máquinas de escrever e outros equipamentos de escritório; hoje, Unisys)

Computadores eletrônicos — válvulas 3/3

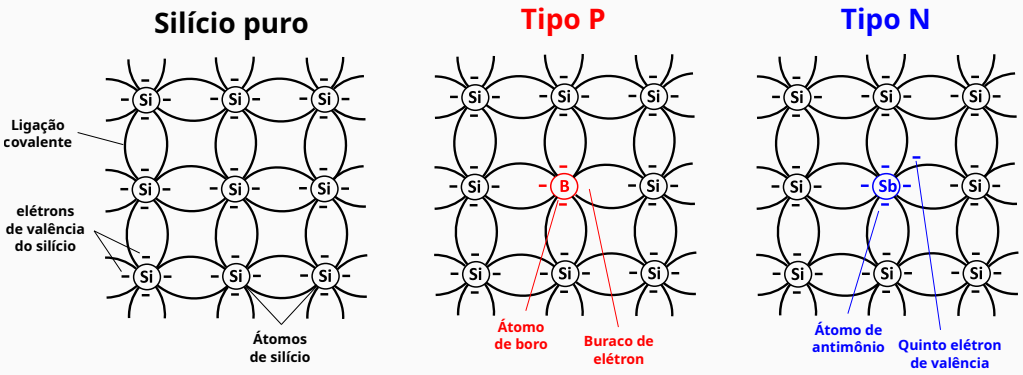
- **Ferranti Mark 1 (1951, Reino Unido), com 4.050 válvulas, pesava 4,5 toneladas e consumia 26kW(?), primeiro computador disponível comercialmente**

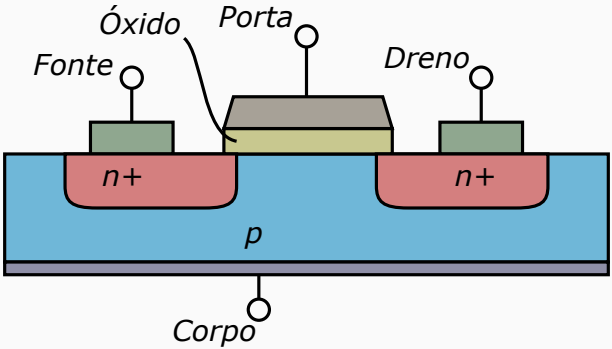
Computadores eletrônicos — válvulas 3/3

- Ferranti Mark 1 (1951, Reino Unido), com 4.050 válvulas, pesava 4,5 toneladas e consumia 26kW(?), primeiro computador disponível comercialmente
- IBM 650 (1954), pesava 2,4–2,8 toneladas e consumia 23–50kW

Computadores eletrônicos — válvulas 3/3

- **Ferranti Mark 1 (1951, Reino Unido), com 4.050 válvulas, pesava 4,5 toneladas e consumia 26kW(?), primeiro computador disponível comercialmente**
- **IBM 650 (1954), pesava 2,4–2,8 toneladas e consumia 23–50kW**
  - "sucesso comercial" (quase 2.000 unidades vendidas em dez anos)


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


Óxido
Porta
Fonte
Dreno
n+
n+
p
Corpo

Computadores eletrônicos — transístores e CIs

- **O transístor (1947) também pode realizar as mesmas operações, mas 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 consome muito menos energia, é bem mais durável e muito menor**
  - PDP-1 (1959, pesava 730kg, primeiro “minicomputador”, berço da “cultura hacker”), NCR 315 (1962, pesava 601kg) e outros

Computadores eletrônicos — transístores e CIs

- **Circuitos pré-prontos genéricos no lugar de componentes individuais e *designs* incompatíveis**

Computadores eletrônicos — transístores e CIs

- **Circuitos pré-prontos genéricos no lugar de componentes individuais e *designs* incompatíveis**
  - IBM System/360, 1964

Computadores eletrônicos — transístores e CIs

- **Circuitos pré-prontos genéricos no lugar de componentes individuais e *designs* incompatíveis**
  - IBM System/360, 1964
    - » *uma só linha de computadores para atender usuários científicos e comerciais*

Computadores eletrônicos — transístores e CIs

- **Circuitos pré-prontos genéricos no lugar de componentes individuais e *designs* incompatíveis**
  - IBM System/360, 1964
    - » *uma só linha de computadores para atender usuários científicos e comerciais*
    - » *Computadores com capacidades de processamento diferentes, mas compatíveis, simplificando upgrades*

Computadores eletrônicos — transístores e CIs

- **Circuitos pré-prontos genéricos no lugar de componentes individuais e *designs* incompatíveis**
  - IBM System/360, 1964
    - » *uma só linha de computadores para atender usuários científicos e comerciais*
    - » *Computadores com capacidades de processamento diferentes, mas compatíveis, simplificando upgrades*
- **Fotolitografia ⟶ é possível fabricar circuitos com vários transístores de uma vez**

Computadores eletrônicos — transístores e CIs

- **Circuitos pré-prontos genéricos no lugar de componentes individuais e *designs* incompatíveis**
  - IBM System/360, 1964
    - » *uma só linha de computadores para atender usuários científicos e comerciais*
    - » *Computadores com capacidades de processamento diferentes, mas compatíveis, simplificando upgrades*
- **Fotolitografia → é possível fabricar circuitos com vários transístores de uma vez**
  - Conjuntos de circuitos integrados padronizados
    - » *computadores de bordo do projeto Apolo, 1966*
    - » *PDP-8/I, 1968 (110kg) e PDP-8/L, 1968 (36kg)*

Computadores eletrônicos — transístores e CIs

- **Circuitos pré-prontos genéricos no lugar de componentes individuais e *designs* incompatíveis**
  - IBM System/360, 1964
    - » *uma só linha de computadores para atender usuários científicos e comerciais*
    - » *Computadores com capacidades de processamento diferentes, mas compatíveis, simplificando upgrades*
- **Fotolitografia → é possível fabricar circuitos com vários transístores de uma vez**
  - Conjuntos de circuitos integrados padronizados
    - » *computadores de bordo do projeto Apolo, 1966*
    - » *PDP-8/I, 1968 (110kg) e PDP-8/L, 1968 (36kg)*
  - Microprocessadores (Intel 4004, 1971)

Computadores eletrônicos — transístores e CIs

- **Circuitos pré-prontos genéricos no lugar de componentes individuais e *designs* incompatíveis**
  - IBM System/360, 1964
    - » *uma só linha de computadores para atender usuários científicos e comerciais*
    - » *Computadores com capacidades de processamento diferentes, mas compatíveis, simplificando upgrades*
- **Fotolitografia → é possível fabricar circuitos com vários transístores de uma vez**
  - Conjuntos de circuitos integrados padronizados
    - » *computadores de bordo do projeto Apolo, 1966*
    - » *PDP-8/I, 1968 (110kg) e PDP-8/L, 1968 (36kg)*
  - Microprocessadores (Intel 4004, 1971)
    - » *Início dos "computadores pessoais" e videogames*

