# Arquitetura de Máquina
# Armazenamento de Dados

(Introdução à Computação)

Cleber Zanchettin - cz@cin.ufpe.br
UFPE - Universidade Federal de Pernambuco
CIn - Centro de Informática



## Tópicos

- Armazenamento de dados
  - Bits e seu armazenamento
  - Memória principal
  - Armazenamento em massa
  - Representação da informação como padrões de bits
  - O sistema binário
  - A representação de números inteiros
  - A representação de frações
  - Compressão de dados
  - Erros de comunicação
  - Representação da Imagem
  - Representação de som



## Bits e seu armazenamento

Computadores representam a informação por meio de padrões de *bits*.

***Bit*** – *dígito binário* pode assumir valores **0** e **1**.

Significado do *bit* varia de uma aplicação para outra. Algumas vezes representa valores numéricos e, em outras, caracteres ou outros símbolos.

Portas lógicas (*gates*) e *flip-flops*

George Boole, (1815-1864)

**Operações booleanas** – Operações que manipulam valores verdadeiro/falso.

**Portas Lógica** – Dispositivo que fornece a saída de uma operação booleana.



## Álgebra booleana

- Os filósofos gregos antigos criaram um sistema para formalizar argumentos chamado de lógica proposicional
- Uma proposição é uma declaração que pode ser **TRUE** ou **FALSE**
- Proposições podem ser compostas através dos operadores AND, OR e NOT
  - *AND = "e"*
  - *OR = "ou"*
  - *NOT = "não"*

## Álgebra booleana

- Proposições podem ser verdadeiras ou falsas (TRUE or FALSE):
  - Está chovendo
  - A previsão do tempo indica chuva
- Proposições combinadas:
  - Está chovendo **OR** a previsão do tempo indica chuva
- Resultados de proposições:
  - Eu levarei guarda-chuva = Está chovendo **OR** a previsão do tempo indica chuva
- Se estiver chovendo **OR** a previsão do tempo indicar chuva então eu levarei o guarda-chuva



## Bits e seu armazenamento

A operação **OR**

| | | |
|---|---|---|
| 0 | 0 | 1 | 1 |
| **OR** 0 | **OR** 1 | **OR** 0 | **OR** 1 |
| 0 | 1 | 1 | 1 |

A operação **AND**

| | | |
|---|---|---|
| 0 | 0 | 1 | 1 |
| **AND** 0 | **AND** 1 | **AND** 0 | **AND** 1 |
| 0 | 0 | 0 | 1 |



## Operações booleanas

A operação **XOR**

| | | |
|---|---|---|
| 0 | 0 | 1 | 1 |
| **XOR** 0 | **XOR** 1 | **XOR** 0 | **XOR** 1 |
| 0 | 1 | 1 | 0 |

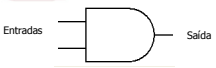
A operação **NOT**

| | |
|---|---|
| **NOT** 0 | **NOT** 1 |
| 1 | 0 |



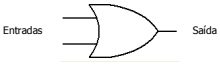
## Portas lógicas (*gates*)

AND




| Entradas | | Saída |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

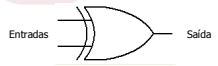
OR




| Entradas | | Saída |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

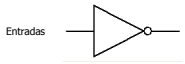
## Portas lógicas (*gates*)

XOR




| Entradas | | Saída |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOT



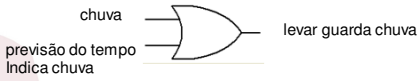

| Entradas | Saída |
|---|---|
| 0 | 1 |
| 1 | 0 |

- Portas lógicas podem ser implementadas por meio de diversas tecnologias, tais como engrenagens, relés e dispositivos óticos.
- Nos computadores modernos são constituídas de pequenos circuitos eletrônicos, 0 e 1 são representados por níveis de tensão.



## Portas lógicas - exemplo

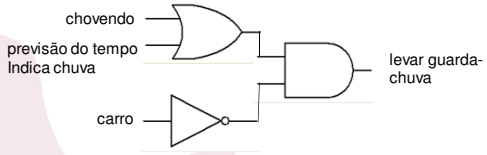
- Pode-se pensar que a proposição *guarda-chuva* como um resultado que deve ser calculado pelo combinação dos resultados das proposições *chovendo* e *previsão do tempo:*






## Portas lógicas - exemplo

- Proposições podem ser mais complexas, por exemplo:
  - (levar guarda-chuva ) = ( NOT (*ir de carro* ) ) AND ( (*previsão de chuva* ) OR (*chovendo* ) )

