Estratégia Concursos

Aula 00 (Thais Martins) - Somente em PDF

CODEVASF (Analista de Desenvolvimento Regional - Engenharia Elétrica) Conhecimentos Específicos

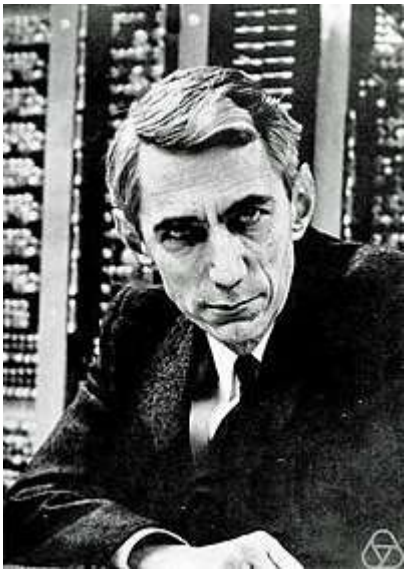
Autor:
Edimar Natali Monteiro, Mariana Moronari, Thais Martins

19 de Fevereiro de 2024

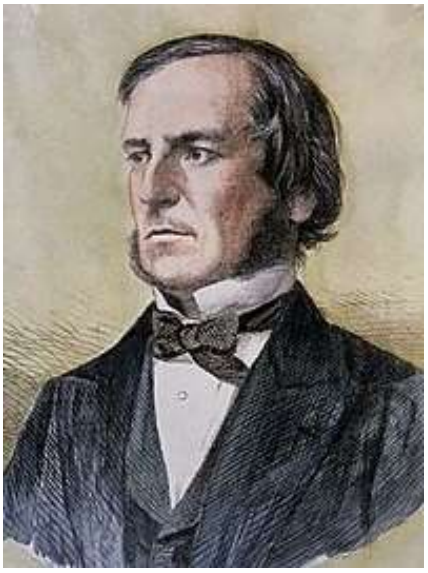
Índice

CONSIDERAÇÕES INICIAIS

Iniciaremos agora nossos estudos sobre **a Eletrônica Digital**! Este nicho da Eletrônica teve início em 1938, quando o matemático e engenheiro norte-americano Claude Shannon (1916-2001) organizou o trabalho do também matemático e filósofo britânico **George Boole** (1815-1864), considerado o **pai da álgebra booleana**, como sendo uma convenção para o emprego de conceitos de eletrônica.



Fonte: https://pt.wikipedia.org/wiki/Claude_Shannon



Fonte: https://pt.wikipedia.org/wiki/George_Boole

E desde o final da década de 1930 vem se aprimorando de maneira arrebatadora. Iniciaremos com alguns **conceitos introdutórios**, tais como **Bases Numéricas**, **Operações de Adição e Subtração com Números Binários**, **Portas Lógicas**, passando a seguir para os **Circuitos Digitais** propriamente ditos.

CONCEITOS INTRODUTÓRIOS

Bases Numéricas

Trabalharemos com **4 sistemas numéricos**: **binário**, **octal**, **decimal** e **hexadecimal**. Os sistemas **binário**, **octal** e **hexadecimal** guardam uma **correlação** entre si, visto que as bases destes sistemas possuem em comum o fato de **serem todas potências de 2**. E, por guardarem uma correlação tão íntima, as conversões entre estas bases são *relativamente* fáceis.

O **sistema numérico binário** (*ou base numérica binária*) pode representar qualquer número com apenas **dois algarismos**: 0 e 1. No sistema binário, o termo dígito binário (*binary digit*) é abreviado para o termo **bit** e representa um único "0" ou um único "1" dentro de um número maior, composto por diversos zeros e uns. O sistema numérico binário precisa de uma determinada resolução em números possíveis de bits para poder representar os demais sistemas numéricos, conforme será visto a seguir.

Um outro termo recorrente na análise de sistemas numéricos binários é o de um conjunto de 8 bits, que forma **1 byte**.

Também nas representações por sistemas numéricos binários se empregam os termos "*dígito mais significativo*" e "*dígito menos significativo*".

O *dígito mais significativo* é o **bit mais à esquerda** do número na base binária. Ele é assim chamado pois seu valor é o que possui **maior relevância**, ou seja, representa o número equivalentemente superior na base decimal.

Já o *dígito menos significativo* é o **bit mais à direita** do número na base binária. Ele é assim chamado pois seus valor é o que possui **menor relevância**, ou seja, representa o número equivalentemente inferior na base decimal.

O **sistema numérico octal** (*ou base numérica octal*) pode representar qualquer número com apenas **8 algarismos**: 0, 1, 2, 3, 4, 5, 6, 7.

O **sistema numérico decimal** é o que usamos no nosso dia a dia. Ele pode representar qualquer número com **10 algarismos**: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Já o **sistema numérico hexadecimal** pode representar qualquer número com **16 símbolos**: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

A título de curiosidade segue uma tabela com a equivalência entre os números primordiais da base decimal para as demais bases:

BASE DECIMAL	BASE BINÁRIA	BASE OCTAL	BASE HEXADECIMAL
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D

14	1110	16	E
15	1111	17	F

Para se converter qualquer número da **base decimal** para **outra base**, deve-se empregar **divisões sucessivas pelo número da base de destino**, ou seja, se temos um número "X" na base 10, e queremos representa-lo na base 2, devemos dividir este número "X" por 2 "n" vezes, sempre se anotando o **resto da divisão**. A seguir iremos melhor definir este conceito.

Conversões de uma base qualquer para decimal

Pensemos no número 1010 na **base binária** - *doravante descrito como* ***(1010)₂***.

Queremos encontrar seu equivalente na **base decimal**. O procedimento é o seguinte:

$$(1010)_2 = 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 = 8+0+2+0 = (10)_{10}$$

Ou seja, multiplicamos o número "0" ou "1" na base 2 pelas **potências de 2** representativas de sua **posição** *(ou seja, o mais à esquerda ganha maior relevância - maior expoente - e vai se reduzindo até o expoente zero*).

Pensemos agora no número 12 na **base octal** - ou $(12)_8$. Queremos encontrar seu equivalente na **base decimal**. O procedimento é idêntico, porém agora usamos como **base o número 8**.

$$(12)_8 = 1\times 8^1 + 2\times 8^0 = 8+2 = (10)_{10}$$

Por último, pensemos agora no "*número*" A1 na **base hexadecimal** - ou $(A1)_{16}$. Queremos encontrar seu equivalente na **base decimal**. *Como fazer*?

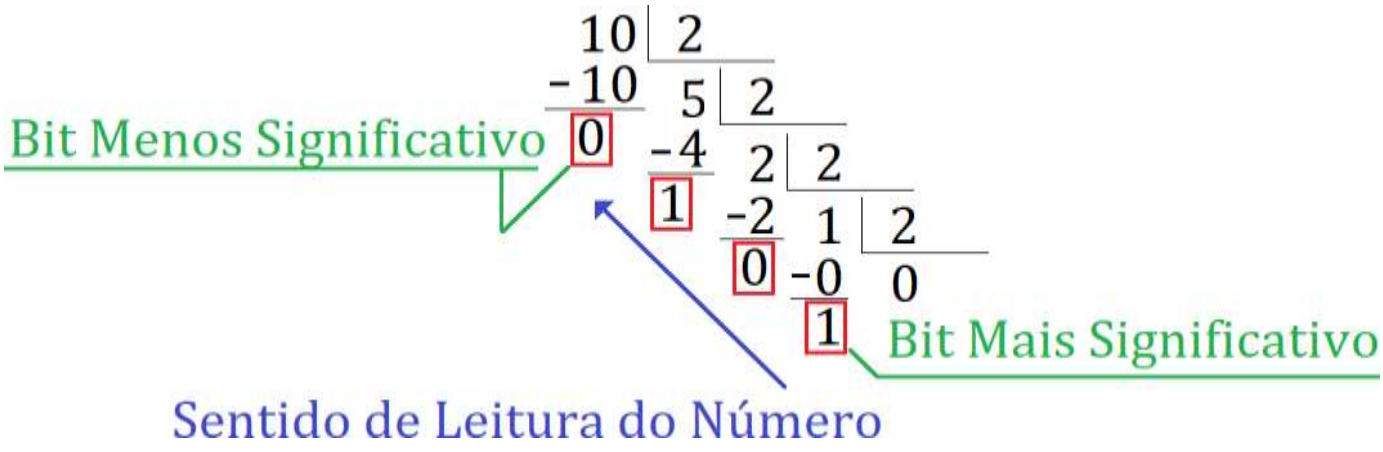
Bom, como não adianta multiplicar "*A1*" por nada, devemos saber **previamente** a correlação das letras da **base hexadecimal** com os seus equivalentes na **base decimal**, conforme disposto na tabela apresentada na página anterior. Sendo assim:

$$(A1)_{16} = 10\times 16^1 + 1\times 16^0 = 160+1 = (161)_{10}$$

Simples, não?

Conversões de decimal para uma base qualquer

Façamos agora o **caminho contrário**. Temos o número 10 na **base decimal** equeremos encontrar seu equivalente na **base binária**. O procedimento é o seguinte:



Divide-se **sucessivamente** o número na **base original** pelo número da **base de destino**, preservando-se o valor dos **restos das divisões**, **até que o quociente da última divisão seja zero**.

Assim o **número na base de destino** será aquele formando pelos **restos das divisões**, no sentido apontado na Figura acima.

Neste caso, o número formado é o $(1010)_2$ corresponde ao número $(10)_{10}$.

(FUNDEP/Prefeitura de Uberlândia - 2019) Em relação aos sistemas digitais, a conversão decimal binário é de fundamental importância. Assinale a alternativa que apresenta o equivalente binário ao número decimal 35.

A) 100001

B) 100111

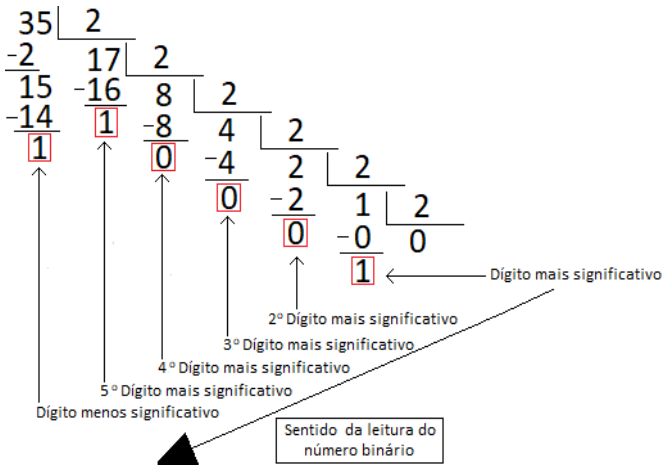
C) 110001

D) 100011

Comentários:

Gabarito: Alternativa B.

Para se fazer a conversão de um **número decimal** para um **número binário** é necessário fazer uma **sucessão de divisões por 2.** O resultado é então lido ao contrário, conforme segue.

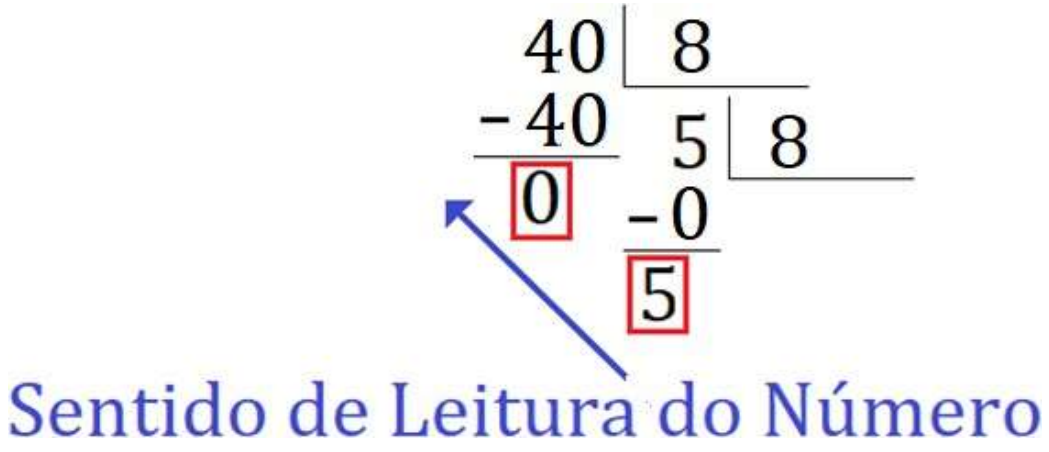


Portanto, o número 35 em decimal é representado em binário pelo número 100011. Assim, a alternativa correta é a letra D.

Façamos agora a conversão do **número 40 na base decimal** - ou $(40)_{10}$ - para a **baseoctal**. O procedimento é o mesmo.

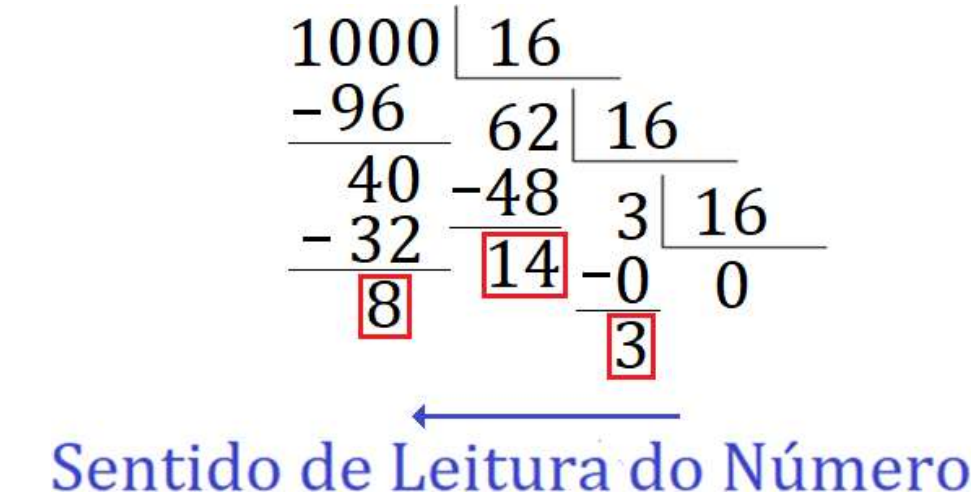
Divide-se sucessivamente o **número na base original** pelo **número da base de destino**, preservando-se o valor dos **restos das divisões, até que o quociente da última divisão seja zero**.

