# ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

PROF ME MARCO IKURO HISATOMI

# Livro didático


**Fonte:** Tangon, Leonardo Guimarães, 2016

# Conteúdo Programático

## Unidade 4 | Álgebra Booleana e Lógica Digital

# Conteúdo Programático

- Unidade 3 - Sistemas numéricos: conceitos, simbologia e representação de base numérica
- Sistemas numéricos: conceitos, simbologia e representação de base numérica
- **Conversão entre bases numéricas: decimal**
- Conversão entre bases numéricas: binário
- Conversão entre bases numéricas: octal

# Situação Geradora de Aprendizagem

NÚMEROS BINÁRIOS TRABALHADOS PELO PROCESSADOR

# Contextualizando

- Considere que você faz parte de um time de desenvolvimento de hardware de uma empresa LOGIC que necessita de construir um circuito de lógica
- Após adquirir o conhecimento desta aula, você vai compreender como funciona o **sistema de esteira de um caixa de supermercado: rolando até que um produto fique em frente ao sensor**.

# Contextualizando

- Aprenderemos operações lógicas básicas conhecidas como NOT, AND e OR.
- Criar as tabelas verdade para cada caso. Essas tabelas demonstram logicamente o que está entrando de informação e qual será o resultado de saída, sempre verdadeiro ou falso.

# INTRODUÇÃO À ÁLGEBRA BOOLEANA

# Contextualizando

- No estado "fechado" o interruptor permitirá que a corrente elétrica passe através do ponto, permitindo, assim, que uma lâmpada seja acesa, por exemplo. Já no estado "aberto" não se permite a passagem da corrente elétrica pelo ponto, fazendo com que a lâmpada fique desligada.
- Você deverá usar para o estado "fechado" o valor 1 e para o estado "aberto" o valor 0.

- **George Boole** é considerado um dos fundadores da ciência da computação, mesmo não existindo computadores em seus dias (1815 – 1864)

# Contextualizando

- (SHANNON,1938) sugeriu que a já conhecida álgebra booleana poderia ser usada para resolver problemas relativos a projeto de circuitos de comutação de relés.
- **Relés** – Um dispositivo eletromecânico, formado por um magneto móvel, que se deslocava unindo dois contatos metálicos, como uma chave.
- **Circuitos de Comutação** – É um processo onde se pode interligar dois ou mais processos entre si, neste caso, utilizando-se dos relés, que nada mais é que uma chave.

# Símbolo

- É Como em qualquer uso de álgebra convencional, a álgebra booleana utiliza-se de variáveis e operações lógicas. Essa variável pode ter o valor lógico 1 (verdadeiro) ou 0 (falso).
- Cada variável pode assumir um único valor, sendo ele: 1 ou 0, verdadeiro ou falso, true ou false, sim ou não, aberto ou fechado, aceso ou apagado, entre outros.
- As operações lógicas básicas são: AND (E), OR (OU) e NOT (NÃO)

Tabela 4.1 – Simbologia de Operações Lógicas.

| Operações Lógicas Básicas | AND (E) | OR (OU) | NOT (NÃO) |
|---|---|---|---|
| Simbologia utilizada na matemática | • | + | ′ |
| Simbologia utilizada em computação | ∧ | ∨ | ! OU ¯ |

# Exemplo

A = Ana viaja

B = Ana brinca

- $A \land B \rightarrow$ Ana viaja **e** Ana brinca
- $A \lor B \rightarrow$ Ana Viaja **ou** Ana Brinca
- $!A \rightarrow$ Ana **não** viaja

A partir das 3 operações lógicas temos:

- AND → Produto Lógico
- OR → Soma Lógica
- Not → Negação

## Tabela verdade **AND**

Nessa tabela usamos a proposições **p** ∧ **q**. Para que **p** ∧ **q** sejam verdadeiras, as duas proposições tem de ser verdadeiras (STALLINGS, 2003).