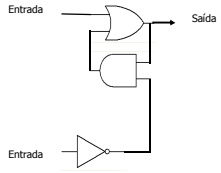




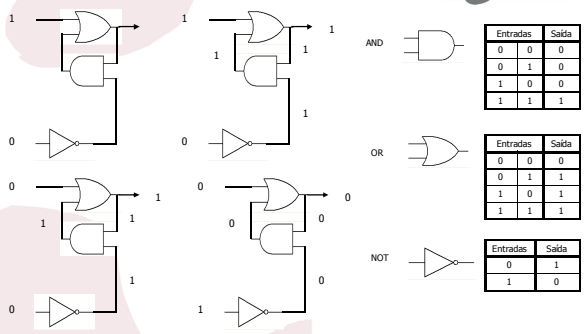

## *Flip-flops*

Blocos de portas lógicas

- Flip-flop é um circuito cuja saída apresenta um dos dois valores binários, permanecendo assim até que um pulso temporário em sua entrada, proveniente de outro circuito, venha a forçá-lo a modificar sua saída.
- Importância é que ele se mostra ideal para o armazenamento de um *bit* no interior de um computador.
- Pode ser usado como módulos para construir circuitos mais complexos.
- O projeto dos circuitos de um computador se faz em uma estrutura hierárquica, onde cada nível usa os componentes do nível abaixo como ferramentas abstratas.



## Flip-flops



AND

| Entradas | | Saída |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR

| Entradas | | Saída |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

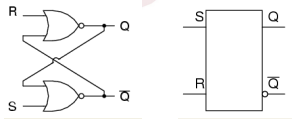
NOT

| Entradas | Saída |
|---|---|
| 0 | 1 |
| 1 | 0 |

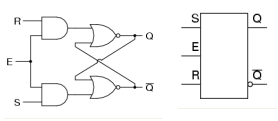


## Flip-flops

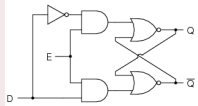
R-S flip-flop



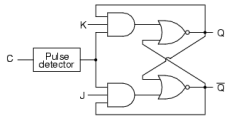
R-S enable flip-flop



D flip-flop



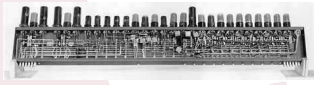
J-K flip-flop






## Outras técnicas de armazenamento

- Na década de 1960, computadores utilizavam núcleo magnético como uma forma de armazenamento de bits.
- Método mais recente para armazenar um *bit* é o capacitor. A tecnologia atual é capaz de construir milhões de pequenos capacitores, bem como seu circuitos, tudo em um única pastilha (**CHIP**)
- Flip-flops, núcleos e capacitores são exemplos de sistemas de armazenamento com diferentes graus de volatilidade.
- A memória do computador construída com essa tecnologia, freqüêntemente é chamada de memória dinâmica.





## A notação hexadecimal

- Quando consideramos as atividades internas de um computador, devemos lidar com cadeia de *bits*. (**stream**)
- Notação hexadecimal é mais compacta, facilitando assim a memorização da sequência.
- Exemplos:

  1011 0101 – B5

  1010 0100 1100 1000 – A4C8

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

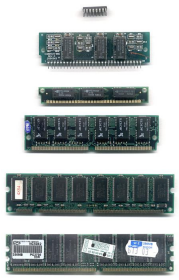
## Memória principal

- Circuitos de armazenamento da memória principal são organizados em unidades manipuláveis denominadas células (ou *palavras*), geralmente de oito bits (**byte**).
- Representação

Bit de ordem mais elevada: 0 1 0 1 1 0 1 0 — Bit de ordem mais baixa

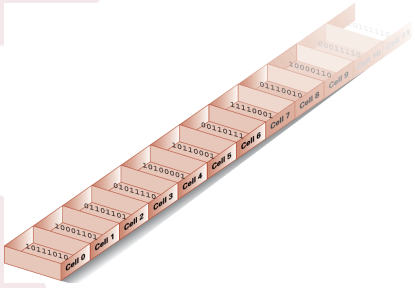
Bit mais significativo — Bit menos significativo

- Para distinguir cada célula individual, esta é identificada por um nome único, denominado endereço.
- Se a memória estiver dividida em células com tamanho de um byte cada, podemos representar uma cadeia de 16 bits utilizando duas células consecutivas de memória.