O resultado é o número 50 na base octal - ou $(50)_8$.



Por último, façamos agora a conversão do **número 1000 na base decimal** - ou $(1000)_{10}$

- para a **base hexadecimal**. O procedimento é semelhante.



Como um dos **restos** resultou no **número 14**, e tal número possui um equivalente na **basehexadecimal** (*a letra "E"*), a representação do número $(1000)_{10}$ fica sendo $(3E8)_{16}$.

Conversões entre bases binária, octal e hexadecimal

Para se realizar a **conversão** entre estas **bases**, por elas serem **todas potências de 2**,é possível se empregar uma técnica que torna a **conversão muito mais rápida**.

Tomemos por base o **número binário 010010010011**. Este número possui **12 bits**. Para convertê-lo para a **base octal**, por exemplo, primeiro o separamos em grupos de **3 bits** (*afinal $2^3 = 8$, e buscamos a base **octal***).

$$(010010010011)_2=(010\,010\,010\,011)_2$$

Ok, nada de novo por aqui. Ainda tenho o mesmo número na base 2...

Agora que vem a parte interessante! Vamos escrever estes números tal como se fosse para a base decimal, porém para cada grupo de 3 bits, desta maneira:

$$(010\,010\,010\,011)_2=(\underbrace{010}_{(2)_8}\,\underbrace{010}_{(2)_8}\,\underbrace{010}_{(2)_8}\,\underbrace{011}_{(3)_8})_2$$

Ou seja, o número $(010010010011)_2$ equivale ao número $(2223)_8$.

Vamos fazer o mesmo para a **base hexadecimal**. Tomemos de novo o número $(010010010011)_2$. Para convertê-lo para a **base hexadecimal**, agora o separamos em grupos de 4 (*afinal $2^4 = 16$, e buscamos a base hexadecimal*):

$$(010010010011)_2=(0100\,1001\,0011)_2$$

E fazemos o **mesmo procedimento** adotado anteriormente para a base octal:

$$(0100\,1001\,0011)_2=(\underbrace{0100}_{(4)_{16}}\,\underbrace{1001}_{(9)_{16}}\,\underbrace{0011}_{(3)_{16}})_2$$

Ou seja, o número $(010010010011)_2$ equivale ao número $(493)_{16}$.

E se o número não puder ser separado em grupos uniformes de 3 ou 4 bits?

Boa pergunta! Então você deve agrupá-los sempre em grupos de 3 ou de 4 bits, vindo da direita para a esquerda, e os bits que "*faltarem*" você **preenche com zeros**. Tomemos por exemplo o número $(0101001011)_2$, um número com 10 bits. Vamos convertê-los para as bases **octal** e **hexadecimal**:

$$(0101001011)_2 = \begin{cases} \left(\underbrace{000}_{(0)_8}\underbrace{101}_{(5)_8}\underbrace{001}_{(1)_8}\underbrace{011}_{(3)_8}\right)_2 = (513)_8 \\ \left(\underbrace{0001}_{(1)_{16}}\underbrace{0100}_{(4)_{16}}\underbrace{1011}_{(B)_{16}}\right)_2 = (14B)_{16} \end{cases}$$

Os bits "faltantes" foram destacados na cor cinza claro.

Com isto finalizamos esta parte de representações numéricas e suas conversões. A seguir serão apresentadas as **operações de soma e subtração com números binários, e em seguida as portas lógicas.**

(CESPE/ Telebras - Assistente Técnico - 2015) Acerca de sistemas numéricos, representação interna dos dados e transmissão de dados entre computadores, julgue o seguinte item.

Para converter números representados na base octal em números binários, deve-se separar cada dígito do número octal e substituí-lo por seu valor correspondente binário de quatro *bits*.

A) Certo

B) Errado

Comentários:

Gabarito: B) Errado.

A afirmativa está incorreta. A **base** do sistema de numeração **octal é 8** porque consiste no dígito de **0 a 7**. Por outro lado, a **base do dígito binário é 2** e consiste apenas em dois dígitos **0 e 1**. Para a **conversão de dados da base octal para binário**, **separa-se cada**

dígito da base octal e o converte para seu respectivo binário de 3 bits. A seguinte tabela ilustra as correspondências entre binário e octal:

Binário	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Soma Aritmética com Números Binários

É possível realizar a **soma aritmética** que se faz com os números na **base decimal** também na **base binária**, entretanto não se pode esquecer de algumas peculiaridades.

Na **aritmética binária**, o bit de "*carry*" (*popularmente conhecido como "vai 1"*) é **fundamental** e aparece em praticamente todas as operações.

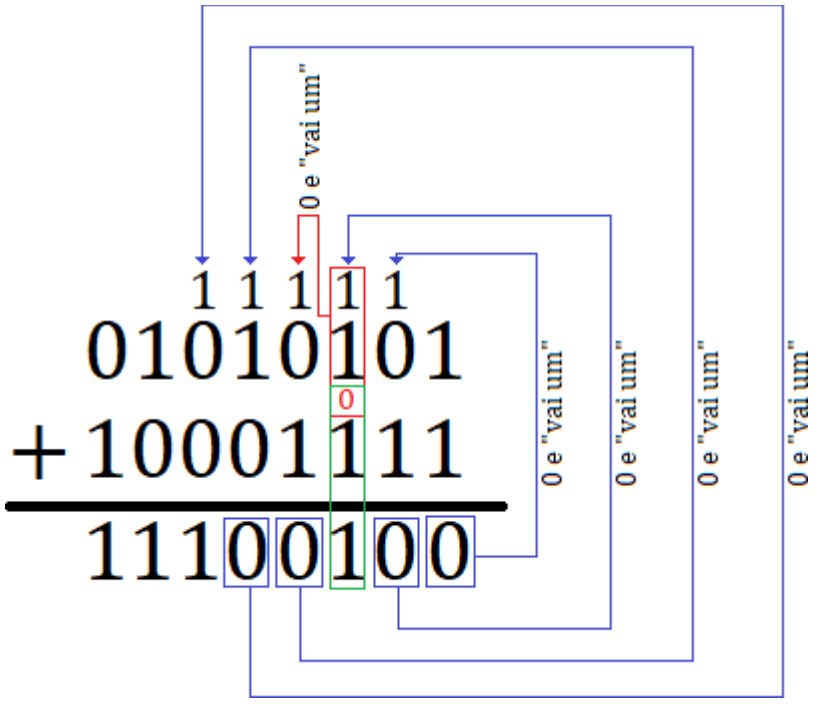
A **aritmética binária** é basicamente feita em cima do que é mostrado na Tabela a seguir.

Bit A	Bit B	Bit A + Bit B
0	0	0
0	1	1
1	0	1
1	1	0 e "*vai um*"

Um **erro comum** é associar a operação "OR" das **operações booleanas** coma **operação aritmética de soma,** pois ambas são definidas pelo operador "+".

Deve-se estar atento ao contexto do que é pedido na questão.

Vamos empregar a **soma de dois bytes A e B**, com o byte A sendo definido por $(01010101)_2$ e o byte B sendo definido por $(10001111)_2$. Começa-se a operação tal como na base decimal, ou seja, da direita para a esquerda.



Bom, se fazemos a soma, então também é possível fazer subtrações, certo?

Sim! Porém, assim como existem regras para se fazer **soma** com números binários existem regras para se fazer **subtração** com números binários.

A **lógica** aqui é a de se somar à parcela da qual se quer subtrair determinada quantidade um **número negativo**, desta forma: consideremos os números $(2)_{10}$ e $(-2)_{10}$ e a operação de soma

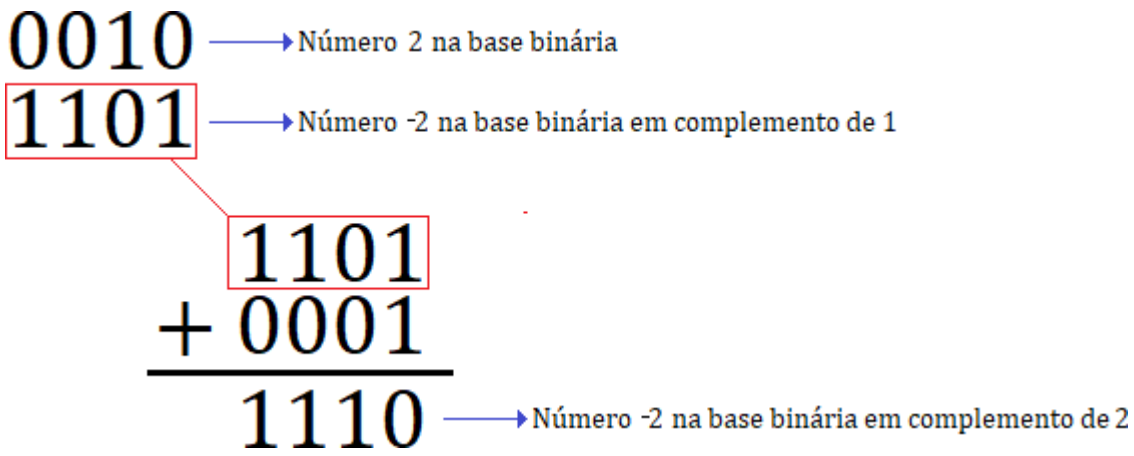
$$2+(-2)=0$$

Para se implementar um **número negativo** na **base binária**, a convenção usada é a de**inverter todos os bits**. Esta é a convenção chamada de **complemento de 1**. Ou seja, sabendo que o número $(0010)_2$ representa o número $(2)_{10}$, então para se obter $(-2)_{10}$ basta **inverter os bits**, ficando $(1101)_2$. E assim teremos sempre o bit mais significativo como sendoo chamado bit de sinal (onde 0 irá representar o sinal "+" e o 1 irá representar o sinal "-").

Com isto, um número binário com **4 bits**, em vez de poder representar de $(0)_{10}$ a $(15)_{10}$,iria representar de $(-7)_{10}$, descrito por $(1111)_2$, até o que viria a ser o **absurdo** de **$(-0)_{10}$**, dadopor $(1000)_2$. E a parte positiva iria de $(+0)_{10}$, **também absurda** e dada por $(0000)_2$, até $(+7)_{10}$,dada por $(0100)_2$.

Por isto se usa em **operações aritméticas com números binários** o **complemento de 2**, que conserta esta deficiência do complemento de 1.

O **complemento de 2** nada mais é que **somar 1 ao número já representado em complemento de 1**. Assim:



Entretanto, agora poderemos representar com **4 bits** os números **$(-8)_{10}$**, dado por $(1111)_2$, até o **$(+7)_{10}$**, dado por $(0111)_2$. O **número zero** na **base binária** fica sendo representado apenas por $(0000)_2$.

Por último, vamos apresentar o conceito de *bit de overflow*. O *overflow* acontece quando uma **operação aritmética** recebe um **bit de *carry*** (*o "vai-um"*) na sua **última operação** e este bit não tem com o que se somar e fica "*sobrando*". Desta maneira:

Soma entre dois números positivos

$$\begin{array}{r} 0010 \\ +0011 \\ \hline 0101 \end{array}$$

Soma entre dois números negativos

$$\begin{array}{r} \text{Overflow} \rightarrow \boxed{1}1100 \\ +1010 \\ \hline 0110 \end{array}$$

Na Figura acima se verifica que a **soma** à sua esquerda, envolvendo os números $(0010)_2$, que representa o número $(2)_{10}$, com o número $(0011)_2$, que representa o número $(3)_{10}$,resulta no número $(0101)_2$, que representa o número $(5)_{10}$. **Tudo correto**, **sem overflow**!

Já na soma ao lado temos **2 números negativos** em **complemento de 2**. O número $(1100)_2$, que representa o número $(-4)_{10}$, e o número $(1010)_2$, que representa o número $(-6)_{10}$, que resultam no número $(0110)_2$, que representa o número $(+6)_{10}$, ***o que está claramente errado***, visto que **a soma de dois números negativos não pode resultar em um número positivo**. **Este erro pode é apontado pelo bit de overflow destacado acima**!

Isto acontece porque com **4 bits** não se consegue representar o número $(-10)_{10}$. Para isto seriam necessários pelo menos **5 bits**.

Variáveis Booleanas

Através das **tabelas verdade** podemos determinar a **saída** de um determinado **circuito lógico**. Tal tabela relaciona **todas as variações possíveis das entradas de um circuito lógico** e apresenta as respectivas **saídas** para cada uma destas combinações. A seguir apresenta-se um exemplo de **tabela verdade** com duas **variáveis de entrada** (A e B)e uma **variável de saída** (Y).

A	B	Y
0	0	0
0	1	1
1	0	0
1	1	1

A primeira linha da tabela mostra as **variáveis de entrada** e as **variáveis de saída**. Note que o número de linhas abaixo da primeira representa as 4 combinações possíveis destas duas variáveis (A = 0 e B = 0; A = 0 e B = 1; A = 1 e B = 0; A = 1 e B = 1).

Se fossem 3 variáveis de entrada, então haveria 8 combinações. Se fossem 4 variáveisde entrada, haveria 16 combinações.

Percebe como o número de combinações é sempre 2^N, onde N é o número de **variáveis de entrada**? Isso sempre irá ocorrer.