Computadores eletrônicos — transístores e CIs

- **Circuitos pré-prontos genéricos no lugar de componentes individuais e *designs* incompatíveis**
  - IBM System/360, 1964
    - » *uma só linha de computadores para atender usuários científicos e comerciais*
    - » *Computadores com capacidades de processamento diferentes, mas compatíveis, simplificando upgrades*
- **Fotolitografia → é possível fabricar circuitos com vários transístores de uma vez**
  - Conjuntos de circuitos integrados padronizados
    - » *computadores de bordo do projeto Apolo, 1966*
    - » *PDP-8/I, 1968 (110kg) e PDP-8/L, 1968 (36kg)*
  - Microprocessadores (Intel 4004, 1971)
    - » *Início dos "computadores pessoais" e videogames*
    - » *Estamos aqui* 🙂

Armazenamento de dados

- Armazenar programas em cartões perfurados era incômodo, mas funcionava

Armazenamento de dados

- Armazenar programas em cartões perfurados era incômodo, mas funcionava
- Mas é preciso armazenar dados também

Armazenamento de dados

- Armazenar programas em cartões perfurados era incômodo, mas funcionava
- Mas é preciso armazenar dados também
- Imprimir em papel (como as tabelas de logaritmos de Babbage) não é adequado para a maioria dos casos

Armazenamento de dados

- **Armazenar programas em cartões perfurados era incômodo, mas funcionava**
- **Mas é preciso armazenar dados também**
- **Imprimir em papel (como as tabelas de logaritmos de Babbage) não é adequado para a maioria dos casos**
  - Por exemplo, saldo bancário dos clientes no final do dia

Armazenamento de dados

- **Armazenar programas em cartões perfurados era incômodo, mas funcionava**
- **Mas é preciso armazenar dados também**
- **Imprimir em papel (como as tabelas de logaritmos de Babbage) não é adequado para a maioria dos casos**
  - Por exemplo, saldo bancário dos clientes no final do dia
  - No dia seguinte, lê os dados do dia anterior, atualiza e guarda o novo valor

Armazenamento de dados

- **Armazenar programas em cartões perfurados era incômodo, mas funcionava**
- **Mas é preciso armazenar dados também**
- **Imprimir em papel (como as tabelas de logaritmos de Babbage) não é adequado para a maioria dos casos**
  - Por exemplo, saldo bancário dos clientes no final do dia
  - No dia seguinte, lê os dados do dia anterior, atualiza e guarda o novo valor
- **Memória não-volátil**

Armazenamento de dados

- **Armazenar programas em cartões perfurados era incômodo, mas funcionava**
- **Mas é preciso armazenar dados também**
- **Imprimir em papel (como as tabelas de logaritmos de Babbage) não é adequado para a maioria dos casos**
  - Por exemplo, saldo bancário dos clientes no final do dia
  - No dia seguinte, lê os dados do dia anterior, atualiza e guarda o novo valor
- **Memória não-volátil**
  - Fitas magnéticas $\longrightarrow$ acesso sequencial

Armazenamento de dados

- **Armazenar programas em cartões perfurados era incômodo, mas funcionava**
- **Mas é preciso armazenar dados também**
- **Imprimir em papel (como as tabelas de logaritmos de Babbage) não é adequado para a maioria dos casos**
  - Por exemplo, saldo bancário dos clientes no final do dia
  - No dia seguinte, lê os dados do dia anterior, atualiza e guarda o novo valor
- **Memória não-volátil**
  - Fitas magnéticas ⟶ acesso sequencial
  - Disquetes, HDs, CDs/DVDs, SSDs... ⟶ acesso aleatório

Passo

Sistemas analógicos e digitais

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

Sistemas analógicos e digitais

- **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 e digitais

- **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 e digitais

- **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 e digitais

- **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 e digitais

- **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

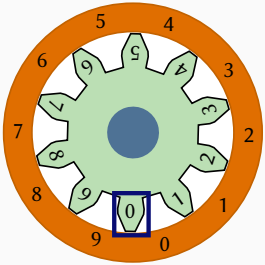
Sistemas analógicos e digitais

- **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**

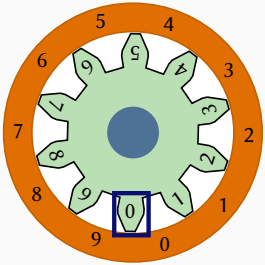
Sistemas analógicos e digitais

- **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

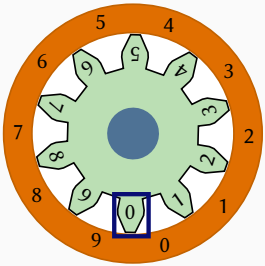


Pascalina passo a passo



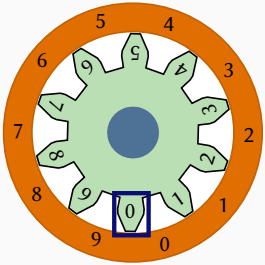
- Cada vez que movemos a engrenagem, realizamos um **passo**

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**

Sistemas analógicos e digitais

- O mecanismo de Anticítera era um computador **analógico**

Sistemas analógicos e digitais

- O mecanismo de Anticítera era um computador **analógico**
- A régua de cálculo é um instrumento **analógico**

Sistemas analógicos e digitais

- O mecanismo de Anticítera era um computador **analógico**
- A régua de cálculo é um instrumento **analógico**
- O ábaco é um instrumento **digital**

Sistemas analógicos e digitais

- O mecanismo de Anticítera era um computador **analógico**
- A régua de cálculo é um instrumento **analógico**
- O ábaco é um instrumento **digital**
- As calculadoras mecânicas são sistemas **digitais**

Sistemas analógicos e digitais

- O mecanismo de Anticítera era um computador **analógico**
- A régua de cálculo é um instrumento **analógico**
- O ábaco é um instrumento **digital**
- As calculadoras mecânicas são sistemas **digitais**

Sistemas digitais têm a noção de passo

Passo em calculadoras mecânicas

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

Passo em calculadoras mecânicas

- **Calculadoras mecânicas como o Arithmomètre (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Um passo $\longrightarrow$ um giro completo de uma manivela

Passo em calculadoras mecânicas

- **Calculadoras mecânicas como o Arithmomètre (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Um passo $\longrightarrow$ um giro completo de uma manivela
- **Existe uma velocidade máxima para a execução dos passos**