## Memória principal






## Memória principal

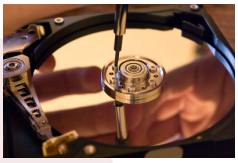
- Outra conseqüência de organizar a memória de uma máquina com pequenas células endereçadas é que cada uma pode ser individualmente referenciada. Dados armazenados na memória principal de uma máquina podem ser processados em qualquer ordem, razão pela qual este tipo de memória é conhecido como memória de acesso aleatório (*random access memory* – RAM).
- Outro tipo de memória:
  *Dynamic random access memory* (DRAM) – armazena cada bit de dados em um capacitor separado. Mais simples, um transistor e um capacitor ou invés de seis transistores.

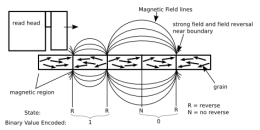
  Variações: Video DRAM (VRAM), Fast Page Mode DRAM (FPM), Window RAM (WRAM), Extended Data Out (EDO) DRAM, Burst EDO (BEDO) DRAM, Multibank DRAM (MDRAM), Synchronous Graphics RAM (SGRAM), Synchronous Dynamic RAM (SDRAM), Direct Rambus DRAM (DRDRAM), Double Data Rate (DDR) SDRAM, Pseudostatic RAM (PSRAM), 1T DRAM, Quad Data Rate (QDR) SDRAM

| | |
|---|---|
| E0 | |
| E1 | |
| E2 | |
| | |
| E518 | 46 |
| | |
| E833 | 17 |
| | |
| En | |

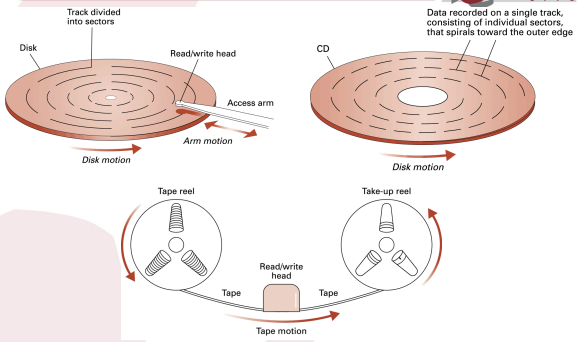


## Armazenamento em massa





## Armazenamento em massa






## Armazenamento em massa




## Representação da informação como padrões de *bits*

- Como a informação pode ser codificada com padrões de *bits*?

Representação de texto

- ANSI – American National Standard Institute.
  - ASCII – American Standard Code for Information Interchange.
    No início o padrão era de 7 *bits*. Atualmente o formato utiliza 8 *bits* (byte).

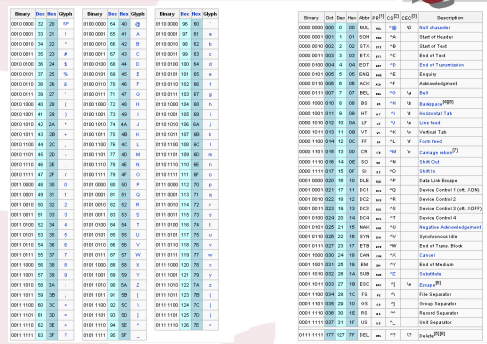
```
01001000  01100101  01101100  01101100  01101111  00101110
   H         e         l         l         o         .
```

- Unicode
  Padrão com 16 bits - 65.536 diferentes padrões, suficiente para representar os símbolos mais comuns dos idiomas chinês e japonês.

- ISO – International Organization for Standardization
  Utiliza padrões de 32 bits, capacidade de representar bilhões de símbolos.



## Representação da informação como padrões de *bits* – Código ASCII



## Representação da informação como padrões de *bits*

### Representação de valores numéricos

- O método de armazenar informação na forma de caracteres codificados é ineficiente quando a informação a ser registrada é puramente numérica.

  Exemplo:

  O número 25 seria armazenado em 16 *bits* (ASCII utiliza um byte por símbolo). Além disso, 99 é o maior número que podemos armazenar desta forma.