O resultado deste circuito lógico é igual às combinações onde a saída seja "1". Em outras palavras, **nesta tabela verdade**, sempre que a **entrada B** for igual a "1", **a saída Y** será também igual a "1", então podemos dizer que Y = B, pois o valor lógico de A não irá importar no resultado deste circuito lógico.

Portas Lógicas

Algumas combinações de **entradas** já possuem blocos padronizados, chamados de **portas lógicas**. Estas **portas lógicas** são denominadas de acordo com a sua respectiva função:

- Porta **AND** (ou seja, realiza a função lógica "**E**");
- Porta **OR** (ou seja, realiza a função lógica "**OU**");
- Porta **NOT** (ou seja, realiza a função de **inverter o valor lógico** de sua entrada)

- Porta **XOR** (ou seja, realiza a função lógica de "**OU-EXCLUSIVO**").

Os blocos apresentados demonstram sempre **2 entradas** (*com exceção da porta NOT, que apresenta uma única entrada*), porém isto **não é uma regra**. As portas que realizam operação entre **pelo menos 2 entradas**, podem realizar com **N entradas**, sem problema algum, bastando que para isto se modifique sua tabela verdade.

Estes blocos podem ser encadeados uns nos outros, de modo que formem um **circuito lógico** mais complexo. A análise de um **circuito lógico** com diversas portas encadeadas virá a seguir através de um exemplo.

Nas **expressões lógicas** que seguem, sempre que **uma barra** aparecer sobre a **variável booleana**, indica que esta variável está "*negada*", ou seja, seu valor lógico originário na **tabela verdade** que deu origem à expressão era 0. Quando a **variável booleana** aparecer **sem barra**, indica que seu valor lógico originário na **tabela verdade** que deu origem à expressão era 1.

Estes blocos padronizados são os que seguem:

PORTA LÓGICA "OR"

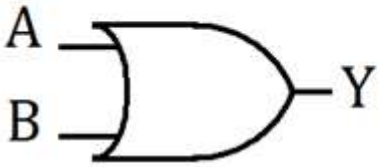


TABELA VERDADE

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

EXPRESSÃO LÓGICA

$$Y = A + B$$

A saída de uma porta **OR** será 1 sempre que pelo menos uma das entradas possuirvalor lógico 1.

PORTA LÓGICA "AND"

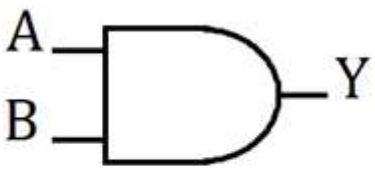


TABELA VERDADE

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

EXPRESÃO LÓGICA

$$Y = A \cdot B$$

A saída de uma porta **AND** será 1 sempre que todas as entradas forem iguais a 1.

PORTA LÓGICA "NOT"

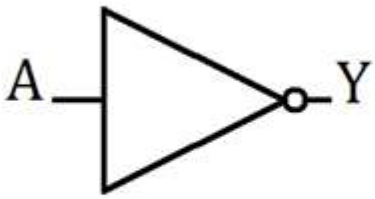


TABELA VERDADE

A	Y
0	1
1	0

EXPRESSÃO LÓGICA

$$Y = \overline{A}$$

A saída de uma porta **NOT** será sempre o inverso do valor lógico da entrada.

PORTA LÓGICA "XOR"

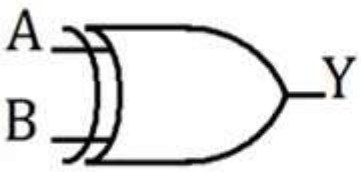


TABELA VERDADE

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

EXPRESSÃO LÓGICA

$$Y = A.\overline{B} + \overline{A}.B = A \oplus B$$

A saída de uma porta **XOR** será sempre 1 sempre que as entradas possuírem valores lógicos diferentes entre si.

Há ainda as portas lógicas **NAND** (ou seja, operação **AND** acrescida da operação **NOT**)e **NOR** (ou seja, operação **OR** acrescida da operação **NOT**). Tais portas **são muito importantes** na implementação de circuitos lógicos de maneira **mais eficiente**. Porém isto será visto *a posteriori*, após a apresentação de **alguns teoremas booleanos fundamentais**.

Teoremas Booleanos e de De Morgan

Os **teoremas booleanos** são **regras** para tratarmos de **circuitos lógicos**. São eles:

Teoremas para a Operação AND:

1. $X \cdot 0=0$, ou seja, qualquer variável booleana que faça uma operação **AND** com um valorlógico 0, terá como saída o valor lógico 0.

2. $X \cdot 1=X$, ou seja, qualquer variável booleana que faça uma operação AND com um valor lógico 1, terá como saída a própria variável booleana.

3. $X \cdot X=X$, ou seja, qualquer variável booleana que faça uma operação **AND** consigo mesma, terá como saída o valor da própria variável booleana.

4. $X \cdot \overline{X} = 0$ ou seja, qualquer variável booleana que faça uma operação **AND** consigo mesma negada, terá como saída o valor lógico 0.

Teoremas para a Operação OR:

1. $X + 0 = X$, ou seja, qualquer variável booleana que faça uma operação **OR** com um valor lógico 0, terá como saída a própria variável.

2. $X + 1 = 1$, ou seja, qualquer variável booleana que faça uma operação **OR** com um valor lógico 1, terá como saída sempre o valor lógico 1.

3. $X + X = X$, ou seja, qualquer variável booleana que faça uma operação **OR** consigo mesma, terá como saída o valor da própria variável booleana.

4. $X + \overline{X} = 1$, ou seja, qualquer variável booleana que faça uma operação **OR** consigo mesma negada, terá como saída o valor lógico 1.

Além destes, há ainda um outro teorema que é **fundamental** para a implementação decircuitos lógicos eficientes, o chamado **Teorema de De Morgan**. Este teorema nos diz que:

$$\underbrace{\overline{(A \cdot B)}}_{\text{Operação NAND}} = \overline{A} + \overline{B} \qquad \underbrace{\overline{A + B}}_{\text{Operação NOR}} = \overline{A} \cdot \overline{B}$$

Todas as **expressões booleanas** consistem em combinações das operações básicas aqui vistas (**OR**, **AND** e **NOT**). Entretanto, é possível implementar qualquer expressão booleana empregando **APENAS** portas **NAND** e portas **NOR**. A esta característica se dá o nome de universalidade das portas **NAND** e **NOR**. E tal universalidade advém diretamente do**Teorema de De Morgan**.

Os **circuitos integrados** que implementam as **funções lógicas** possuem **não uma**, **porém várias portas lógicas implementadas em si**. Se em um determinado circuito lógico tivermos a necessidade de implementar apenas uma porta **OR**, e diversas outras portas **AND** e **NOT**, teremos "*desperdiçado*" espaço em nossa placa com um circuito integrado que faráapenas uma função.

Em vez disso, podemos **trocar todas as nossas portas lógicas** por suas equivalentes**NAND** ou **NOR** e implementarmos em nossa placa **apenas** portas **NAND** ou **NOR**. A probabilidade de que economizemos espaço em nossa placa é **muito maior**. Isso gerará também a chamada **economia de escala**, uma vez que se nosso circuito for ser implementado em uma linha de montagem, poderemos comprar somente circuitos integrados de portas **NAND**, gerando economia de custos com o fornecedor, pois agora compraremos inúmeros circuitos integrados de portas **NAND**, em vez de conjuntos menores de circuitos integrados de portas **NOT**, **AND** e **OR**.

Mas, antes de prosseguirmos, vamos apresentar as portas **NAND** e **NOR**, do mesmo jeito que apresentamos as portas **OR**, **AND**, **NOT** e **XOR**.

PORTA LÓGICA "NOR"

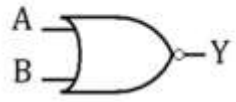


TABELA VERDADE

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

EXPRESSÃO LÓGICA

$$Y = \overline{A + B}$$

PORTA LÓGICA "NAND"

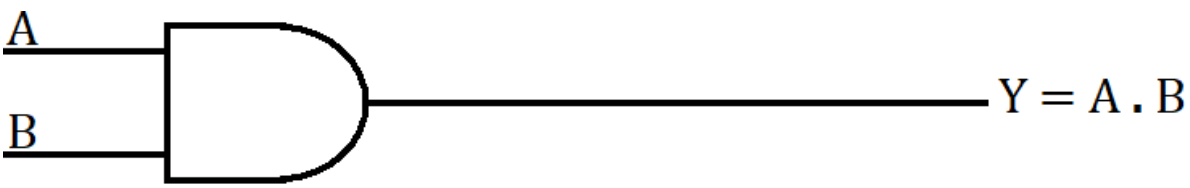
TABELA VERDADE

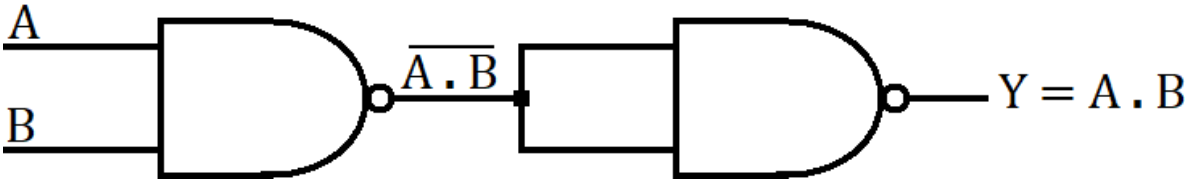
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

EXPRESÃO LÓGICA

$$Y = \overline{A \cdot B}$$

Exemplo: *Como implementar uma porta **AND** através de portas **NAND**?* Perceba que na representação de cima a porta **AND** possui duas entradas: A e B.





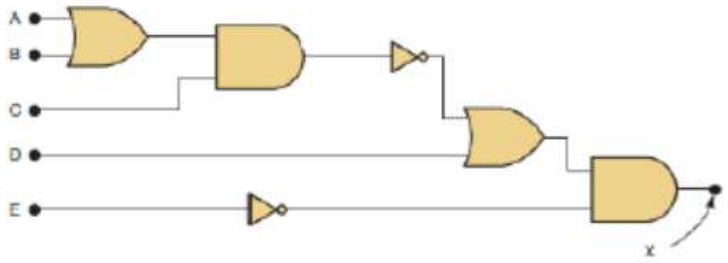
Na representação acima temos as mesmas duas entradas A e B entrando numa porta **NAND**. A saída é $\overline{A \cdot B}$. Em seguida, fazemos com que esta saída $\overline{A \cdot B}$ entre nas duas entradas da **NAND** posterior.

*Dos teoremas de portas **AND** temos que $X \cdot X = X$, lembra?*

Acontece que agora é uma porta **NAND**, que nada mais é do que uma porta **AND** com uma outra porta **NOT** **acopladada à sua saída**. Sendo assim, o resultado da expressão é o mesmo Y = A . B da primeira.

FGV/IMBEL - Engenheiro de Telecomunicações - 2021

Analise o circuito digital a seguir.



Defina a expressão x:

a) $[D + \overline{(A+B)C}]\bar{E}$

b) $[D + \overline{(A+B)C}]E$

c) $[\bar{D} + (A+B)C]E$

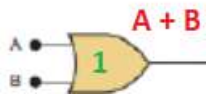
d) $[D + (A+B)\bar{C}]\bar{E}$

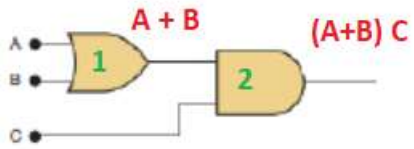
e) $[D + \overline{(A+B)\bar{C}}]\bar{E}$

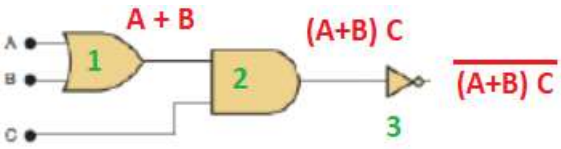
Comentários

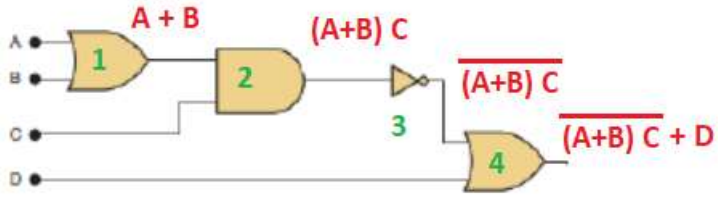
Gabarito - Letra A

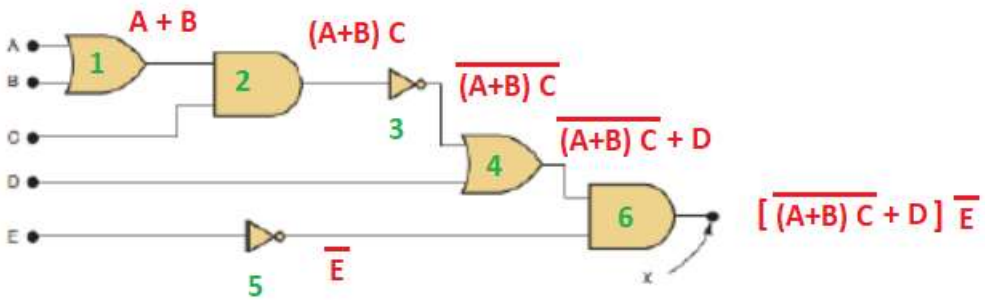
Podemos obter a expressão na saída de cada uma das portas lógicas, as quais foram numeradas de 1 a 6:











Portanto, a expressão x é dada por:

$$x = \left[D + \overline{(A+B)C}\right]\bar{E}$$

Logo, o gabarito da questão é **Letra A**.