Passo em calculadoras mecânicas

- **Calculadoras mecânicas como o Arithmomètre (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Um passo $\longrightarrow$ um giro completo de uma manivela
- **Existe uma velocidade máxima para a execução dos passos**
  - Potência necessária para girar a manivela

Passo em calculadoras mecânicas

- **Calculadoras mecânicas como o Arithmomètre (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Um passo $\longrightarrow$ um giro completo de uma manivela
- **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 como o Arithmomètre (e a máquina analítica) processavam todos os dígitos em um único passo**
  - Um passo $\longrightarrow$ um giro completo de uma manivela
- **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 usam passos**

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também usam passos**
  - O estado do sistema não faz sentido quando um relê está ligando/desligando

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também usam passos**
  - O estado do sistema não faz sentido quando um relê está ligando/desligando
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica não é instantânea)*

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também usam passos**
  - O estado do sistema não faz sentido quando um relê está ligando/desligando
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica não é instantânea)*
  - Só faz sentido "ler" uma nova entrada depois que a anterior foi processada até o fim (o resultado pode fazer parte da próxima entrada)

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também usam passos**
  - O estado do sistema não faz sentido quando um relê está ligando/desligando
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica não é instantânea)*
  - Só faz sentido "ler" uma nova entrada depois que a anterior foi processada até o fim (o resultado pode fazer parte da próxima entrada)
  - Um circuito pode ser usado em dois passos seguidos, então só é possível executar o segundo após o primeiro terminar

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também usam passos**
  - O estado do sistema não faz sentido quando um relê está ligando/desligando
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica não é instantânea)*
  - Só faz sentido "ler" uma nova entrada depois que a anterior foi processada até o fim (o resultado pode fazer parte da próxima entrada)
  - Um circuito pode ser usado em dois passos seguidos, então só é possível executar o segundo após o primeiro terminar
- **Existe uma velocidade máxima para a execução dos passos**

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também usam passos**
  - O estado do sistema não faz sentido quando um relê está ligando/desligando
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica não é instantânea)*
  - Só faz sentido "ler" uma nova entrada depois que a anterior foi processada até o fim (o resultado pode fazer parte da próxima entrada)
  - Um circuito pode ser usado em dois passos seguidos, então só é possível executar o segundo após o primeiro terminar
- **Existe uma velocidade máxima para a execução dos passos**
  - Por exemplo, uma impressora que imprime um caracter por vez só pode ler o próximo após terminar de imprimir o anterior

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também usam passos**
  - O estado do sistema não faz sentido quando um relê está ligando/desligando
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica não é instantânea)*
  - Só faz sentido "ler" uma nova entrada depois que a anterior foi processada até o fim (o resultado pode fazer parte da próxima entrada)
  - Um circuito pode ser usado em dois passos seguidos, então só é possível executar o segundo após o primeiro terminar
- **Existe uma velocidade máxima para a execução dos passos**
  - Por exemplo, uma impressora que imprime um caracter por vez só pode ler o próximo após terminar de imprimir o anterior
  - Passos mais rápidos consomem mais energia e geram mais calor

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também usam passos**
  - O estado do sistema não faz sentido quando um relê está ligando/desligando
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica não é instantânea)*
  - Só faz sentido "ler" uma nova entrada depois que a anterior foi processada até o fim (o resultado pode fazer parte da próxima entrada)
  - Um circuito pode ser usado em dois passos seguidos, então só é possível executar o segundo após o primeiro terminar
- **Existe uma velocidade máxima para a execução dos passos**
  - Por exemplo, uma impressora que imprime um caracter por vez só pode ler o próximo após terminar de imprimir o anterior
  - Passos mais rápidos consomem mais energia e geram mais calor
- **Existe um "metrônomo" (*clock*, no sentido de "tic-tac") que controla a velocidade dos passos**

Passo em computadores eletromecânicos e eletrônicos

- **Computadores eletrônicos também usam passos**
  - O estado do sistema não faz sentido quando um relê está ligando/desligando
    - » *(mesmo em sistemas totalmente eletrônicos, a corrente elétrica não é instantânea)*
  - Só faz sentido "ler" uma nova entrada depois que a anterior foi processada até o fim (o resultado pode fazer parte da próxima entrada)
  - Um circuito pode ser usado em dois passos seguidos, então só é possível executar o segundo após o primeiro terminar
- **Existe uma velocidade máxima para a execução dos passos**
  - Por exemplo, uma impressora que imprime um caracter por vez só pode ler o próximo após terminar de imprimir o anterior
  - Passos mais rápidos consomem mais energia e geram mais calor
- **Existe um "metrônomo" (*clock*, no sentido de "tic-tac") que controla a velocidade dos passos**
  - "Meu computador é de 3.2GHz"

Como fazer computadores mais rápidos?

Como fazer computadores mais rápidos?

1. Aumentar a velocidade dos passos (*clock*)

Como fazer computadores mais rápidos?

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

Como fazer computadores mais rápidos?

1. **Aumentar a velocidade dos passos (*clock*)**
   - Um tanto óbvio 😛
   - Circuitos mais complexos
     - » *(como veremos mais adiante)*

Como fazer computadores mais rápidos?

1. **Aumentar a velocidade dos passos (*clock*)**
   - Um tanto óbvio 😛
   - Circuitos mais complexos
     - » *(como veremos mais adiante)*
2. **Fazer operações complexas em um único passo**

Como fazer computadores mais rápidos?

1. **Aumentar a velocidade dos passos (*clock*)**
   - Um tanto óbvio 😛
   - Circuitos mais complexos
     - » *(como veremos mais adiante)*
2. **Fazer operações complexas em um único passo**
   - Circuitos mais complexos

Como fazer computadores mais rápidos?

1. **Aumentar a velocidade dos passos (*clock*)**
   - Um tanto óbvio 😛
   - Circuitos mais complexos
     - » *(como veremos mais adiante)*
2. **Fazer operações complexas em um único passo**
   - Circuitos mais complexos
3. **Fazer várias coisas ao mesmo tempo**

Como fazer computadores mais rápidos?

1. **Aumentar a velocidade dos passos (*clock*)**
   - Um tanto óbvio 😛
   - Circuitos mais complexos
     - » *(como veremos mais adiante)*
2. **Fazer operações complexas em um único passo**
   - Circuitos mais complexos
3. **Fazer várias coisas ao mesmo tempo**
   - Circuitos mais complexos

Lei de Moore

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

Lei de Moore

- O processador 8088 (1979), usado nos primeiros PCs da IBM, tinha 29.000 transístores
- O Athlon 64 (2007) tinha 122 milhões

Lei de Moore

- 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

Lei de Moore

- 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
- Lei de Moore: “o número de componentes em um circuito integrado dobra a cada dois anos”

Lei de Moore

- 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
- Lei de Moore: "o número de componentes em um circuito integrado dobra a cada dois anos"
  - O ovo ou a galinha?