- Modo mais eficiente, representação em base dois, ou seja, notação binária.

a. O sistema de base dez

| 3 | 7 | 5 | - Representação |
|---|---|---|---|
| Cem | Dez | Um | - Quantidade da posição |

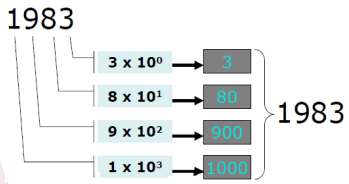
b. O sistema de base dois

| 1 | 0 | 1 | 1 | - Representação |
|---|---|---|---|---|
| Oito | Quatro | Dois | Um | - Quantidade da posição |
| $2^3$ | $2^2$ | $2^1$ | $2^0$ | |



## Sistemas de numeração

- Representação de valores
  - Números decimais
    - Dígitos arábicos: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9

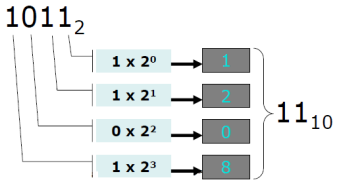



exemplo extraído da aula do prof. Jander Moreira DC/UFSCar



## Sistemas de numeração

- Representação de valores
  - Números binários
    - Dígitos arábicos: 0, 1

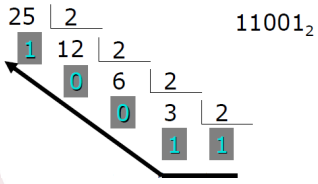



exemplo extraído da aula do prof. Jander Moreira DC/UFSCar



## Sistemas de numeração

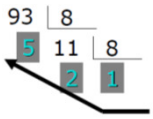
- Conversão de base: 10 para 2
  - Divisões sucessivas pela base
  - Controle do resto




exemplo extraído da aula do prof. Jander Moreira DC/UFSCar

## Sistemas de numeração

- Conversão de base: 10 para 8
  - Divisões sucessivas pela base
  - Controle do resto




$125_8$

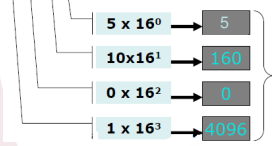
exemplo extraído da aula do prof. Jander Moreira DC/UFSCar



## Sistemas de numeração

- Representação de valores
  - Números hexadecimais
    - Dígitos arábicos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

$10A5_{16}$




$4261_{10}$

exemplo extraído da aula do prof. Jander Moreira DC/UFSCar



## Exercício

- Converter para binário

10 =
25 =
43 =
127 =

- Converter para decimal

$11011_2$ =
$100000_2$ =
$1000001_2$ =



## Sistema binário

### Adição binária

As tabuadas de adição binária

| 0 | 1 | 0 | 1 |
|---|---|---|---|
| + 0 | + 0 | + 1 | + 1 |
| 0 | 1 | 1 | 10 |

```
  00111010
+ 00011011
```

```
    1
  00111010
+ 00011011
        01
```

```
    1
  00111010
+ 00011011
      0101
```

```
    1
  00111010
+ 00011011
    010101
```

```
  00111010
+ 00011011
  01010101
```

## Sistema binário

### Frações de números binários

- Utilizamos uma vírgula (ponto – radix point), que funciona do mesmo modo que a vírgula da notação decimal.
- Os dígitos à direita da vírgula representam a parte fracionária do valor, sendo interpretado de modo semelhante aos outros bits, exceto em que às posições são associadas quantidades fracionárias: à primeira posição à direita da vírgula, é associada a quantidade ½; à posição seguinte, ¼ ; e assim por diante.

| 1 | 0 | 1 . | 1 | 0 | 1 |
|---|---|---|---|---|---|
| $2^2$ | $2^1$ | $2^0$ . | $1/2^1$ | $1/2^2$ | $1/2^3$ |
| 4 | 0 | 1 . | 1/2 | 0 | 1/8 = 5(5/8) |

10,011
+ 100,110
111,001



## A representação de números inteiros

### Notação de complemento de dois

- Este sistema emprega um número fixo de bits para representar cada valor numérico. O bit mais à esquerda do padrão indica o sinal do valor representado chamado de bit de sinal.

| Padrão de bits | Valor representado |
|---|---|
| 011 | 3 |
| 010 | 2 |
| 001 | 1 |
| 000 | 0 |
| 111 | -1 |
| 110 | -2 |
| 101 | -3 |
| 100 | -4 |

Padrões de comprimento três

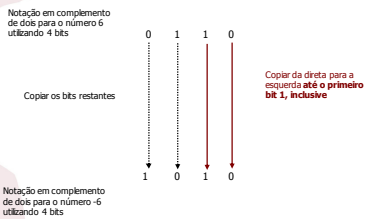
| Padrão de bits | Valor representado |
|---|---|
| 0111 | 7 |
| 0110 | 6 |
| 0101 | 5 |
| 0100 | 4 |
| 0011 | 3 |
| 0010 | 2 |
| 0001 | 1 |
| 0000 | 0 |
| 1111 | -1 |
| 1110 | -2 |
| 1101 | -3 |
| 1100 | -4 |
| 1011 | -5 |
| 1010 | -6 |
| 1001 | -7 |
| 1000 | -8 |