Bom, a partir do conhecimento dos **conceitos** apresentados até aqui, que são **fundamentais**, já estamos aptos para aprender sobre os **circuitos digitais.**

CIRCUITOS DIGITAIS

A partir de agora veremos as técnicas de simplificação de projetos de **circuitos lógicos**.

Para que se possa simplificar **circuitos lógicos**, é necessário que eles estejam expressos na forma de **soma de produtos**. *Mas o que seria isso*? Bom, eis alguns exemplos:

$$Y = ABC + \overline{A}\,B\overline{C} \quad Y = A\overline{B} + \overline{C}\,D \quad Y = \overline{A}\,B + A\overline{B}C + A\overline{C}$$

Simplificação Algébrica

Após se obter a expressão de um **circuito lógico**, pode ser que seja necessário **simplificá-lo**, de modo a se obter a expressão com **menor número de operações lógicas possível**. Um dos métodos de simplificação do circuito é a **simplificação algébrica**.

Por exemplo, tomemos a expressão:

$$Y = ABC + A\overline{B}\cdot\left(\overline{\overline{A}\,\overline{C}}\right)$$

Primeiro vamos "quebrar" as barras de *AC*, aplicando o **Teorema de De Morgan**, ficando:

$$Y = ABC + A\overline{B}\cdot\left(A + C\right)$$

Vamos agora **expandir** o $A\overline{B}\cdot\left(A + C\right)$:

$$Y = ABC + A\overline{B}A + A\overline{B}C$$

Sabendo do teorema $X \cdot X = X$, podemos **simplificar** a expressão $A\overline{B}A$ acima:

$$Y = ABC + A\overline{B} + A\overline{B}C$$

Agora vamos procurar por variáveis em comum e **fatorá-las**:

$$Y = AC\left(\underbrace{B + \overline{B}}_{=1}\right) + A\overline{B} \quad \rightarrow \quad Y = AC + A\overline{B} \quad \rightarrow \quad Y = A\left(\overline{B} + C\right)$$

E assim, a nossa expressão inicial $\boldsymbol{Y = ABC + A\overline{B}\left(\overline{A\overline{C}}\right)}$ se resumiu a $\boldsymbol{Y = A(\overline{B} + C)}$.

Projetos de Circuitos Lógicos Combinacionais

Para se projetar **circuitos lógicos**, primeiramente precisamos lançar mão da técnica da **tabela verdade**. Nela escrevemos as nossas **entradas** e as nossas **saídas**, bem como todas as possibilidades de combinações entre as entradas e suas correspondentes saídas.

Feito isto, selecionamos as **entradas** onde o resultado lógico foi igual a 1, escrevendo a expressão como uma **soma de produtos**. O exemplo a seguir ilustra este procedimento.

Imagine um **circuito lógico** que possua 3 entradas (**A**, **B** e **C**) e 1 saída **Y** e que se monta a **tabela verdade** deste circuito lógico obtendo-se o seguinte resultado.

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Vamos agora retirar da **tabela verdade** os valores que nos interessam para formar a resposta do nosso **circuito lógico**.

O primeiro valor que nos interessa é quando $Y = \bar{A}\bar{B}C$ (ou seja, Y é "um" quando A é "zero", B é "zero" e C é "um").

O segundo valor é quando $Y = \bar{A}B\bar{C}$. O terceiro valor é quando $Y = A\bar{B}\bar{C}$. O quarto valor é quando $Y = A\bar{B}C$. O quinto valor é quando $Y = ABC$. Vamos agora montar a **expressão lógica** através de uma **soma de produtos**.

$$Y = \overline{A}\,\overline{B}C + \overline{A}\,B\overline{C} + A\,\overline{B}\overline{C} + A\,\overline{B}C + ABC$$

Esta é a nossa **expressão base**. Podemos simplificá-la através de **simplificação algébrica**, ou então fazer uso de uma **poderosa técnica de simplificação** chamada **Mapa de Karnaugh**, que veremos a seguir, após o exemplo.

FGV/IMBEL - Engenheiro de Telecomunicações - 2021

A figura a seguir apresenta uma tabela verdade de uma porta lógica com entradas A e B e saída S.

TABELA VERDADE

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

Esta tabela é referente a uma porta

A) OR.
B) NOR.
C) AND.
D) NAND.

E) XOR.

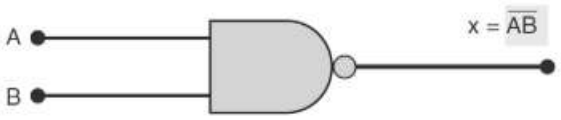
Comentários

Gabarito – Alternativa D

Observando a tabela verdade, é possível dizer que a lógica NAND é inversa à lógica AND, onde a saída será nível BAIXO (zero) quando todas as entradas forem níveis ALTO (um).

		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

NAND significa “NOT AND”, o que denota sua principal característica – ser uma função de complemento em relação à função AND, e podemos ver seu símbolos na figura a seguir:



Portanto, a tabela verdade da questão representa a **porta lógica NAND,** sendo o gabarito da questão **Letra D.**

Mapas de Karnaugh

O **Mapa de Karnaugh** é um **método gráfico** usado para **simplificar expressões lógicas de até 4 variáveis**. Acima de 4 variáveis ainda é possível usar Mapas de Karnaugh, entretanto o nível de abstração passa a ser cada vez mais elevado, fazendo-se necessário o uso de ferramentas computacionais para implementá-lo.

Os **Mapas de Karnaugh**, para os fins dos nossos estudos, serão apenas de 2, 3 ou 4 variáveis.

A equivalência entre **tabela verdade** e **Mapa de Karnaugh** de 2 variáveis é mostrado na Figura a seguir.

Equivalência entre
Tabela Verdade e Mapa de Karnaugh

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

A \ B	0	1
0	1	1
1	1	0

A montagem do **Mapa de Karnaugh** se dá através da inserção dos **valores lógicos da entrada A** no lado de fora do mapa (*na vertical*) e dos **valores lógicos da entrada B** tambémno lado de fora do mapa (*na horizontal*). Os **valores lógicos da saída** são colocados dentro do mapa, nas respectivas "*coordenadas*" formadas pelos valores lógicos das entradas.

Para se obter a **expressão lógica correspondente à saída**, destaca-se no mapa os valores lógicos 1 que sejam adjacentes na **horizontal** ou na **vertical**, sempre em conjuntos cujo número de componentes seja uma potência de 2 (*2, 4, 8 etc.*).

Para melhor entendimento, façamos agora um comparativo entre o método de obtenção do **circuito lógico equivalente** pelo **método de simplificação algébrica** e pelo **método do Mapa de Karnaugh**.

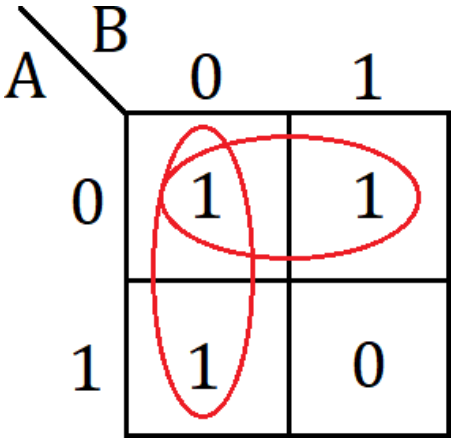
Método da Simplificação Algébrica:

A expressão resultante da **tabela verdade** é:

$$Y=\overline{A}\,\overline{B}+\overline{A}\,B+A\,\overline{B} \quad \rightarrow \quad Y=\overline{A}\left(\underbrace{\overline{B}+B}_{=1}\right)+A\,\overline{B} \quad \rightarrow \quad Y=\overline{A}+A\,\overline{B}$$

Método do Mapa de Karnaugh:

Observando-se as marcações na figura, verifica-se que na primeira linha a **variável de entrada A não muda de valor lógico** (*permanece igual a 0*). Na primeira coluna verifica-se que é a **variável de entrada B que não muda de valor lógico** (*permanece igual a 0*).



Desta maneira, a **expressão lógica de saída** fica sendo

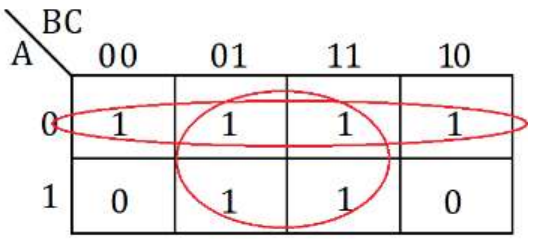
$$Y=\overline{A}+\overline{B}$$

Percebe que o **Mapa de Karnaugh** nos leva **direto** a uma **resposta mais simples** do que a simplificação algébrica? A expressão $Y = \bar{A} + A\bar{B}$ não está errada, apenas não está simplificada o máximo possível.

Vejamos agora como é montado o **Mapa de Karnaugh** para **3 variáveis**.

Equivalência entre
Tabela Verdade e Mapa de Karnaugh

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Podemos ver que neste mapa pudemos selecionar com apenas 2 marcações todos os valores lógicos "1". A expressão lógica resultante fica sendo:

$$Y = \bar{A} + C$$

Na primeira linha (*onde marcou-se a linha inteira*), **apenas a variável lógica A não muda devalor lógico** (*igual a 0*) ao longo de todas as variações de B e de C, por isso que na **expressão de saída** entra o $\bar{A}$ apenas.

Já na marcação central, verifica-se que a **variável A muda de valor** e que a **variável B também tem seu valor comutado**. Apenas a **variável C não muda** (*permanece sendo igual a1*). Por isto que entra o C na **expressão de saída**.

Vejamos agora como é montando o **Mapa de Karnaugh** para **4 variáveis**.

Equivalência entre Tabela Verdade e Mapa de Karnaugh

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

AB \ CD	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	1	1	1
10	1	0	0	1

Neste mapa também pudemos envolver todas os níveis lógicos iguais a 1 com **apenas duas marcações de 8 elementos**.

Perceba que inclusive fizemos uma marcação que dá **a volta no mapa**. Isto é **perfeitamente possível** e **altamente recomendável** se você quiser obter a **menor expressão possível**!

A nossa resposta final fica sendo:

$$Y = B + \overline{D}$$

HORA DE PRATICAR!

(FGV/Prefeitura de Salvador - 2019) Considere a tabela verdade de um dado circuito digital, a seguir.

a	b	c	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

A saída Y desse circuito é:

A) $Y=bc+a\bar{c}+\bar{a}c+\bar{a}b$

B) $Y=b\bar{c}+a\bar{c}+\bar{a}c+\bar{a}b$

C) $Y=b\bar{c}+a\bar{c}+\bar{a}c+ac$

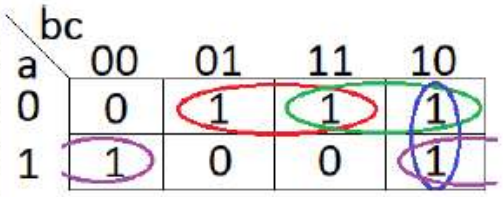
D) $Y=b\bar{c}+a\bar{c}+bc+\bar{a}b$

E) $Y=b\bar{c}+a\bar{c}+\bar{a}\bar{c}+\bar{a}b$

Comentários:

Gabarito: Alternativa B.

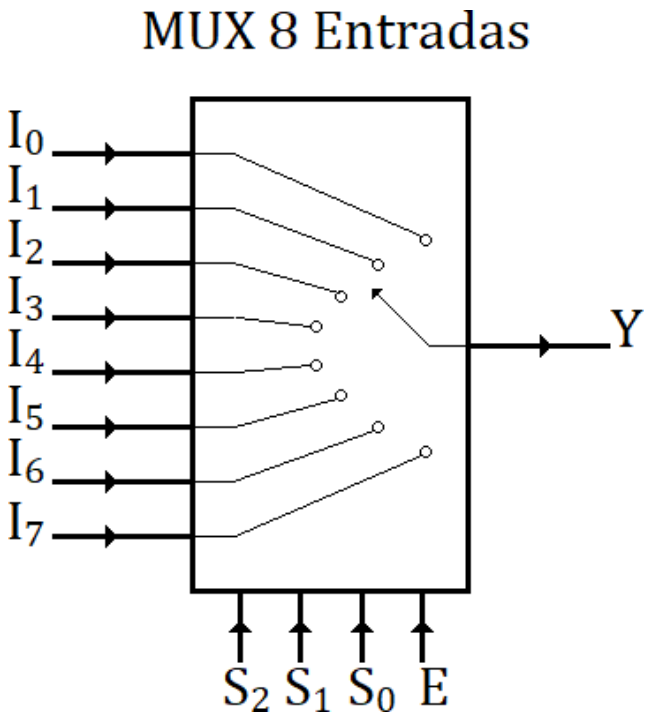
Desenhando o **Mapa de Karnaugh** desta tabela verdade de **3 bits** e associando os termos que podem ser associados entre si. Lembrando que os **termos são agrupáveis sempre em potências de 2** (ou seja, 2^0, 2^1, 2^2, ...) e **sempre com vizinhos na horizontal ou vertical**.



E a expressão que ele representa fica sendo: $Y=\overbrace{\bar{a}c}^{Vermelho}+\underbrace{\bar{a}b}_{Verde}+\overbrace{b\bar{c}}^{Azul}+\underbrace{a\bar{c}}_{Roxo}$.

Multiplexadores e Demultiplexadores

Os **multiplexadores** (*comumente referidos apenas como MUX*) são nada mais que **seletores de dados**. Ele seleciona um dos diversos sinais de entrada que um dispositivo possui e o redireciona para a saída. A Figura a seguir ilustra um **MUX**.



Os **MUX** possuem **N bits de controle** (*entradas S_X na figura ao lado*), um bit de "*Enable*" (*dado pela entrada E na figura ao lado, que habilita ou não o MUX para funcionar*), 2^N entradas de dados (*entradas I_X na figura ao lado*) e uma saída (*pino Y na figura ao lado*).