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**
  - Limitações físicas (velocidade de comunicação), de custo, de temperatura, de tamanho, de consumo energético...

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**
  - Limitações físicas (velocidade de comunicação), de custo, de temperatura, de tamanho, de consumo energético...
  - Nem todos os "pedaços" do computador podem operar na velocidade máxima

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**
  - Limitações físicas (velocidade de comunicação), de custo, de temperatura, de tamanho, de consumo energético...
  - Nem todos os "pedaços" do computador podem operar na velocidade máxima
  - Clocks diferentes

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**
  - Limitações físicas (velocidade de comunicação), de custo, de temperatura, de tamanho, de consumo energético...
  - Nem todos os "pedaços" do computador podem operar na velocidade máxima
  - Clocks diferentes
  - Memória "dentro" do processador (registradores) funciona no clock do processador

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**
  - Limitações físicas (velocidade de comunicação), de custo, de temperatura, de tamanho, de consumo energético...
  - Nem todos os "pedaços" do computador podem operar na velocidade máxima
  - Clocks diferentes
  - Memória "dentro" do processador (registradores) funciona no clock do processador
  - Memória adicional é muito maior, mas usa um clock mais lento

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**
  - Limitações físicas (velocidade de comunicação), de custo, de temperatura, de tamanho, de consumo energético...
  - Nem todos os "pedaços" do computador podem operar na velocidade máxima
  - Clocks diferentes
  - Memória "dentro" do processador (registradores) funciona no clock do processador
  - Memória adicional é muito maior, mas usa um clock mais lento
  - Processador precisa "esperar" para acessar a memória adicional

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**
  - Limitações físicas (velocidade de comunicação), de custo, de temperatura, de tamanho, de consumo energético...
  - Nem todos os "pedaços" do computador podem operar na velocidade máxima
  - Clocks diferentes
  - Memória "dentro" do processador (registradores) funciona no clock do processador
  - Memória adicional é muito maior, mas usa um clock mais lento
  - Processador precisa "esperar" para acessar a memória adicional
    - » *Memória cache*

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**
  - Limitações físicas (velocidade de comunicação), de custo, de temperatura, de tamanho, de consumo energético...
  - Nem todos os "pedaços" do computador podem operar na velocidade máxima
  - Clocks diferentes
  - Memória "dentro" do processador (registradores) funciona no clock do processador
  - Memória adicional é muito maior, mas usa um clock mais lento
  - Processador precisa "esperar" para acessar a memória adicional
    - » *Memória cache*
    - » *Leitura antecipada*

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**
  - Limitações físicas (velocidade de comunicação), de custo, de temperatura, de tamanho, de consumo energético...
  - Nem todos os "pedaços" do computador podem operar na velocidade máxima
  - Clocks diferentes
  - Memória "dentro" do processador (registradores) funciona no clock do processador
  - Memória adicional é muito maior, mas usa um clock mais lento
  - Processador precisa "esperar" para acessar a memória adicional
    - » *Memória cache*
    - » *Leitura antecipada*
    - » *Processador precisa de fato "esperar" em algo como 10% dos acessos à memória*

Como fazer computadores mais rápidos?

- **Aumentar a velocidade dos passos**
  - Limitações físicas (velocidade de comunicação), de custo, de temperatura, de tamanho, de consumo energético...
  - Nem todos os "pedaços" do computador podem operar na velocidade máxima
  - Clocks diferentes
  - Memória "dentro" do processador (registradores) funciona no clock do processador
  - Memória adicional é muito maior, mas usa um clock mais lento
  - Processador precisa "esperar" para acessar a memória adicional
    - » *Memória cache*
    - » *Leitura antecipada*
    - » *Processador precisa de fato "esperar" em algo como 10% dos acessos à memória*
  - O mesmo princípio vale para discos, teclado/tela, rede...

Como fazer computadores mais rápidos?

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

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - Podemos realizar multiplicações como uma série de adições...

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 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**
  - 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**
  - 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**
  - 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, "central" de automóveis...*

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 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, "central" de automóveis...*
  - Coprocessadores dedicados

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 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, "central" de automóveis...*
  - Coprocessadores dedicados
    - » *Digidesign Pro Tools*

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 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, "central" de automóveis...*
  - Coprocessadores dedicados
    - » *Digidesign Pro Tools*
    - » *GPUs (cálculos vetoriais – SIMD)*

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 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, "central" de automóveis...*
  - Coprocessadores dedicados
    - » *Digidesign Pro Tools*
    - » *GPUs (cálculos vetoriais – SIMD)*
    - » *Apple Silicon (M1/M2/M3...)*

Como fazer computadores mais rápidos?

- **Fazer operações complexas em um único passo**
  - 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, "central" de automóveis...*
  - Coprocessadores dedicados
    - » *Digidesign Pro Tools*
    - » *GPUs (cálculos vetoriais – SIMD)*
    - » *Apple Silicon (M1/M2/M3...)*
    - » *FPGAs ("hardware programável")*

Como fazer computadores mais rápidos?

- **Fazer várias coisas ao mesmo tempo**

Como fazer computadores mais rápidos?

- **Fazer várias coisas ao mesmo tempo**
  - Linha de produção (*pipelining*)

Como fazer computadores mais rápidos?

- **Fazer várias coisas ao mesmo tempo**
  - Linha de produção (*pipelining*)
  - Execução especulativa e fora de ordem

Linha de produção (*Pipelining*)

1. Lê instrução da memória

Linha de produção (*Pipelining*)

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

Linha de produção (*Pipelining*)

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

Linha de produção (*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

Linha de produção (*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!

Linha de produção (*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

Linha de produção (*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

Linha de produção (*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

Linha de produção (*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?

- **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**
- **Coprocessadores sofrem com o atraso da comunicação**

Como fazer computadores mais rápidos?

- **Pipelining, execução especulativa e fora de ordem já são amplamente usados**
- **Coprocessadores sofrem com o atraso da comunicação**
- ***Clock* mais rápido gasta mais energia**

Como fazer computadores mais rápidos?

- **Pipelining, execução especulativa e fora de ordem já são amplamente usados**
- **Coprocessadores sofrem com o atraso da comunicação**
- ***Clock* mais rápido gasta mais energia**
- **Circuitos mais complexos são maiores e também gastam mais energia**

Como fazer computadores mais rápidos?

- **Pipelining, execução especulativa e fora de ordem já são amplamente usados**
- **Coprocessadores sofrem com o atraso da comunicação**
- ***Clock* mais rápido gasta mais energia**
- **Circuitos mais complexos são maiores e também gastam mais energia**
- **Mais energia $\longrightarrow$ mais calor**

Como fazer computadores mais rápidos?