Padrões de comprimento quatro



## A representação de números inteiros

### Notação de complemento de dois






## A representação de números inteiros

### Adição na notação de complemento de dois

| Problema na base de dez | Problema em complemento de dois | Resposta na base de dez |
|---|---|---|
| 3 + 2 | 0011 + 0010 = 0101 | 5 |
| -3 + -2 | 1101 + 1110 = 1011 | -5 |
| 7 + -5 | 0111 + 1011 = 0010 | 2 |

A vantagem da notação de complemento de dois é que a adição de qualquer combinação de números, positivos e negativos, podem ser efetuadas usando o mesmo algoritmo, portanto o mesmo circuito.

7 + -5 = 0111 + 1011 = 0010 = 2

7 - 0101

## A representação de números inteiros

### O problema do estouro

- Em qualquer sistema de complemento de dois, existe sempre um limite para o tamanho dos números a serem representados.
  Exemplo: quando usamos complemento de dois, com padrões de quatro bits, ao valor 9 não esta associado padrão algum, por isso, não conseguimos obter uma resposta certa para a soma 5+4, o resultado apareceria como –2.
- Usando 32 bits para armazenar valores em complemento de dois teremos 2.147.483.647 valores antes que ocorra estouro.
- 16 bits –> $2^{15}$=32.768



## A representação de números inteiros

### A notação de excesso

| Padrão de bits | Valor representado |
|---|---|
| 1111 | 7 |
| 1110 | 6 |
| 1101 | 5 |
| 1100 | 4 |
| 1011 | 3 |
| 1010 | 2 |
| 1001 | 1 |
| 1000 | 0 |
| 0111 | -1 |
| 0110 | -2 |
| 0101 | -3 |
| 0100 | -4 |
| 0011 | -5 |
| 0010 | -6 |
| 0001 | -7 |
| 0000 | -8 |

Tabela de conversão para a notação de excesso de oito.

| Padrão de bits | Valor representado |
|---|---|
| 111 | 3 |
| 110 | 2 |
| 101 | 1 |
| 100 | 0 |
| 011 | -1 |
| 010 | -2 |
| 001 | -3 |
| 000 | -4 |

Sistema de notação de excesso, com padrões de três bits de comprimento.



## A representação de frações

### A notação de vírgula flutuante

- posição de bits
Mantissa
Expoente
Bits de sinal

01101011
o bit de sinal é 0,
o expoente é 110
a mantissa é 1011

Convertendo:
1 0, 1 1
$2^1$ $2^2$,1/$2^1$ 1/$2^2$
---
2 0, 1/2 1/4
2, 3/4

Extraímos a mantissa e colocamos a vírgula binária à sua esqueda
,1011
Extraímos o conteúdo do campo do expoente (110) => 2 em representação de excesso
Desta forma, deslocamos a virgula binária dois dígitos para a direita
10,11
= 2(3/4)
O bit de sinal é 0, assim o valor representado é não-negativo.
01101011 = 2(3/4)



## A representação de frações

### A notação de vírgula flutuante

- posição de bits
Mantissa
Expoente
Bits de sinal

1(1/8)
Expressamos na notação binária
1,001
Copiamos o padrão de bits no campo da mantissa, da esquerda para a direita , iniciando com o primeiro bit não-nulo
_ _ _ _ 1001
O expoente deve 1 positivo para o deslocamento da vírgula a direita
- 101 1001
O valor do bit de sinal é positivo
0 101 1001

## Compressão de dados

- Com a finalidade de armazenar e transferir dados, freqüêntemente é útil reduzir o seu tamanho.

Técnicas genéricas de compressão

- Codificação de tamanho de seqüência
  Substituir seqüências pro um código indicativo do valor repetido e do número de vezes que ele ocorre.
- Codificação relativa
  Registrar as diferenças entre blocos consecutivos em vez dos blocos inteiros, isto é cada bloco é codificado em termos de sua relação com o bloco precedente.
- Codificação dependente da freqüência.
- Códigos de comprimento variável.
- Códigos de Huffman (depende da freqüência)
- Codificação de Lempel-Ziv
- Codificação adaptável de dicionário