A tabela verdade do MUX (*que é proporcional à quantidade de entradas que se pretende ter*) é então a que segue na página seguinte.

Dela podemos montar o diagrama do circuito lógico, composto de portas **AND**, **OR** e **NOT**.

Os **multiplexadores** são úteis em aplicações que precisam de **seleções de dados**, **roteamento de dados**, **sequenciamento de operações**, **geração de formas de onda** e **geração de funções lógicas**.

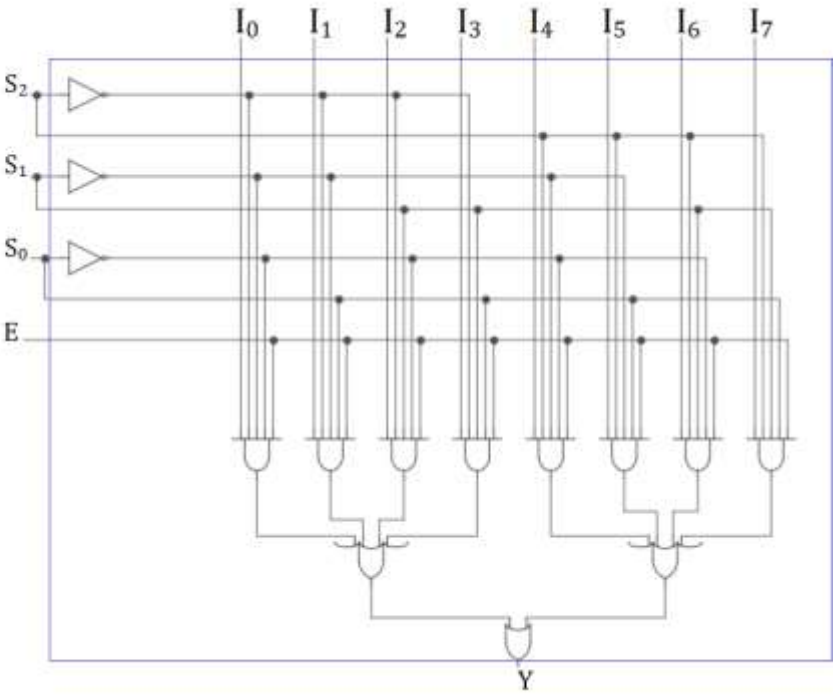
Uma aplicação de **multiplexador** pode ser, por exemplo, numa mesa de som, onde temos entradas individuais de diversos instrumentos e o operador quer, por algum motivo qualquer, escutar somente o vocalista, ou somente a guitarra, ou somente a bateria etc. Em vez de ter que trazer

todos os sinais de áudio diretamente até a mesa de som, é possível traze-los até o **MUX** e o operador verificar se cada sinal está bom individualmente, um de cada vez.

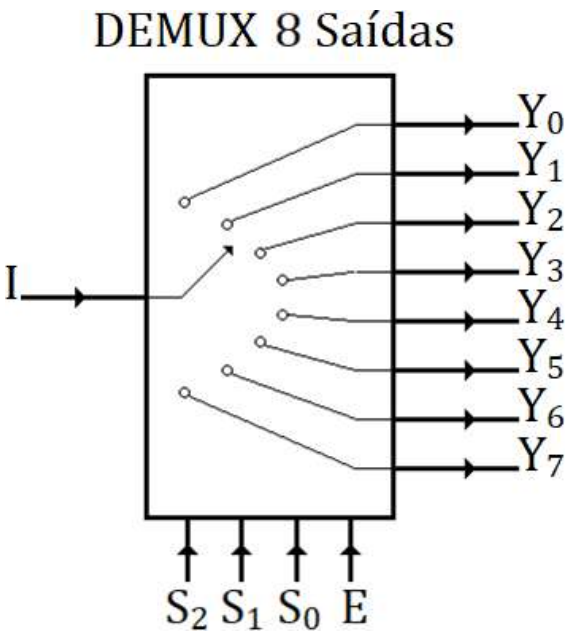
TABELA VERDADE

E	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	I_0
1	0	0	1	I_1
1	0	1	0	I_2
1	0	1	1	I_3
1	1	0	0	I_4
1	1	0	1	I_5
1	1	1	0	I_6
1	1	1	1	I_7

CIRCUITO LÓGICO DO MUX DE 8 ENTRADAS



Já os **demultiplexadores** (ou **DEMUX**) realizam a função inversa do **MUX**, ou seja, se os **MUX** são os **seletores de dados**, os **DEMUX** são os **distribuidores de dados**. Ele recebe uma única entrada de dados e a distribui para 2^N saídas, onde N é o número de bits de controle do DEMUX. A Figura a seguir ilustra um DEMUX.



Os **DEMUX** possuem **N bits de controle** (*entradas S_X na figura ao lado*), um bit de "*Enable*" (*dado pela entrada E na figura ao lado, que habilita ou não o DEMUX para funcionar*), 2^N saídas de dados (*saídas Y_X na figura ao lado*) e uma entrada (*pino I na figura ao lado*).

A tabela verdade do **DEMUX** (*que será proporcional à quantidade de saídas que se pretende ter*) é então a que segue na página seguinte.

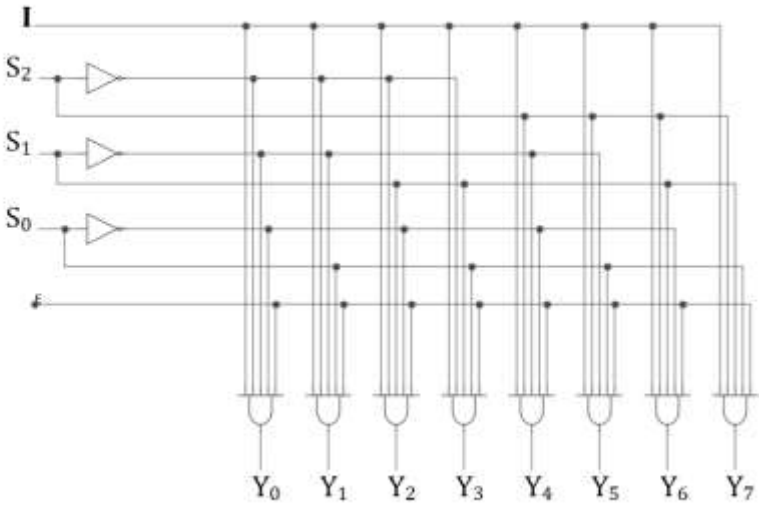
Dela podemos montar o diagrama do circuito lógico, composto de portas **AND**, **OR** e **NOT**.

Uma das aplicações de um **DEMUX** é, por exemplo, no caso de um sistema de monitoramento de segurança. Imagine que diversas portas precisam estar fechadas para que dê início a um processo químico dentro de uma indústria. Cada uma destas portas conta com um sensor que envia "1" se está fechada e "0" se está aberta. Em vez de monitorar todas as portas individualmente, é possível utilizar um **DEMUX**. O sistema pode fazer uma "varredura" nas entradas S2, S1 e S0 (por exemplo), e com isso o operador poder visualizar se alguma porta permanece aberta e tomar a providência necessária que ela seja fechada.

TABELA VERDADE

E	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	I
1	0	0	1	0	0	0	0	0	0	I	0
1	0	1	0	0	0	0	0	0	I	0	0
1	0	1	1	0	0	0	0	I	0	0	0
1	1	0	0	0	0	0	I	0	0	0	0
1	1	0	1	0	0	I	0	0	0	0	0
1	1	1	0	0	I	0	0	0	0	0	0
1	1	1	1	I	0	0	0	0	0	0	0

CIRCUITO LÓGICO DO DEMUX DE 8 SAÍDAS



Passemos agora aos *flip-flops*, importante **elementos básicos** de memória em circuitos integrados e da **Eletrônica Digital** como um todo.

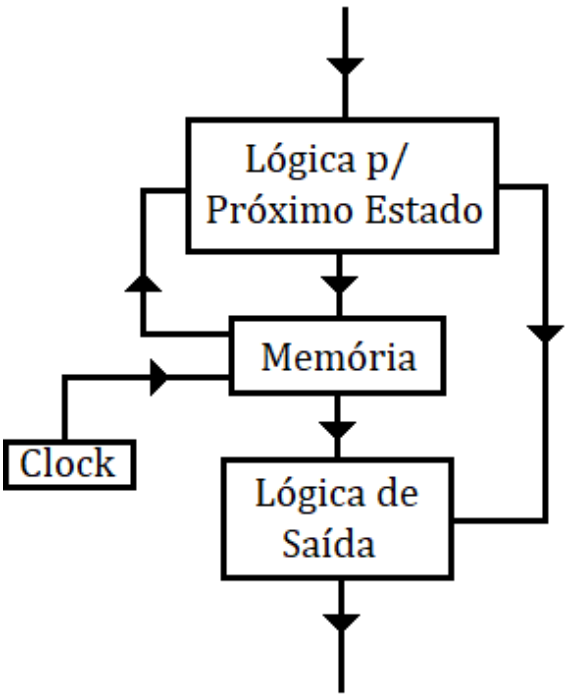
Circuitos Sequenciais

Até este momento estudamos somente os **circuitos lógicos combinacionais**, ou seja, dependem **somente do instante presente** para realizar as funções lógicas nele implementadas.

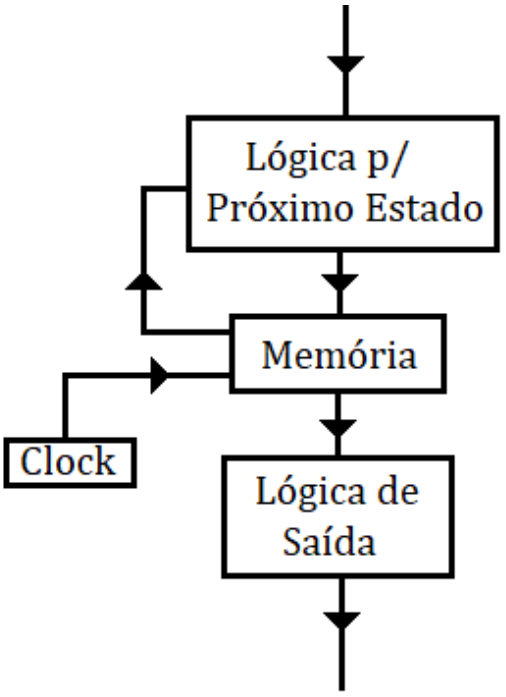
Estudaremos agora dispositivos que **guardam ou alteram** o estado lógico frente a alguma **combinação de entradas**. Estes elementos podem ser entendidos como as memórias de um circuito lógico. São os chamados **circuitos sequenciais**.

Máquinas de Estados

O **funcionamento** dos circuitos sequenciais pode ser representado por uma **máquina de estado**. Estas máquinas de estado possuem duas topologias: **Máquina de Moore e Máquina de Mealy**. Abaixo elas são descritas através de diagrama de blocos.



Máquina de Estados
Topologia de Mealy



Máquina de Estados
Topologia de Moore

Na **máquina de Moore**, a saída depende somente dos estados dos elementos de memória do circuito sequencial (*flip-flops, a serem vistos mais adiante neste capítulo*). A saída somente irá mudar quando houver um pulso de *clock (também será explicado mais adiante neste capítulo)*.

Já na **máquina de Mealy**, a saída depende tanto dos estados dos elementos de memória quanto das entradas do circuito sequencial. A saída poderá mudar a qualquer instante de tempo, não dependendo exclusivamente de um pulso de *clock*.

Podemos representar as máquinas de **Moore** e de **Mealy** através de diagramas de estados. No diagrama de estados de uma **máquina de Moore**, os arcos recebem a descrição apenas dos parâmetros de entrada que "autorizam" aquela transição, ou seja, a transição indicada somente ocorre quando as entradas ali descritas assumirem aquelas condições e vier um pulso de *clock*.

Já no diagrama de estados de uma **máquina de Mealy**, os arcos recebem a descrição tanto dos parâmetros de entrada que permitem aquela transição, quanto dos parâmetros de saída que também são necessários para a transição.

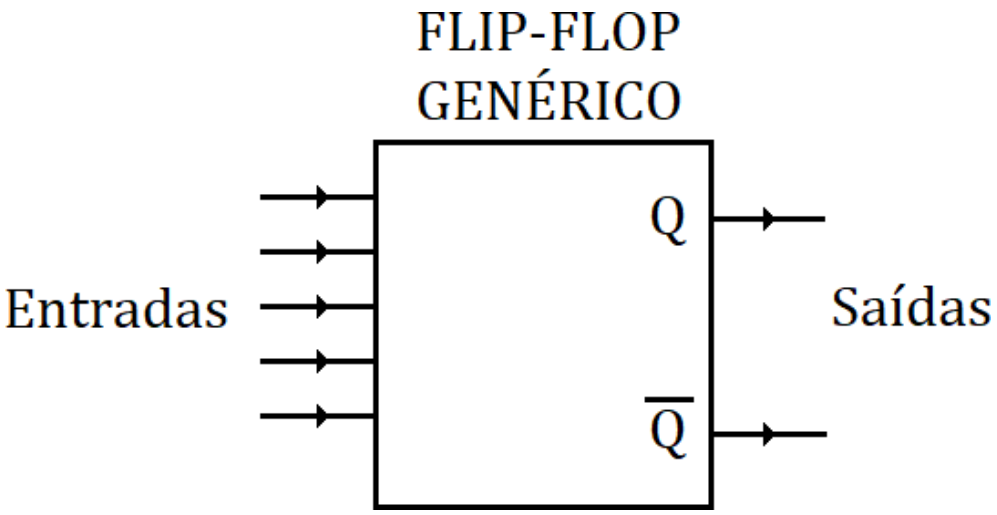
Com ambas as máquinas se pode montar circuitos lógicos sequenciais equivalentes entre si, entretanto, cada uma possui suas **vantagens** e **desvantagens**.