Tabela 4.2 – Tabela-verdade AND.

| p | Q | p ∧ q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Tabela verdade

Se usarmos fechado e aberto para **p** ∧ **q**.

| p | q | p ∧ q |
|---|---|---|
| Aberto | Aberto | Aberto |
| Aberto | Fechado | Aberto |
| Fechado | Aberto | Aberto |
| Fechado | Fechado | Fechado |

# Exemplo

- $x = 3$
- $y = 5$
- A expressão $(x = 4) \wedge (y=5)$ é verdadeira (1) ou falsa (0)?

| $(x = 4)$ | $(y=5)$ | $(x = 4) \wedge (y=5)$ |
|---|---|---|
| Falsa | Verdadeira | Falsa |

- Temos: (falsa) $\wedge$ (verdadeira)
- Logo o valor para $(x = 4) \wedge (y=5)$ = **falsa**

# Tabela verdade **OR**

Nessa tabela usamos a proposições **p** V **q**.

Para que **p** V **q** seja verdadeira basta que uma delas seja verdadeira para (STALLINGS, 2003).

Tabela 4.3 – Tabela-verdade OR.

| p | q | p ∨ q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# Exemplo

- k = azul
- w = verde
- A expressão (k = vermelho) v (w=verde) é verdadeira (1) ou falsa (0)?

| (k = vermelho) | (w=verde) | (k = vermelho) v (w=verde) |
|---|---|---|
| Falsa | Verdadeira | |

- Temos: (falsa) V (verdadeira)
- Logo o valor para **p** V **q** =

# Exemplo

- k = azul
- w = verde
- A expressão (k = vermelho) v (w=verde) é verdadeira (1) ou falsa (0)?

| (k = vermelho) | (w=verde) | (k = vermelho) ∧ (w=verde) |
|---|---|---|
| Falsa | Verdadeira | Verdadeira |

- Temos: (falsa) V (verdadeira)
- Logo o valor para **p** V **q** = **verdadeira**

# Tabela verdade **NOT**

Nessa tabela usamos a proposições **p** e **!p**.

Como podemos ver em:

Tabela 4.4 – Tabela-verdade NOT.

| p | !p |
|---|---|
| 0 | 1 |
| 1 | 0 |

# Exemplo

- **p** = 65

Qual é o valor de **!p**?

- Temos **!p** =

# Tabela verdade **NAND**

Equivale a uma porta lógica AND seguida de NOT → (p ∧ q)!

Tabela 4.5 – Tabela-verdade NAND.

| **p** | **q** | **p ∧ q!** |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Tabela verdade **NAND**

Se usarmos fechado e aberto para **(p ∧ q)!**

| p | q | (p ^ q)! |
|---|---|---|
| Aberto | Aberto | |
| Aberto | Fechado | |
| Fechado | Aberto | |
| Fechado | Fechado | |

# Tabela verdade **NAND**

Se usarmos fechado e aberto para **(p ∧ q)!**

| p | q | (p ∧ q)! |
|---|---|---|
| Aberto | Aberto | Fechado |
| Aberto | Fechado | Fechado |
| Fechado | Aberto | Fechado |
| Fechado | Fechado | Aberto |

# Tabela verdade **NOR**

Equivale a uma porta lógica OR seguida de NOT → (p v q)!

Tabela 4.6 – Tabela-verdade NOR.

| p | q | p ∨ q! |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

# Tabela verdade **NOR**

Se usarmos fechado e aberto para (**p** v **q**)!

| p | q | (p v q)! |
|---|---|---|
| Aberto | Aberto | Fechado |
| Aberto | Fechado | Aberto |
| Fechado | Aberto | Aberto |
| Fechado | Fechado | Aberto |

# Tabela verdade **XOR**

Conhecida como OU EXCLUSIVO (⊕). Ela compara dois valores e, se o resultado for diferente, mostra como saída o valor 1 → p ⊕ q.

Somente terá resultado como verdadeiro quando os valores de p e q tiveram resultado diferente entre eles.

Tabela 4.7 – Tabela-verdade XOR.

| **p** | **Q** | **p ⊕ q** |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Tabela verdade **XNOR**

É a negação de XOR → (p ⊕ q)!.

Somente terá resultado como verdadeiro quando os valores de p e q tiverem resultado iguais entre eles.