## Compressão de dados

Código Lempel-Ziv LZ77

xyxxyzy(5,4,x)

xy.xxyzy
[conte retroativamente 5 dígitos]

xy[xxyz]y
[identifique o segmento a ser adicionado]

xy.xxyz.y[xxyz]
[copie o segmento]

xy.xxyz.y.[xxyz].x
[acrescente o símbolo final]



## Compressão de dados

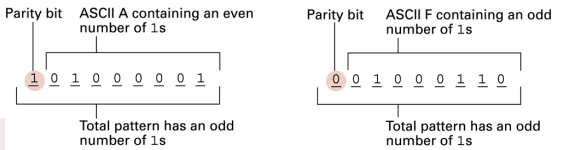
Exercício

xyxxyzy(5,4,x)(0,0,w)(8,6,y)



## Erros de comunicação

Bits de paridade



## Códigos de Correção de Erros

Distância de Hamming

| Symbol | Code |
|---|---|
| A | 000000 |
| B | 001111 |
| C | 010011 |
| D | 011100 |
| E | 100110 |
| F | 101001 |
| G | 110101 |
| H | 111010 |

Decodificação do padrão 010100

| Character | Distance between the received pattern and the character being considered |
|---|---|
| A | 2 |
| B | 4 |
| C | 3 |
| D | 1 Smallest distance |
| E | 3 |
| F | 5 |
| G | 2 |
| H | 4 |

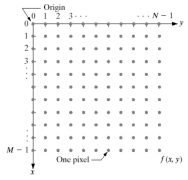
E o padrão 001111 100100 001100?



## Representação da Imagem



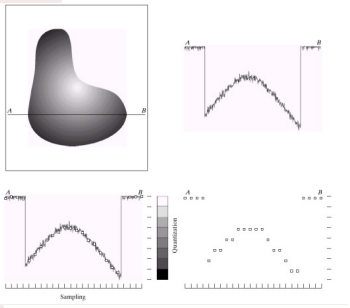

## Amostragem e quantificação
## Representação da Imagem



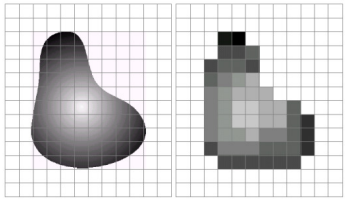

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$



## Amostragem e quantificação



## Amostragem e quantificação

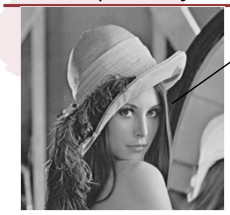




## Amostragem e quantificação
## Representação da Imagem



## Amostragem e quantificação
## Representação da Imagem

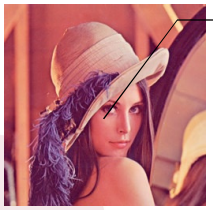


| | | | | |
|---|---|---|---|---|
| 142 | 152 | 152 | 132 | |
| 151 | 212 | 154 | 232 | |
| 121 | 254 | 132 | 215 | |
| 252 | 224 | 121 | 151 | |
| 254 | 181 | 145 | 212 | |

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |



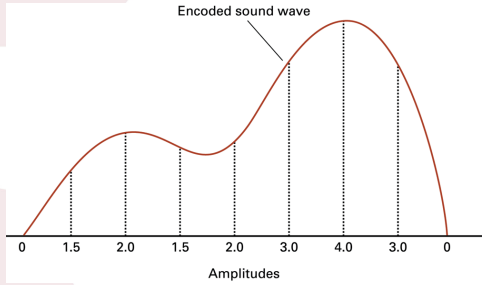
## Amostragem e quantificação
## Representação da Imagem



| | | | | |
|---|---|---|---|---|
| 14 | 15 | 15 | 13 | 15 |
| 15 | 21 | 54 | 32 | 15 |
| 21 | 54 | 32 | 15 | 54 |
| 52 | 24 | 21 | 51 | 51 |
| 54 | 81 | 45 | 12 | 15 |

| C | M | Y |
|---|---|---|
| R | G | B |
| 0.3212 | 0.2541 | 0.4121 |
| 0.1312 | 0.1215 | 0.3121 |
| 0.2115 | 0.1511 | 0.0212 |
| 0.0215 | 0.5152 | 0.3211 |
| 0.3551 | 0.1212 | 0.3251 |

## Representação de som






## Crédito

Site do Prof. **Tsang Ing Ren** / CIN - UFPE:
http://www.cin.ufpe.br/~tir/

Site do Prof. **George Cavalcanti** / CIN - UFPE:
http://www.cin.ufpe.br/~gdcc/