- **Pipelining, execução especulativa e fora de ordem já são amplamente usados**
- **Coprocessadores sofrem com o atraso da comunicação**
- ***Clock* mais rápido gasta mais energia**
- **Circuitos mais complexos são maiores e também gastam mais energia**
- **Mais energia → mais calor**
  - existe um limite de temperatura viável!

Como fazer computadores mais rápidos?

- A miniaturização permite reduzir a voltagem do circuito $\longrightarrow$ circuitos mais complexos e com *clock* maior sem aumentar o tamanho e o gasto energético

Como fazer computadores mais rápidos?

- **A miniaturização permite reduzir a voltagem do circuito $\longrightarrow$ circuitos mais complexos e com *clock* maior sem aumentar o tamanho e o gasto energético**
  - Mas já chegamos nos limites mínimo da voltagem e máximo do *clock* por volta de 2005 😭

Como fazer computadores mais rápidos?

- **A miniaturização permite reduzir a voltagem do circuito $\longrightarrow$ circuitos mais complexos e com *clock* maior sem aumentar o tamanho e o gasto energético**
  - Mas já chegamos nos limites mínimo da voltagem e máximo do *clock* por volta de 2005 😭
  - Na fotolitografia, é preciso usar luz ultravioleta ou raios X porque o tamanho dos elementos no circuito é menor que o comprimento de onda da luz visível

Como fazer computadores mais rápidos?

- **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 nos limites mínimo da voltagem e máximo do *clock* por volta de 2005 😭
  - Na fotolitografia, é preciso usar luz ultravioleta ou raios X porque o tamanho dos elementos no circuito é menor que o comprimento de onda da luz visível
    - » *Estamos nos aproximando do limite da miniaturização*

Como fazer computadores mais rápidos?

- **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 nos limites mínimo da voltagem e máximo do *clock* por volta de 2005 😭
  - Na fotolitografia, é preciso usar luz ultravioleta ou raios X porque o tamanho dos elementos no circuito é menor que o comprimento de onda da luz visível
    - » *Estamos nos aproximando do limite da miniaturização*

E agora?! O que ainda dá para fazer?

Como fazer computadores mais rápidos?

- Todas essas técnicas acontecem “escondidas” do programador

Como fazer computadores mais rápidos?

- **Todas essas técnicas acontecem “escondidas” do programador**
  - O programador pode fazer de conta que todos os passos acontecem sequencialmente e de maneira síncrona

Como fazer computadores mais rápidos?

- **Todas essas técnicas acontecem "escondidas" do programador**
  - O programador pode fazer de conta que todos os passos acontecem sequencialmente e de maneira síncrona
    - » *Muito mais fácil de escrever os programas*

Como fazer computadores mais rápidos?

- **Todas essas técnicas acontecem "escondidas" do programador**
  - O programador pode fazer de conta que todos os passos acontecem sequencialmente e de maneira síncrona
    - » *Muito mais fácil de escrever os programas*
- **O próximo passo é fazer mais coisas ao mesmo tempo com programas explicitamente criados para isso**

Como fazer computadores mais rápidos?

- **Processamento (explicitamente) paralelo (com vários processadores)**

Como fazer computadores mais rápidos?

- **Processamento (explicitamente) paralelo (com vários processadores)**
  - Em servidores, podem atender vários usuários

Como fazer computadores mais rápidos?

- **Processamento (explicitamente) paralelo (com vários processadores)**
  - Em servidores, podem atender vários usuários
    - » *Este é um caso "simples" para o programador; a complexidade fica por conta do sistema operacional*

Como fazer computadores mais rápidos?

- **Processamento (explicitamente) paralelo (com vários processadores)**
  - Em servidores, podem atender vários usuários
    - » *Este é um caso "simples" para o programador; a complexidade fica por conta do sistema operacional*
  - 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?

- **Processamento (explicitamente) paralelo (com vários processadores)**
  - Em servidores, podem atender vários usuários
    - » *Este é um caso "simples" para o programador; a complexidade fica por conta do sistema operacional*
  - Em computadores pessoais, divisão de tarefas; por exemplo, cada uma das imagens em uma página web pode ser renderizada por um processador

MAS...

Como fazer computadores mais rápidos?

- **Processamento (explicitamente) paralelo (com vários processadores)**
  - Em servidores, podem atender vários usuários
    - » *Este é um caso "simples" para o programador; a complexidade fica por conta do sistema operacional*
  - Em computadores pessoais, divisão de tarefas; por exemplo, cada uma das imagens em uma página web pode ser renderizada por um processador

MAS...

- **Nem tudo pode ser paralelizado**

Como fazer computadores mais rápidos?

- **Processamento (explicitamente) paralelo (com vários processadores)**
  - Em servidores, podem atender vários usuários
    - » *Este é um caso "simples" para o programador; a complexidade fica por conta do sistema operacional*
  - Em computadores pessoais, divisão de tarefas; por exemplo, cada uma das imagens em uma página web pode ser renderizada por um processador

MAS...

- **Nem tudo pode ser paralelizado**
- **Quanto mais aumentamos o paralelismo, menor o ganho**

Como fazer computadores mais rápidos?

- **Processamento (explicitamente) paralelo (com vários processadores)**
  - Em servidores, podem atender vários usuários
    - » *Este é um caso "simples" para o programador; a complexidade fica por conta do sistema operacional*
  - Em computadores pessoais, divisão de tarefas; por exemplo, cada uma das imagens em uma página web pode ser renderizada por um processador

MAS...

- **Nem tudo pode ser paralelizado**
- **Quanto mais aumentamos o paralelismo, menor o ganho**
- **Programar aplicações paralelas é significativamente mais complexo**

E o software?

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)

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 esperar o processamento anterior terminar e, em seguida, carregar (manualmente) o novo programa

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia esperar o processamento anterior terminar e, em seguida, carregar (manualmente) o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia esperar o processamento anterior terminar e, em seguida, carregar (manualmente) 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 esperar o processamento anterior terminar e, em seguida, carregar (manualmente) o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Ambiente de programação padronizado

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia esperar o processamento anterior terminar e, em seguida, carregar (manualmente) o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Ambiente de programação padronizado
    - » *Drivers (acesso simplificado e unificado para dispositivos similares de fabricantes diferentes)*

Software — Sistemas Operacionais

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia esperar o processamento anterior terminar e, em seguida, carregar (manualmente) o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Ambiente de programação padronizado
    - » *Drivers (acesso simplificado e unificado para dispositivos similares de fabricantes diferentes)*
    - » *Sistemas de arquivos (mecanismo de acesso padronizado aos dados)*