Tabela 4.8 – Tabela-verdade XNOR.

| p | Q | p ⊕ q! |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

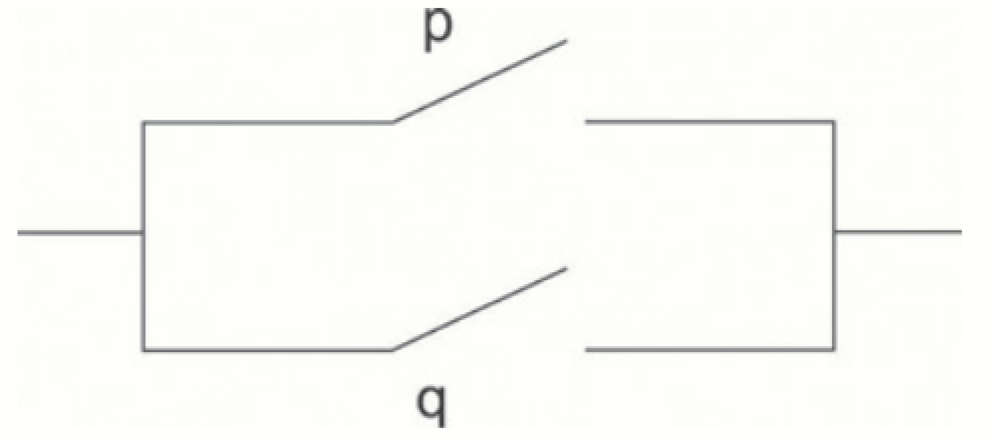
# CIRCUITO

# Desafio 1 circuito

Agora que você já conhece os conceitos e passos para a elaboração de uma PCI simples, para criar a PCI é necessário “confeccionar” seu diagrama do circuito de um interruptor.

A Expressão que representa a imagem é:

**AND** ou **OR ?**



# Desenvolvendo circuito

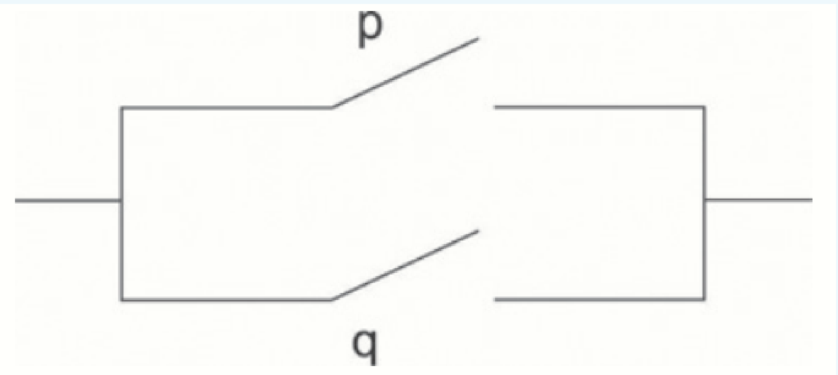



Tabela 4.9 – Tabela-verdade OR para estado do interruptor.

| Estado de p | Estado de q | Estado de p ∨ q | Estado do interruptor |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 1 | |
| 1 | 1 | 1 | |

# Desenvolvendo circuito

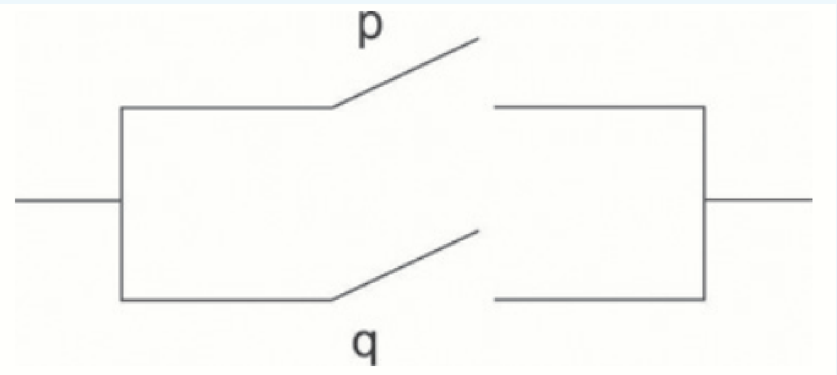



Tabela 4.9 – Tabela-verdade OR para estado do interruptor.