A **máquina de Moore**, por exemplo, permite um projeto mais simples, enquanto a **máquina de Mealy** permite montar diagramas com menores números de estados.

Flip-Flops

O elemento de memória mais **importante** é o chamado ***flip-flop***, implementado a partir de portas lógicas. Embora portas lógicas não tenham memória, combinações entre elas podem gerar o efeito de memória, permitindo o armazenamento de determinada informação.

A seguir se apresenta o que seria um ***flip-flop* genérico** (*existem alguns tipos, conforme veremos em seguida*). O ***flip-flop* genérico** possui **duas saídas** (Q *e* $\overline{Q}$) A saída Q é a "**saída normal**" do flip-flop, enquanto a saída $\overline{Q}$ é a "**saída invertida**" do ***flip-flop***.



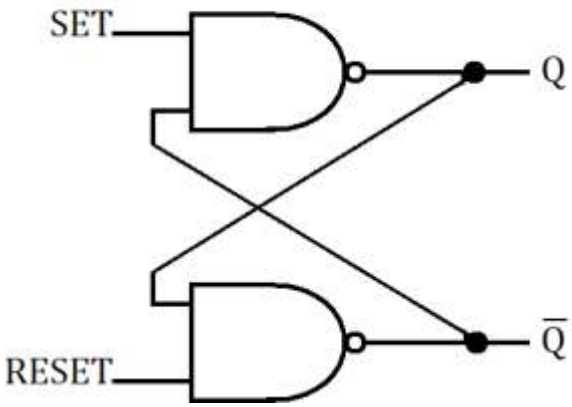
O estado **ALTO** do ***flip-flop*** (Q=1) também é conhecido como **SET**. Sempre que as entradas do ***flip-flop*** levarem para Q=1, dizemos que o ***flip-flop*** foi "*setado*".

O estado **BAIXO** do ***flip-flop*** (Q=0) também é conhecido como **RESET**. Sempre que asentradas do ***flip-flop*** levado para Q=0, dizemos que o ***flip-flop*** foi "*resetado*".

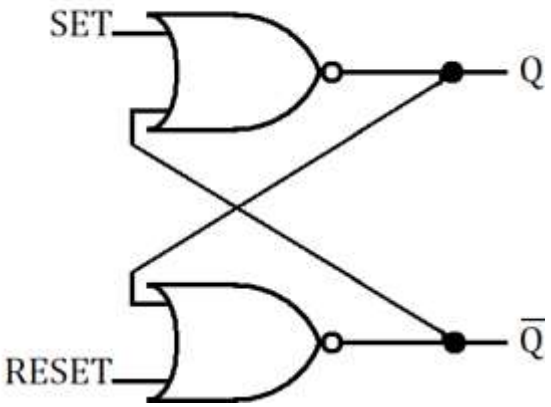
Outros nomes para o ***flip-flop*** é e *multivibrador biestável*, sendo este é o nome mais técnico do ***flip-flop***, enquanto o primeiro se aplica mais a alguns tipos de flip-flop apenas.

Porém, um circuito ainda mais simples de ***flip-flop*** pode ser concebido com **apenas duas portas lógicas do tipo NAND**, ou duas portas lógicas do tipo **NOR**, conforme a figura que segue.

Latch com Portas NAND



Latch com Portas NOR



Este é chamado de *latch* de portas **NAND**, ou *latch* de portas NOR, ou simplesmente *latch*. O funcionamento deste circuito é descrito na Tabela a seguir.

LATCH COM PORTAS NAND			LATCH COM PORTAS NOR		
SET =1	**RESET =1**	**$Q = Q_{ANTERIOR}$** Estágio de repouso do *latch*. Nada acontece neste estágio e o *latch* mantém na sua saída o que já havia antes.	**SET =0**	**RESET =0**	**$Q = Q_{ANTERIOR}$** Estágio de repouso do *latch*. Nada acontece neste estágio e o*latch* mantém na sua saída o que já havia antes.
SET =0	**RESET =1**	**Q = 1** A saída vai para 1 e permanece em 1 mesmo que SET volte a ser 1.	**SET =0**	**RESET =1**	**Q = 0** A saída vai para 0 e permanece em 0 mesmo que RESET volte a ser 0.
SET =1	**RESET =0**	**Q = 0** A saída vai para 0 e permanece em 0 mesma que RESET volte aser 1.	**SET =1**	**RESET =0**	**Q = 1** A saída vai para 1 e permanece em 1 mesma que SET volte a ser0.
SET =0	**RESET =0**	**$Q = Q_{INDEFINIDO}$** Estágio não utilizado, pois, o *latch* tenta fazer com que a saída seja 1 e 0 ao mesmo tempo.	**SET =1**	**RESET =1**	**$Q = Q_{INDEFINIDO}$** Estágio não utilizado, pois, o *latch* tenta fazer com que a saída seja 1 e 0 ao mesmo tempo.

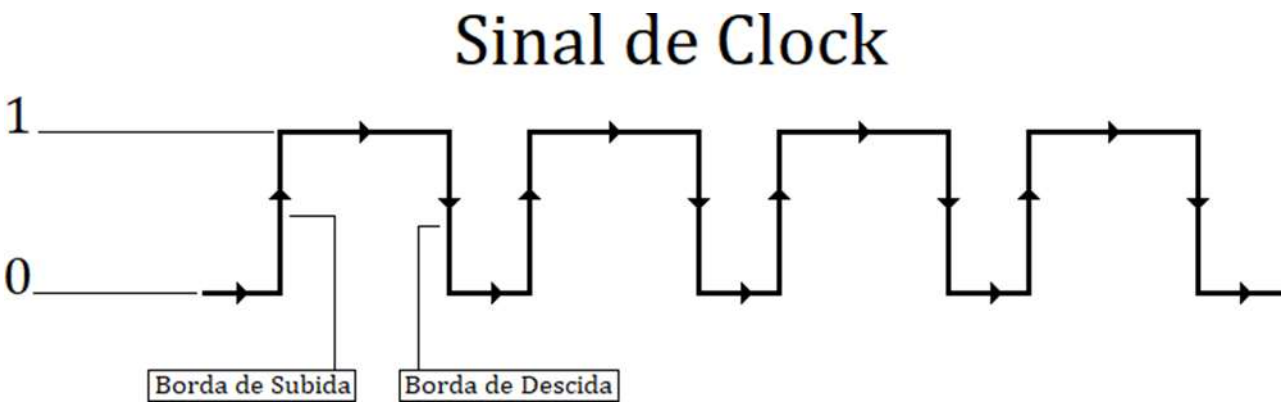
Perceba que os *latches* com portas **NAND** e **NOR** são duais entre si, embora implementem as **mesmas** funções de memória.

Como já vimos, os circuitos digitais sequenciais podem operar, basicamente, em **dois modos**: **modo assíncrono** e **modo síncrono**.

No **modo assíncrono** as saídas dos circuitos lógicos sequenciais podem mudar de estado **a qualquer momento**, bastando que para isto as suas entradas mudem de estado.

Já no **modo síncrono**, as saídas somente irão mudar de estado após uma **"autorização"**, dada por um **sinal síncrono**, chamado comumente de *clock*. Quando se impõe um *clock* a um *latch* se tem então um ***flip-flop***. Veremos os tipos de ***flip-flop*** a seguir.Mas antes, vamos definir o que é o *clock*.

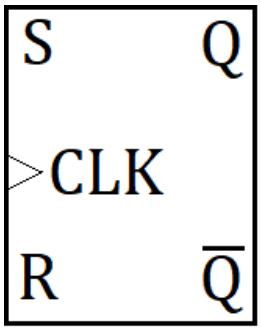
O sinal de *clock* é normalmente um **trem de pulsos retangulares**, sendo distribuído **igualmente** para todo o circuito digital. As transições de estado ocorrem somente após uma transição do sinal do *clock* (*as conhecidas bordas de subida ou bordas de descida*).



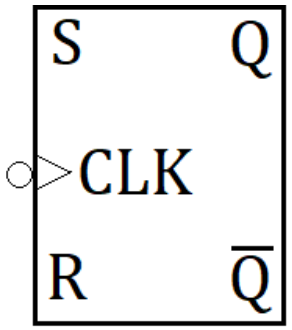
Os ***flip-flops*** podem operar com *clocks* em **borda de subida** ou **borda de descida**, dependendo basicamente de suas especificações internas. Nos blocos de ***flip-flops*** dos diagramas, podemos perceber que os ***flip-flops*** que comutam em **borda de descida** possuem um **pequeno círculo junto à entrada de *clock***. Os que comutam em **borda de subida** não possuem tal círculo.

- **Flip-Flop Tipo SR (Set-Reset)**

Quando se inclui o *clock* como variável de decisão para a mudança de estado, o *latch* com portas **NOR** que vimos há pouco passa a ser chamado de ***flip-flop SR*** (**S**et-**R**eset). Seu funcionamento é idêntico ao *latch* com portas **NOR**, porém seus blocos de implementação são os que seguem.



Flip-Flop SR
Disparado com Borda de Subida (ou borda positiva)



Flip-Flop SR
Disparado com Borda de Descida (ou borda negativa)

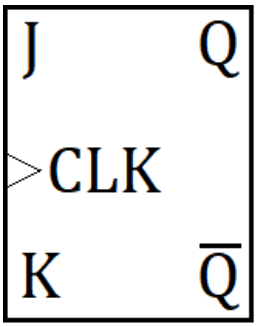
A tabela de comutação destes ***flip-flops*** fica sendo a seguinte:

Flip-Flop SR com Borda Positiva				Flip-Flop SR com Borda Negativa			
S	**R**	**CLK**	**Q**	**S**	**R**	**CLK**	**Q**
0	0	↑	$Q = Q_{ANTERIOR}$ (não muda)	0	0	↓	$Q = Q_{ANTERIOR}$ (não muda)
0	1	↑	Q = 0	0	1	↓	Q = 0
1	0	↑	Q = 1	1	0	↓	Q = 1
1	1	↑	$Q = Q_{INDEFINIDO}$ (não utilizado)	1	1	↓	$Q = Q_{INDEFINIDO}$ (não utilizado)

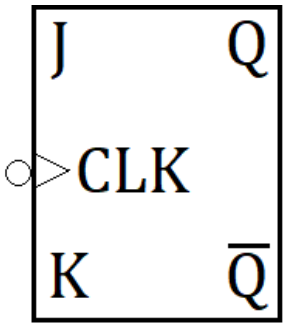
- **Flip-Flop Tipo JK**

Outro tipo de ***flip-flop*** bastante utilizado é o ***flip-flop JK*** (**J**ump-**K**ill). Ele funciona de modo bastante semelhante ao ***flip-flop SR***, a **única diferença** se dá quando as **entradas são ambas iguais a 1**. Em vez de ir para um **estado indefinido**, o ***flip-flop JK*** **inverte o sinal pré- existente** (*ou seja, se havia 1 na saída, ele vira 0. E vice-versa*).

Seus blocos de implementação são os que seguem.



Flip-Flop JK
Disparado com Borda de Subida (ou borda positiva)



Flip-Flop JK
Disparado com Borda de Descida (ou borda negativa

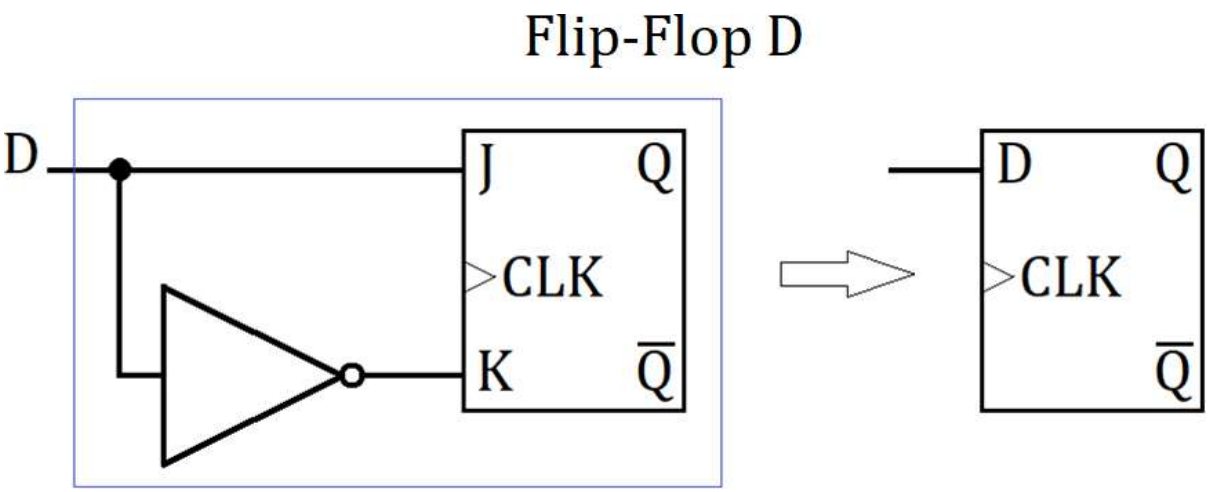
A tabela de comutação destes ***flip-flops*** fica sendo a seguinte:

FLIP-FLOP JK COM BORDA POSITIVA				FLIP-FLOP JK COM BORDA NEGATIVA			
J	K	CLK	Q	J	K	CLK	Q
0	0	↑	$Q = Q_{ANTERIOR}$ (não muda)	0	0	↓	$Q = Q_{ANTERIOR}$ (não muda)
0	1	↑	Q = 0	0	1	↓	Q = 0
1	0	↑	Q = 1	1	0	↓	Q = 1
1	1	↑	$Q = \sim Q_{ANTERIOR}$ (comuta)	1	1	↓	$Q = \sim Q_{ANTERIOR}$ (comuta)

Este é o ***flip-flop*** mais cobrado em provas de concursos.