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia esperar o processamento anterior terminar e, em seguida, carregar (manualmente) o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Ambiente de programação padronizado
    - » *Drivers (acesso simplificado e unificado para dispositivos similares de fabricantes diferentes)*
    - » *Sistemas de arquivos (mecanismo de acesso padronizado aos dados)*
    - » *Bibliotecas e interface com o usuário padronizada*

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia esperar o processamento anterior terminar e, em seguida, carregar (manualmente) o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Ambiente de programação padronizado
    - » *Drivers (acesso simplificado e unificado para dispositivos similares de fabricantes diferentes)*
    - » *Sistemas de arquivos (mecanismo de acesso padronizado aos dados)*
    - » *Bibliotecas e interface com o usuário padronizada*
  - Sistema para executar vários programas ao mesmo tempo (*time sharing*)

Software — Sistemas Operacionais

- **Sistemas Operacionais**
  - Nos anos 1950, executar programas diferentes envolvia esperar o processamento anterior terminar e, em seguida, carregar (manualmente) o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Ambiente de programação padronizado
    - » *Drivers (acesso simplificado e unificado para dispositivos similares de fabricantes diferentes)*
    - » *Sistemas de arquivos (mecanismo de acesso padronizado aos dados)*
    - » *Bibliotecas e interface com o usuário padronizada*
  - 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 esperar o processamento anterior terminar e, em seguida, carregar (manualmente) o novo programa
    - » *Dado o custo dos computadores, isso era um desperdício*
  - Sistemas para carregar automaticamente o "próximo" programa
  - Ambiente de programação padronizado
    - » *Drivers (acesso simplificado e unificado para dispositivos similares de fabricantes diferentes)*
    - » *Sistemas de arquivos (mecanismo de acesso padronizado aos dados)*
    - » *Bibliotecas e interface com o usuário padronizada*
  - 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*

Software — Técnicas e Gestão

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

Software — Técnicas e Gestão

- **Como produzir programas complexos com qualidade e em tempo razoável?**
  - Prazos
  - Custos
  - Divisão do trabalho
  - ...

Software — Técnicas e Gestão

- **Como produzir programas complexos com qualidade e em tempo razoável?**
  - Prazos
  - Custos
  - Divisão do trabalho
  - ...
- **Engenharia de Software**

Software — Técnicas e Gestão

- **Como produzir programas complexos com qualidade e em tempo razoável?**
  - Prazos
  - Custos
  - Divisão do trabalho
  - ...
- **Engenharia de Software**
  - Ferramentas de apoio

Software — Técnicas e Gestão

- **Como produzir programas complexos com qualidade e em tempo razoável?**
  - Prazos
  - Custos
  - Divisão do trabalho
  - ...
- **Engenharia de Software**
  - Ferramentas de apoio
  - Técnicas de programação

Software — Técnicas e Gestão

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

Software — Técnicas e Gestão

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

Software — Técnicas e Gestão

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

Software — Técnicas e Gestão

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

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*
    - » *Videogames*

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*
    - » *Videogames*
    - » *Sistemas de controle*

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*
    - » *Videogames*
    - » *Sistemas de controle*
    - » ...

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*
    - » *Videogames*
    - » *Sistemas de controle*
    - » ...
- **Sistemas de informação**
  - Coleta, armazenamento e recuperação da informação

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*
    - » *Videogames*
    - » *Sistemas de controle*
    - » ...
- **Sistemas de informação**
  - Coleta, armazenamento e recuperação da informação
    - » *Segurança da informação (comércio e governo eletrônico, informática médica...)*

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*
    - » *Videogames*
    - » *Sistemas de controle*
    - » ...
- **Sistemas de informação**
  - Coleta, armazenamento e recuperação da informação
    - » *Segurança da informação (comércio e governo eletrônico, informática médica...)*
    - » *Interface com o usuário*

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*
    - » *Videogames*
    - » *Sistemas de controle*
    - » ...
- **Sistemas de informação**
  - Coleta, armazenamento e recuperação da informação
    - » *Segurança da informação (comércio e governo eletrônico, informática médica...)*
    - » *Interface com o usuário*
    - » *Dados georreferenciados, séries temporais*

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*
    - » *Videogames*
    - » *Sistemas de controle*
    - » ...
- **Sistemas de informação**
  - Coleta, armazenamento e recuperação da informação
    - » *Segurança da informação (comércio e governo eletrônico, informática médica...)*
    - » *Interface com o usuário*
    - » *Dados georreferenciados, séries temporais*
    - » *Ciência de dados*

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*
    - » *Videogames*
    - » *Sistemas de controle*
    - » ...
- **Sistemas de informação**
  - Coleta, armazenamento e recuperação da informação
    - » *Segurança da informação (comércio e governo eletrônico, informática médica...)*
    - » *Interface com o usuário*
    - » *Dados georreferenciados, séries temporais*
    - » *Ciência de dados*
    - » ...

Computação, “TI” e telecomunicações

- **Computação**
  - Processamento, cálculos e algoritmos
    - » *Trajetórias de foguetes*
    - » *Projeto de máquinas*
    - » *Processamento de sinais (áudio, vídeo, radar...)*
    - » *Previsão do tempo*
    - » *Videogames*
    - » *Sistemas de controle*
    - » ...
- **Sistemas de informação**
  - Coleta, armazenamento e recuperação da informação
    - » *Segurança da informação (comércio e governo eletrônico, informática médica...)*
    - » *Interface com o usuário*
    - » *Dados georreferenciados, séries temporais*
    - » *Ciência de dados*
    - » ...
- **Telecomunicações**
  - A sinergia perfeita

Popularização e o final da pré-história

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, 4–16KiB 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, 4–16KiB de memória e grande rede de varejo (Radio Shack) oferecendo upgrades, suporte e treinamento
  - Commodore PET — processador 6502, 4–8KiB de memória

O BASIC da Microsoft

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

O BASIC da Microsoft

- **Em 1977, três computadores muito influentes foram lançados:**
  - Tandy TRS-80 — processador Z80, 4–16KiB de memória e grande rede de varejo (Radio Shack) oferecendo upgrades, suporte e treinamento
  - Commodore PET — processador 6502, 4–8KiB de memória
  - apple][— processador 6502, 4–48KiB 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, 4–16KiB de memória e grande rede de varejo (Radio Shack) oferecendo upgrades, suporte e treinamento
  - Commodore PET — processador 6502, 4–8KiB de memória
  - apple][— processador 6502, 4–48KiB 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
    - » *(agenda de compromissos e contatos, jogos, lista de tarefas...)*

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
    - » *(agenda de compromissos e contatos, jogos, lista de tarefas...)*
- **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
    - » *(agenda de compromissos e contatos, jogos, lista de tarefas...)*
- **Em 2002/2003, primeiros smartphones e telefones com câmera**
  - Teclado físico + tela sensível (uso principalmente com caneta)