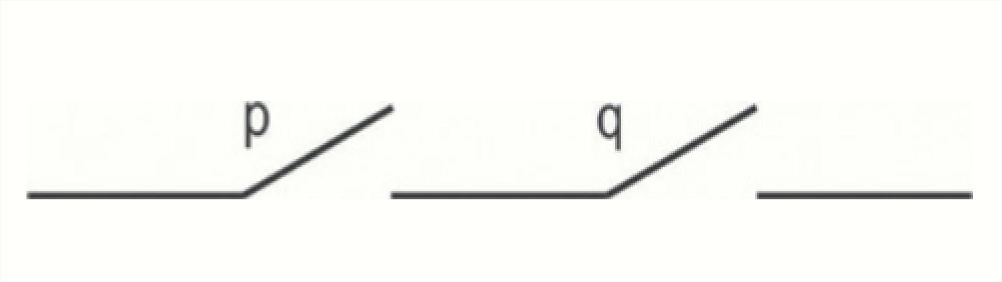
| Estado de p | Estado de q | Estado de p ∨ q | ➡ Estado do interruptor |
|---|---|---|---|
| 0 | 0 | 0 | Desligado |
| 0 | 1 | 1 | Ligado |
| 1 | 0 | 1 | Ligado |
| 1 | 1 | 1 | Ligado |

# Desafio 2 circuito

Agora que você já conhece os conceitos e passos para a elaboração de uma PCI simples, para criar a PCI é necessário "confeccionar" seu diagrama do circuito de um interruptor.

A Expressão que representa a imagem é:

**AND** ou **OR ?**



# Desenvolvendo circuito

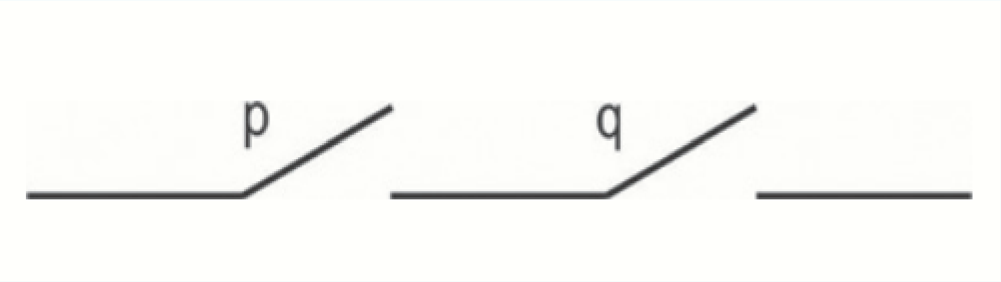



Tabela 4.10 – Tabela-verdade AND para estado do interruptor

| Estado de p | Estado de q | Estado de $p \wedge q$ | Estado do interruptor |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 1 | 0 | |
| 1 | 0 | 0 | |
| 1 | 1 | 1 | |

# Desenvolvendo circuito


Tabela 4.10 – Tabela-verdade AND para estado do interruptor

| Estado de p | Estado de q | Estado de p ∧ q | → Estado do interruptor |
|---|---|---|---|
| 0 | 0 | 0 | Desligado |
| 0 | 1 | 0 | Desligado |
| 1 | 0 | 0 | Desligado |
| 1 | 1 | 1 | Ligado |

# CONVERSÃO ENTRE BASES NUMÉRICAS: DECIMAL

# Fundamento para conversão

# Conversão de decimal para binário

Para isso temos de seguir alguns passos:

a. Fazer a divisão sucessiva por 2 até que o quociente chegue em 0.

b. O quociente só recebe o número inteiro caso o resultado der uma fração.