- **Flip-Flop Tipo D**

Uma variação do ***flip-flop JK*** é o ***flip-flop D*** (*Data*). Este funciona como se ambas as entradas daquele fossem sempre 0 ou 1, ou seja:

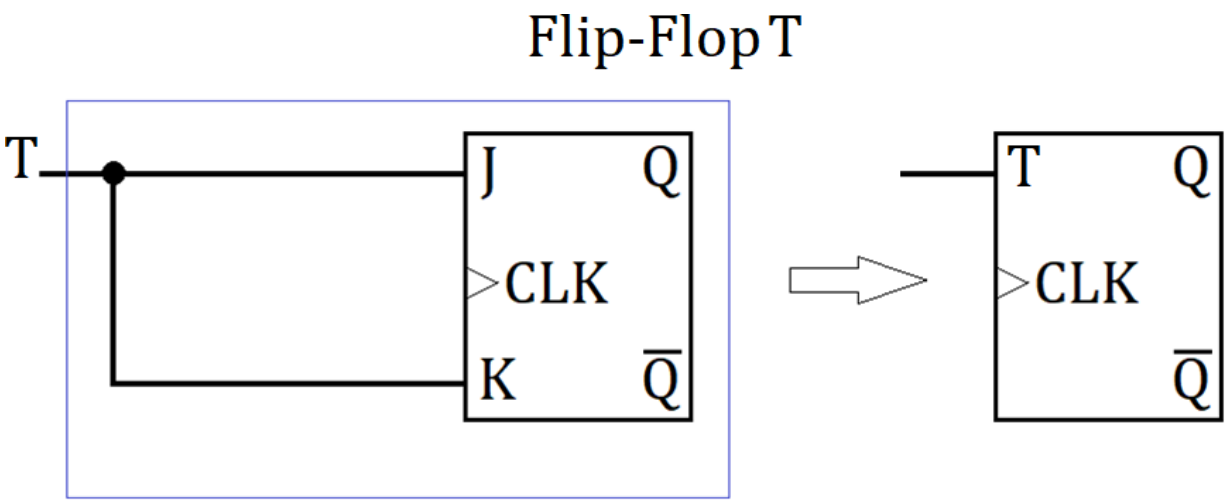


Este ***flip-flop*** gera a seguinte tabela:

Flip-Flop D com Borda Positiva			Flip-Flop D com Borda Negativa		
D	**CLK**	**Q**	**D**	**CLK**	**Q**
0	↑	Q = 0	0	↓	Q = 0
1	↑	Q = 1	1	↓	Q = 1

- **Flip-Flop Tipo T**

O último ***flip-flop*** que veremos é o ***flip-flop T*** (*Toggle*). Ele funciona como se ambas as entradas no ***flip-flop JK*** fossem **curto-circuitadas**, conforme segue:



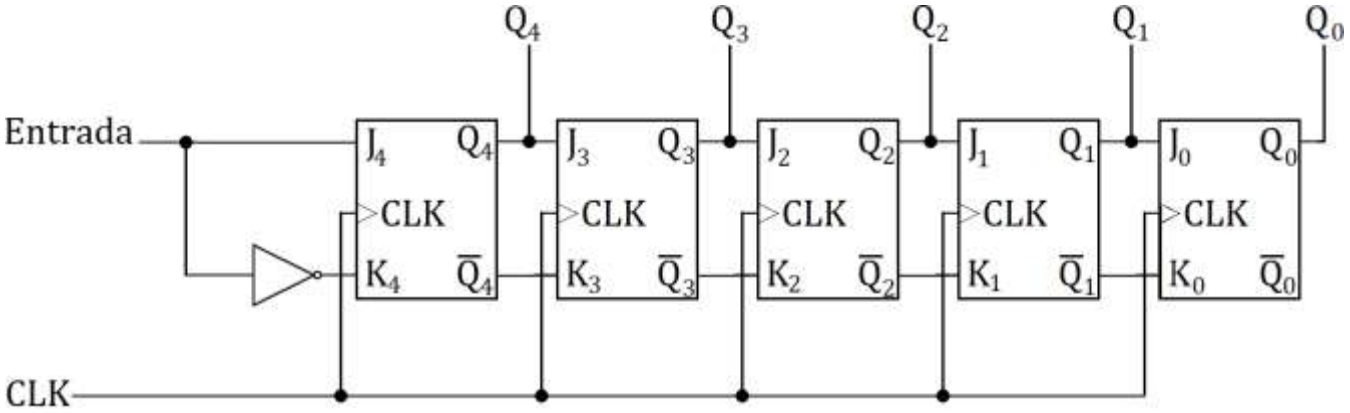
Este ***flip-flop*** gera a seguinte tabela:

Flip-Flop T com Borda Positiva			Flip-Flop T com Borda Negativa		
T	**CLK**	**Q**	**T**	**CLK**	**Q**
0	↑	$Q = Q_{ANTERIOR}$	0	↓	$Q = Q_{ANTERIOR}$
1	↑	$Q = \sim Q_{ANTERIOR}$ (comuta)	1	↓	$Q = \sim Q_{ANTERIOR}$ (comuta)

Onde o "~" denota a operação de inversão.

Registradores

Os **registradores de deslocamento** (também conhecidos como *shift-registers*) são circuitos lógicos que possuem a **capacidade de armazenar e deslocar bits de dados para o elemento de memória seguinte da cadeia**. A Figura a seguir ilustra um registrador de deslocamento que pode armazenar até 5 bits de informação.

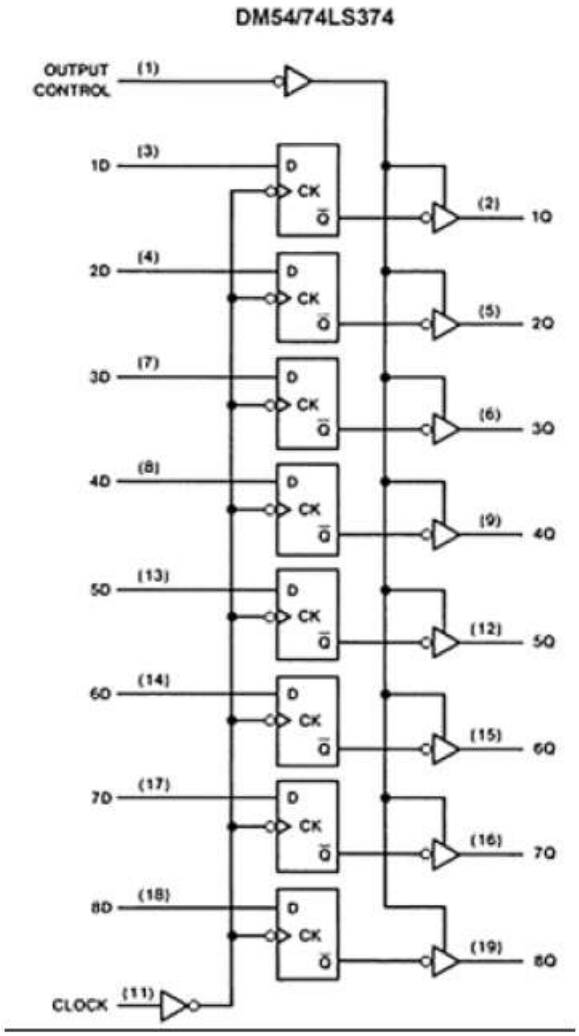


Cada *flip-flop* pode **armazenar 1 bit de informação** (*sua própria saída Q*). Caso se precise guardar informações que tenham **mais de 1 bit**, somente um flip-flop fica insuficiente. Para contornar esta insuficiência, **conecta-se uma série de flip-flops do tipo JK em série**, tal como demonstrado na Figura.

Este circuito faz com que o sinal inserido em entrada seja, **a cada borda de subida do sinal de clock**, **transferido para o flip-flop seguinte**. Pode-se dizer que este registrador de deslocamento é do tipo entrada em série (*um dado entra por vez no circuito*) e saída em paralelo (*todos os dados de saída podem ser lidos individualmente*).

Perceba que **este circuito poderia ser implementado com flip-flops do tipoD**, anulando-se a porta lógica NOT que liga a K4.

(UFMG/UFMG - 2019) Analise a figura que mostra o diagrama interno do circuito integrado 74LS374:



A alternativa que indica a função lógica CORRETA do circuito é:

A) Registrador de deslocamento com entrada paralela e saída paralela.
B) Registrador de entrada paralela e saída paralela com coletor aberto.
C) Registrador de entrada paralela e saída paralela com três estados.
D) Contador de 8 bits síncrono com saída em coletor aberto.

Comentários:

Gabarito: Alternativa C.

Não se trata de um contador, uma vez que os flip-flops são independentes entre si.

Portanto, descarta-se a alternativa D.

Não se trata de um registrador de deslocamento, uma vez que os flip-flops não estão ligados em série, e sim em paralelo. Portanto, descarta-se a alternativa A**.**

Trata-se de um registrador de entrada paralela e saída paralela, porém a saída não é com coletor aberto, uma vez que há um sinal de controle para a saída. Portanto, descarta-se a alternativa B**.**

Devido ao fato de o *buffer de saída* do circuito **integrado** possuir uma segunda entrada**, pode-se dizer que o** circuito integrado possui saída paralela com 3 estados: **1, 0 ou Z (alta impedância). Uma saída Z pode ser considerada como desconectada do circuito. Isto é útil quando se precisa que um bit específico seja lido por uma varredura feita por outro dispositivo, por exemplo**. A saída em Z é diferente da saída 0, uma vez que 0 não significa ausência de tensão, e sim apenas uma convenção para um nível lógico baixo.

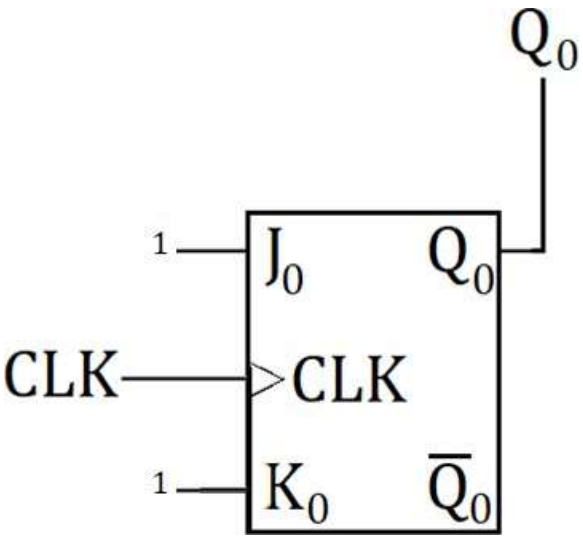
Contadores

Contadores são circuitos lógicos sequenciais capazes de realizar a contagem de determinado número de eventos. Estes circuitos lógicos podem ser do tipo **síncrono** ou **assíncrono**.

Quando **síncronos**, o sinal de *clock* é conectado a todos os *flip-flops*. Quando **assíncronos**, o sinal de *clock* é conectado somente ao primeiro *flip-flop* (*o do bit menos significativo*). Veremos nesta aula o contador de tipo síncrono.

Para que o contador síncrono conte adequadamente, apenas os *flip-flops* **que devam comutar** naquela transição de *clock* **devem ter suas entradas J e K** iguais a 1. Vejamos o passo a passo. Vamos assumir que os flip-flops todos estão inicialmente com suas saídas Q iguais a zero.

Vamos primeiro implementar um contador simples de um único bit. Este contador, óbvio, terá apenas 2 estados: 0 e 1.



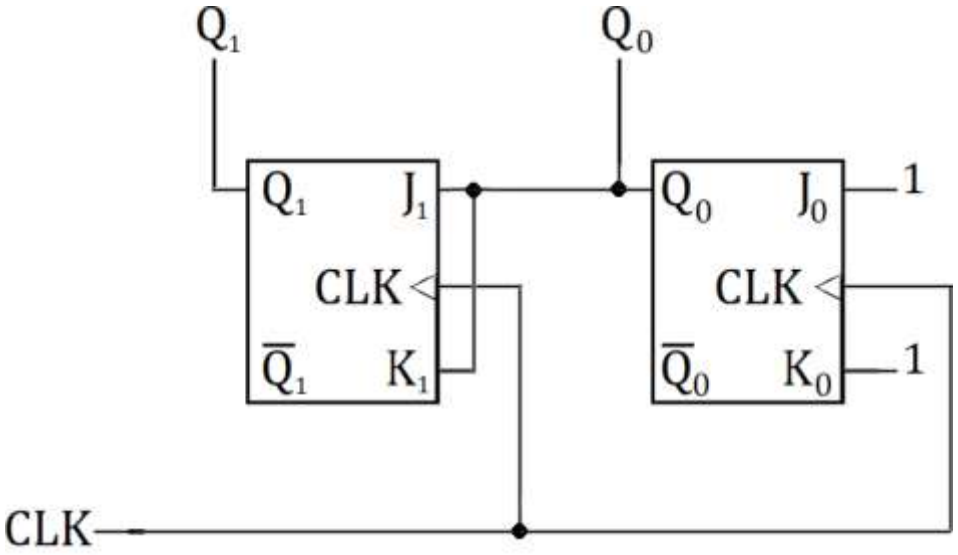
O que deve ser conectado em J_0 e K_0 para que o contador possa realizar a seguintecontagem:

$$0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \ldots?$$

Bom, da teoria dos **flip-flops JK** você irá lembrar que quando J = K = 1, a cada pulso de *clock* a saída Q deste flip-flop comuta, ou seja, inverte seu estado atual.

Bom, se queremos que **a saída seja sempre 0 e 1 alternadamente**, basta ligarmos $J_0 = K_0 = 1$ e teremos nosso contador de 1 bit!

Vamos agora aumentar este contador para 2 bits. Para isto precisaremos de 2 flip-flops JK.