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
    - » *(agenda de compromissos e contatos, jogos, lista de tarefas...)*
- **Em 2002/2003, primeiros smartphones e telefones com câmera**
  - Teclado físico + tela sensível (uso principalmente com caneta)
  - Palm Treo
    - » *Basicamente, um PalmPilot + telefone*

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
    - » *(agenda de compromissos e contatos, jogos, lista de tarefas...)*
- **Em 2002/2003, primeiros smartphones e telefones com câmera**
  - Teclado físico + tela sensível (uso principalmente com caneta)
  - Palm Treo
    - » *Basicamente, um PalmPilot + telefone*
  - Blackberry
    - » *Telefone + serviço de integração com o email corporativo*

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
    - » *(agenda de compromissos e contatos, jogos, lista de tarefas...)*
- **Em 2002/2003, primeiros smartphones e telefones com câmera**
  - Teclado físico + tela sensível (uso principalmente com caneta)
  - Palm Treo
    - » *Basicamente, um PalmPilot + telefone*
  - Blackberry
    - » *Telefone + serviço de integração com o email corporativo*
    - » *Bastante sucesso no mercado empresarial*

A corrida pelo smartphone

- **Em 1996, a Palm lançou seu PDA, o PalmPilot**
  - Computador "auxiliar" portátil, usado de maneira integrada com computadores desktop
    - » *(agenda de compromissos e contatos, jogos, lista de tarefas...)*
- **Em 2002/2003, primeiros smartphones e telefones com câmera**
  - Teclado físico + tela sensível (uso principalmente com caneta)
  - Palm Treo
    - » *Basicamente, um PalmPilot + telefone*
  - Blackberry
    - » *Telefone + serviço de integração com o email corporativo*
    - » *Bastante sucesso no mercado empresarial*
- **Em 2005–2007, a Nokia lançou diversos *tablets* com o sistema Maemo, com interface usando tela sensível ao toque**
  - Fortemente baseado em software livre

A corrida pelo smartphone

- **Em 2007, a Apple lançou o iPhone**

A corrida pelo smartphone

- **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 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

A corrida pelo smartphone

- **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
  - Integração com os serviços da Apple (iTunes, iCloud...)

A corrida pelo smartphone

- **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
  - Integração com os serviços da Apple (iTunes, iCloud...)
  - Mercado “de luxo”, preço alto

A corrida pelo smartphone

- **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
  - Integração com os serviços da Apple (iTunes, iCloud...)
  - Mercado “de luxo”, preço alto
- **Em 2008, primeiro dispositivo Android**

A corrida pelo smartphone

- **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
  - Integração com os serviços da Apple (iTunes, iCloud...)
  - Mercado “de luxo”, preço alto
- **Em 2008, primeiro dispositivo Android**
  - Fortemente baseado em software livre

A corrida pelo smartphone

- **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
  - Integração com os serviços da Apple (iTunes, iCloud...)
  - Mercado “de luxo”, preço alto
- **Em 2008, primeiro dispositivo Android**
  - Fortemente baseado em software livre
  - Integração com os serviços da Google (gmail, google calendar, drive...)

A corrida pelo smartphone

- **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
  - Integração com os serviços da Apple (iTunes, iCloud...)
  - Mercado “de luxo”, preço alto
- **Em 2008, primeiro dispositivo Android**
  - Fortemente baseado em software livre
  - Integração com os serviços da Google (gmail, google calendar, drive...)
  - Mercado “comum”, preço mais baixo

A corrida pelo smartphone

- **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
  - Integração com os serviços da Apple (iTunes, iCloud...)
  - Mercado "de luxo", preço alto
- **Em 2008, primeiro dispositivo Android**
  - Fortemente baseado em software livre
  - Integração com os serviços da Google (gmail, google calendar, drive...)
  - Mercado "comum", preço mais baixo
    - » *Mas hoje há produtos Android que competem com o iPhone no mercado "de luxo"*

A corrida pelo smartphone

- **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
  - Integração com os serviços da Apple (iTunes, iCloud...)
  - Mercado "de luxo", preço alto
- **Em 2008, primeiro dispositivo Android**
  - Fortemente baseado em software livre
  - Integração com os serviços da Google (gmail, google calendar, drive...)
  - Mercado "comum", preço mais baixo
    - » *Mas hoje há produtos Android que competem com o iPhone no mercado "de luxo"*
    - » *Ampla gama de dispositivos com níveis de "qualidade" altamente variáveis*

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

Tendências

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

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"

Intermezzo

Intermezzo

Inteligência Artificial

Intermezzo

Inteligência Artificial

WTF?

Inteligência Artificial

- **O computador tem o mesmo nível de inteligência que um martelo**

Inteligência Artificial

- **O computador tem o mesmo nível de inteligência que um martelo**
  - "Inteligência artificial" não muda isso!

Inteligência Artificial

- **O computador tem o mesmo nível de inteligência que um martelo**
  - "Inteligência artificial" não muda isso!
- **IA são essencialmente técnicas usadas para resolver problemas para os quais não conhecemos algoritmos bem definidos**

Inteligência Artificial

- **O computador tem o mesmo nível de inteligência que um martelo**
  - "Inteligência artificial" não muda isso!
- **IA são essencialmente técnicas usadas para resolver problemas para os quais não conhecemos algoritmos bem definidos**
  - Quais são os passos necessários para

Inteligência Artificial

- **O computador tem o mesmo nível de inteligência que um martelo**
  - "Inteligência artificial" não muda isso!
- **IA são essencialmente técnicas usadas para resolver problemas para os quais não conhecemos algoritmos bem definidos**
  - Quais são os passos necessários para
    - » *Reconhecer o rosto de uma pessoa?*

Inteligência Artificial

- **O computador tem o mesmo nível de inteligência que um martelo**
  - "Inteligência artificial" não muda isso!
- **IA são essencialmente técnicas usadas para resolver problemas para os quais não conhecemos algoritmos bem definidos**
  - Quais são os passos necessários para
    - *Reconhecer o rosto de uma pessoa?*
    - *Decidir se um pedestre vai atravessar fora da faixa?*

Inteligência Artificial

- **O computador tem o mesmo nível de inteligência que um martelo**
  - "Inteligência artificial" não muda isso!
- **IA são essencialmente técnicas usadas para resolver problemas para os quais não conhecemos algoritmos bem definidos**
  - Quais são os passos necessários para
    - » *Reconhecer o rosto de uma pessoa?*
    - » *Decidir se um pedestre vai atravessar fora da faixa?*
    - » *Sugerir melhorias no estilo de escrita de um texto?*

Inteligência Artificial

- **O computador tem o mesmo nível de inteligência que um martelo**
  - "Inteligência artificial" não muda isso!
- **IA são essencialmente técnicas usadas para resolver problemas para os quais não conhecemos algoritmos bem definidos**
  - Quais são os passos necessários para
    - *Reconhecer o rosto de uma pessoa?*
    - *Decidir se um pedestre vai atravessar fora da faixa?*
    - *Sugerir melhorias no estilo de escrita de um texto?*
    - ...