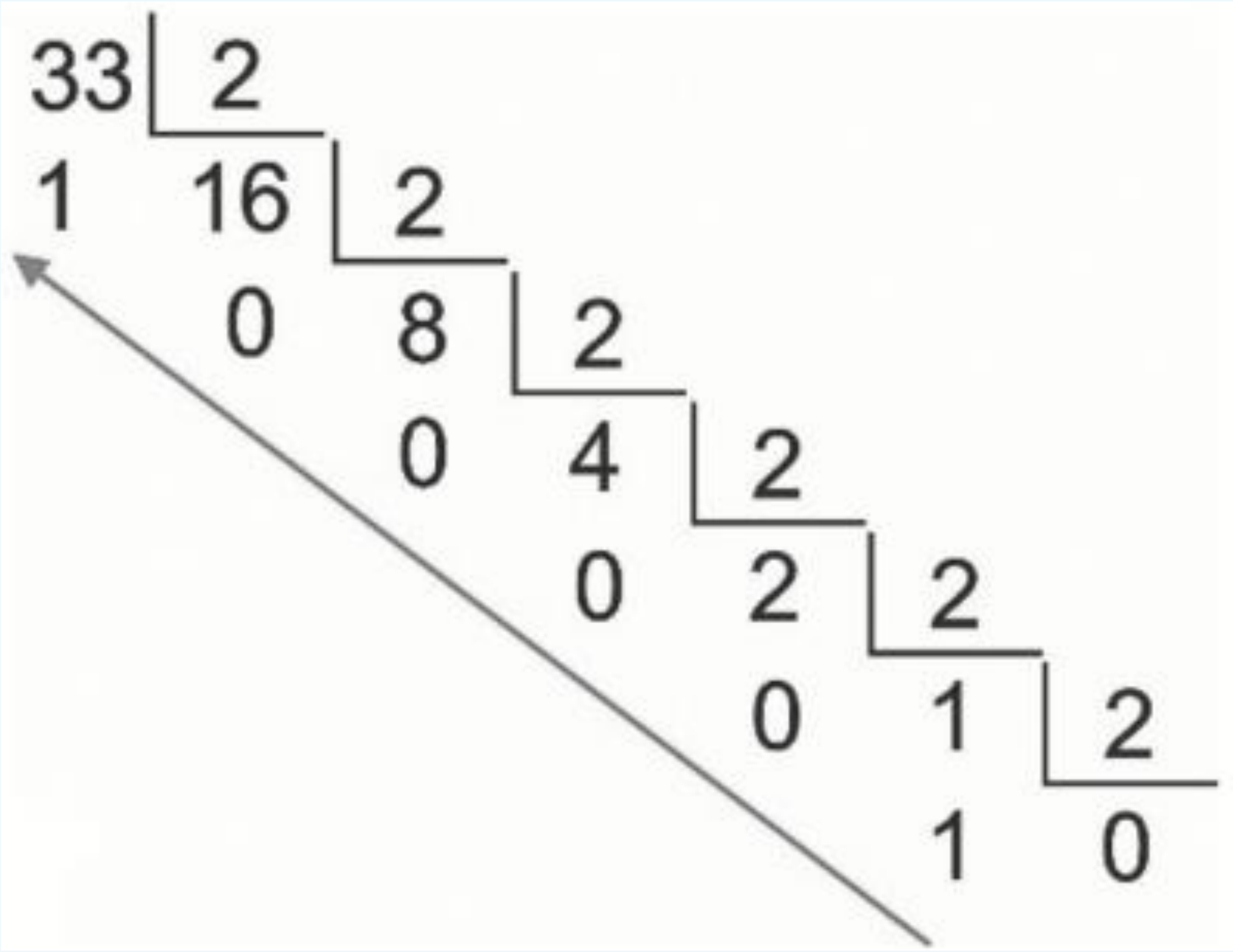
Exemplo: Se o quociente for 4,5, utiliza-se somente o 4.

c. Os restos sempre serão 0 ou 1 (uma dica é verificar se o dividendo é par ou ímpar. Se for par, retornará sempre 0 e, se for ímpar, retornará sempre 1).

d. Quando o quociente chegar a 0, pegam-se os restos de baixo para cima, da direita para a esquerda.

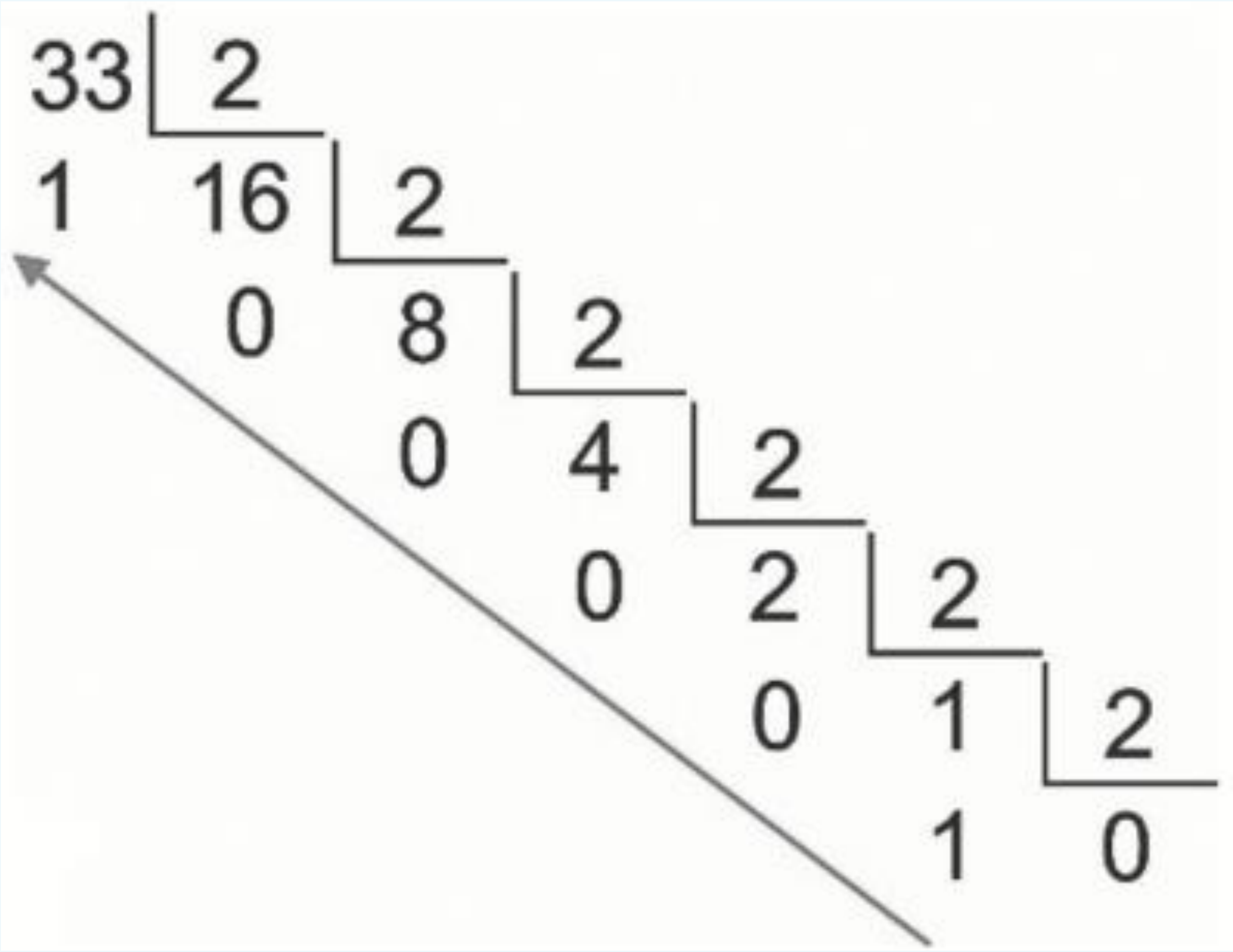
# Conversão de decimal para binário: exemplo

$33_{10}$ - ______ $_2$



# Conversão de decimal para binário: exemplo

$33_{10}$ - **100001** $_2$



# Vídeo: Decimal em Binário

<https://www.youtube.com/watch?v=mttrG_kbHN4>

# Vídeo: Binário em Decimal

<https://www.youtube.com/watch?v=zToihF2FE9I>

# CONVERSÃO ENTRE BASES NUMÉRICAS: DECIMAL

# Exercícios

$10_{10}$ - ______ 2

$10_{10}$ - ______16

$256_{10}$ - __________ 2

# Exercícios

$1487_{10}$ - ______ 2

$FAC3_{16}$ - ______10

$CADE_{16}$ - __________ 10

# RECAPITULANDO

- Conversão de bases numéricas.