Inteligência Artificial

- **O computador tem o mesmo nível de inteligência que um martelo**
  - "Inteligência artificial" não muda isso!
- **IA são essencialmente técnicas usadas para resolver problemas para os quais não conhecemos algoritmos bem definidos**
  - Quais são os passos necessários para
    - *Reconhecer o rosto de uma pessoa?*
    - *Decidir se um pedestre vai atravessar fora da faixa?*
    - *Sugerir melhorias no estilo de escrita de um texto?*
    - *...*
- **Dois exemplos de técnica: busca e análise estatística**

- **Alguns problemas podem ter várias soluções, algumas melhores que outras; como encontrar a "melhor"?**

- **Alguns problemas podem ter várias soluções, algumas melhores que outras; como encontrar a "melhor"?**
  - Problema do caixeiro viajante

- **Alguns problemas podem ter várias soluções, algumas melhores que outras; como encontrar a "melhor"?**
  - Problema do caixeiro viajante
  - Corte de tecido para fabricação de roupas

- **Alguns problemas podem ter várias soluções, algumas melhores que outras; como encontrar a "melhor"?**
  - Problema do caixeiro viajante
  - Corte de tecido para fabricação de roupas
  - ...

- **Alguns problemas podem ter várias soluções, algumas melhores que outras; como encontrar a "melhor"?**
  - Problema do caixeiro viajante
  - Corte de tecido para fabricação de roupas
  - ...
- **O computador é rápido!**

- **Alguns problemas podem ter várias soluções, algumas melhores que outras; como encontrar a "melhor"?**
  - Problema do caixeiro viajante
  - Corte de tecido para fabricação de roupas
  - ...
- **O computador é rápido!**
  - Que tal tentar todas as possibilidades?

- **Alguns problemas podem ter várias soluções, algumas melhores que outras; como encontrar a "melhor"?**
  - Problema do caixeiro viajante
  - Corte de tecido para fabricação de roupas
  - ...
- **O computador é rápido!**
  - Que tal tentar todas as possibilidades?
  - Em geral, não é viável

- **Alguns problemas podem ter várias soluções, algumas melhores que outras; como encontrar a "melhor"?**
  - Problema do caixeiro viajante
  - Corte de tecido para fabricação de roupas
  - ...
- **O computador é rápido!**
  - Que tal tentar todas as possibilidades?
  - Em geral, não é viável
  - Diversos algoritmos de IA existem para

Inteligência Artificial – busca

- **Alguns problemas podem ter várias soluções, algumas melhores que outras; como encontrar a "melhor"?**
  - Problema do caixeiro viajante
  - Corte de tecido para fabricação de roupas
  - ...
- **O computador é rápido!**
  - Que tal tentar todas as possibilidades?
  - Em geral, não é viável
  - Diversos algoritmos de IA existem para
    1. *Testar apenas um subconjunto relativamente pequeno de soluções*

- **Alguns problemas podem ter várias soluções, algumas melhores que outras; como encontrar a "melhor"?**
  - Problema do caixeiro viajante
  - Corte de tecido para fabricação de roupas
  - ...
- **O computador é rápido!**
  - Que tal tentar todas as possibilidades?
  - Em geral, não é viável
  - Diversos algoritmos de IA existem para
    1. *Testar apenas um subconjunto relativamente pequeno de soluções*
    2. *Escolher a melhor com base em alguma métrica (kilometragem percorrida, porcentagem de tecido perdido etc.)*

Inteligência Artificial – análise estatística

- A partir de um grande volume de dados (em geral com uma boa dose de **pré-processamento**), é possível tomar decisões probabilísticas

Inteligência Artificial – análise estatística

- **A partir de um grande volume de dados (em geral com uma boa dose de pré-processamento), é possível tomar decisões probabilísticas**
  - Dado um *corpus* de imagens, é possível determinar uma associação estatística entre aspectos dessas imagens e palavras-chave, como “gato” e “prédio”

Inteligência Artificial – análise estatística

- **A partir de um grande volume de dados (em geral com uma boa dose de pré-processamento), é possível tomar decisões probabilísticas**
  - Dado um *corpus* de imagens, é possível determinar uma associação estatística entre aspectos dessas imagens e palavras-chave, como “gato” e “prédio”
    - » *Com isso, o sistema pode processar uma nova imagem e encontrar qual é a palavra com maior probabilidade de descrevê-la*

Inteligência Artificial – análise estatística

- **A partir de um grande volume de dados (em geral com uma boa dose de pré-processamento), é possível tomar decisões probabilísticas**
  - Dado um *corpus* de imagens, é possível determinar uma associação estatística entre aspectos dessas imagens e palavras-chave, como “gato” e “prédio”
    - » *Com isso, o sistema pode processar uma nova imagem e encontrar qual é a palavra com maior probabilidade de descrevê-la*
  - Dado um *corpus* de texto, é possível determinar as probabilidades tanto para sequências de palavras específicas quanto para estruturas mais abstratas, como “sujeito-verbo-objeto”

Inteligência Artificial – análise estatística

- **A partir de um grande volume de dados (em geral com uma boa dose de pré-processamento), é possível tomar decisões probabilísticas**
  - Dado um *corpus* de imagens, é possível determinar uma associação estatística entre aspectos dessas imagens e palavras-chave, como "gato" e "prédio"
    - » *Com isso, o sistema pode processar uma nova imagem e encontrar qual é a palavra com maior probabilidade de descrevê-la*
  - Dado um *corpus* de texto, é possível determinar as probabilidades tanto para sequências de palavras específicas quanto para estruturas mais abstratas, como "sujeito-verbo-objeto"
    - » *Com isso, o sistema pode processar uma frase e calcular a probabilidade de ela estar "certa" ou "errada" (e sugerir modificações para torná-la "melhor", ou seja, mais provável)*

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

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
- **Computação quântica tem aplicações potenciais importantes**

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
- **Computação quântica tem aplicações potenciais importantes**
  - Mas elas correspondem a apenas um subconjunto pequeno dos usos do computador

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
- **Computação quântica tem aplicações potenciais importantes**
  - Mas elas correspondem a apenas um subconjunto pequeno dos usos do computador
  - Ainda é altamente experimental

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
- **Computação quântica tem aplicações potenciais importantes**
  - Mas elas correspondem a apenas um subconjunto pequeno dos usos do computador
  - Ainda é altamente experimental
- **“All science is computer science”**

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