E agora, o que ligar nas entradas J_1, K_1, J_0 e K_0 para termos a seguinte contagem:

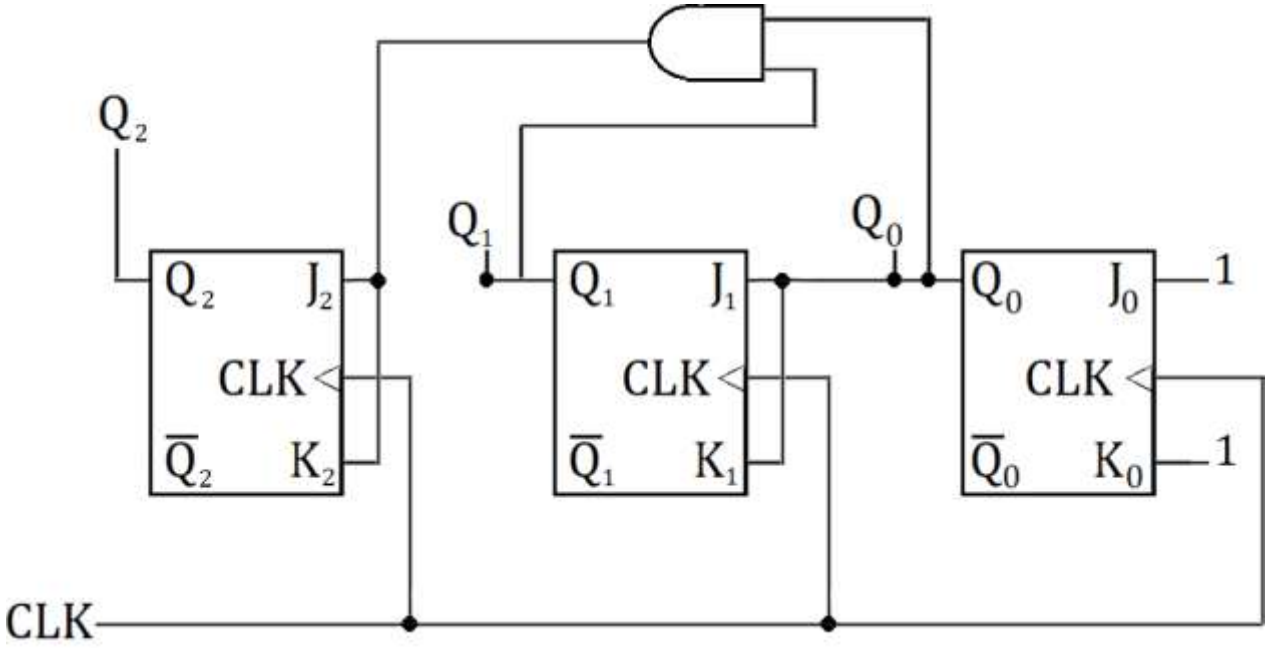
00 → 01 → 10 → 11 → 00 → 01 → 10 → 11 → 00 → 01 → 10 → 11 ...?

Como você pode perceber, **o dígito menos significativo** segue oscilando sempre entre 0 e 1, a cada pulso de *clock,* certo?

E o dígito mais significativo comuta sempre no estágio seguinte ao qual o dígito menos significativo assume o valor 1, correto?

Então podemos implementar a seguinte lógica: $J_0 = K_0 = 1$ (*mantém as entradas do contador de 1 bit*) e $J_1 = K_1 = Q_0$ (ou seja, quando Q_0 for igual a 1, ao receber o pulso de *clock*, Q_1 passa a ser 1 também).

Vamos **aumentar de novo este contador**. Agora para 3 bits. Para isto precisaremos de 3 flip-flops JK.



O que ligar nas entradas J_2, K_2, J_1, K_1, J_0 e K_0 para termos a seguinte contagem:

000 → 001 → 010 → 011 → 100 → 101 → 110 → 111 → 000 → 001 → 010 → 011 ...?

Bom, como já percebemos, **não precisaremos mexer no contador de 2 bits que já implementamos**, uma vez que a variação destes 2 bits menos significativos permanece idêntica.

Mas e com relação a J_2 e K_2? Perceba que a saída do flip-flop que contém o bit mais significativo somente irá comutar quando tanto Q_0 quanto Q_1 forem iguais a 1 e vier um pulso de *clock*.

A análise segue na mesma lógica para se implementar contadores de N bits: sempre a condição para comutação dos bits subsequentes se dará quando todos os bits anteriores forem iguais a 1.

Famílias Lógicas

Veremos nesta seção uma **breve explanação** sobre as **famílias lógicas de circuitos integrados**.

Os **circuitos integrados** digitais podem ser classificados de acordo com o principal componente eletrônico utilizado em seus circuitos internos (**TBJ** ou **MOSFET**).

Existem atualmente **6 níveis de complexidade** dos circuitos integrados:

COMPLEXIDADE	PORTAS LÓGICAS POR CIRCUITO INTEGRADO
SSI (*Short Scale Integration*)	N < 12
MSI (*Medium Scale Integration*)	12 ≤ N < 99
LSI (*Large Scale Integration*)	100 < N < 999
VLSI (*Very Large Scale Integration*)	1000 < N < 99.999
ULSI (*Ultra Large Scale Integration*)	100.000 < 999.999
GSI (*Giga Scale Integration*)	N > 1.000.000

TTL (Transistor-Transistor Logic)

Esta é a principal família de circuitos integrados digitais bipolares (**TBJ**) e teve como antecessoras a **ECL**, **HTL** e **DTL**. A família **TTL** opera com uma tensão nominal de +5 V (V_{CC} = +5 V).

O nível alto na tecnologia **TTL** se dá entre +2,0 V e +5,0 V. Já o nível baixo se dá entre 0,0 V e 0,8 V. Entre +0,8 V e +2,0 V existe uma indeterminação.

Os circuitos integrados da família **TTL** possuem alimentação pelo pino denominado **V_{CC}** e o terra pelo pino denominado **GND**.

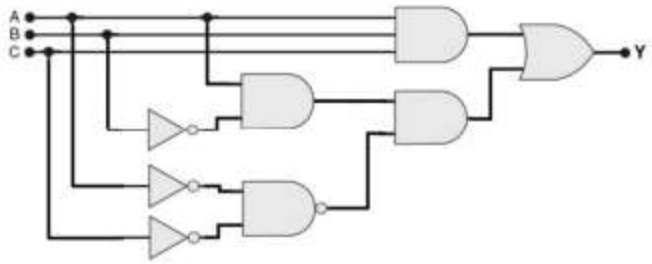
Na família **TTL**, caso se deixe uma entrada flutuante, ou seja, não se conecte nada a ela, o circuito integrado interpreta tal entrada como sendo de nível alto.

Ela é composta por várias subfamílias (ou séries), conforme a tabela a seguir apresenta.

Série	Prefixo	Exemplo
TTL	74	7404
TTL Schottky	74S	74S04
TTL Schottky de Baixa Potência	74LS	74LS04
TTL Schottky Avançado	74AS	74AS04
TTL Schottky Avançado de Baixa Potência	74ALS	74ALS04

HORA DE PRATICAR!

(CESPE/CEBRASPE - POLC AL - Perito Criminal - 2023)



Considerando-se o circuito apresentado na figura precedente e as famílias de circuitos lógicos, julgue os seguintes itens.

Para aumento de desempenho quanto à velocidade, o circuito mostrado deve ser implementado com TTL Schottky de baixa potência ao invés da família TTL padrão.

C) Certo

E) Errado

Comentários:

Gabarito: Errado.

Aproveitaremos esta questão para aprofundar os estudos a respeito do **TTL Schottky de Baixa Potência.**

A série 74LS é uma versão da 74S, que apresenta circuitos integrados com **consumo de potência mais baixo** e **com velocidade também mais baixa**. Esta série usa o transistor Schottky grampeado, com resistores de valor superior aos utilizados na série 74S.

Portanto, quando **objetivamos um aumento de desempenho quanto à velocidade**, a implementação do TTL Schottky de baixa potência não é a melhor escolha. Logo, a afirmativa feita na questão está **errada**.

CMOS (Complementary Metal-Oxide Semiconductor)

Quando a fabricação é feita a partir da escala **LSI**, a tecnologia **CMOS** é a predominante devido à sua simplicidade e a forma compacta com que se consegue construir os circuitos integrados com esta tecnologia. A família **CMOS** operar numa faixa de tensão que vai de +3 Va +18 V, embora a mais utilizada seja a de +5 V.

O nível alto na tecnologia **CMOS** mais empregada se dá entre +3,5 V e +5,0 V. Já o nível baixo se dá entre 0,0 V e + 1,5 V. Entre +1,5 V e +3,5 V existe uma indeterminação.

Os circuitos integrados na família **CMOS** são alimentados pelo pino denominado $\mathbf{V_{DD}}$. Já o terra é conectado ao pino $\mathbf{V_{SS}}$.

Na família **CMOS**, caso se deixe uma entrada flutuante, ou seja, não se conecte nada a ela, o circuito integrado pode superaquecer e se danificar. Desta forma, sempre se deve conectar todas as entradas de um circuito integrado CMOS a um nível lógico baixo ou alto.

Assim como a família **TTL**, a família **CMOS** também é subdivida em várias séries:

SÉRIE	PREFIXO	EXEMPLO
CMOS com porta de metal	40	4001
Porta de metal compatível pino a pino com TTL	74C	74C04
Porta de metal compatível pino a pino com TTL, alta velocidade	74HC	74HC04
Porta de silício, alta velocidade, compatível pino a pino e eletricamente com TTL	74HCT	74HCT04
CMOS de altíssimo desempenho, não compatível com TTL	74AC	74AC04
CMOS de altíssimo desempenho, não compatível pino a pino com TTL	74ACT	74ACT04

QUESTÕES COMENTADAS

1. (UFCG / UFCG - 2019)

Um técnico de laboratório precisa montar um circuito digital de 3 entradas e 1 saída que obedeça a Tabela da Verdade apresentada na tabela abaixo.

Na tabela, A, B e C representam as entradas do circuito digital e Y representa a saída do circuito digital.

É correto afirmar que a expressão lógica de Y é:

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

a) $Y=\overline{C}+A\cdot B$

b) $Y=\overline{A\cdot B}+C$

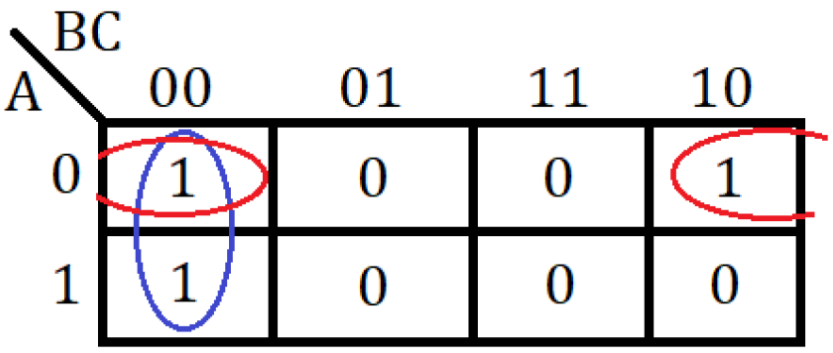
c) $Y=\overline{C}\cdot(\overline{A+B})$

d) $Y=\overline{C+A\cdot B}$

e) $Y=\overline{C}+\overline{A\cdot B}$

Comentários:

Montando o **Mapa de Karnaugh** da **tabela verdade** acima.



Podemos então simplificar a tabela verdade da seguinte maneira:

$$Y=\overline{B}\cdot\overline{C}+\overline{A}\cdot\overline{C}$$

$$Y=\overline{C}\cdot(\overline{A}+\overline{B}) \quad \rightarrow \quad \acute{Y}=\overline{C}\cdot(\acute{\overline{A}}+\overline{B})$$

$$Y=\overline{C+(A\cdot B)}$$

Para se chegar ao resultado conforme o que é apresentado nas alternativas, foi necessário **negar duas vezes** a expressão, de ambos os lados, pois, ao se fazer isto, **não se altera o resultado da expressão**, apenas a **maneira de representação**.

Gabarito: Letra D.

2. (VUNESP/TJM-SP – 2021)

Considere a seguinte tabela-verdade, com três entradas (A, B, C) e uma saída S.

A	B	C	S
0	0	0	**1**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

A saída ***S*** é corretamente expressa em:

Alternativas

A) $A.B.C$

B) $\overline{A}.\overline{B}.\overline{C}$

C) $A \oplus B \oplus C$

D) $(\overline{A}.\overline{B}.\overline{C}) + (A.B.C)$

E) $(\overline{A} + \overline{B} + \overline{C}) \bullet (A + B + C)$

Comentários:

Gabarito: Letra D.

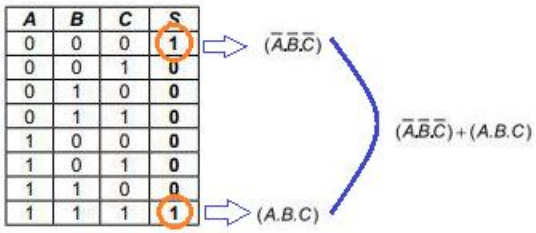
Denominamos de **Mintermo** (ou minitermo) é o termo **produto associado à cada linha da tabela verdade**, no qual todas as variáveis de entrada estão presentes.

Se substituirmos os valores das variáveis associadas (valores da tabela verdade), para um determinado mintermo, **obteremos 1** (saída verdadeira).

Porém, **se substituirmos nesse mesmo mintermo quaisquer outras combinações de valores**, **obteremos 0**.

Dessa forma, **se quisermos encontrar a equação para uma função a partir de sua tabela verdade**, **basta montarmos um OU entre os mintermos associados aos 1s da função.**

Através deste conceito podemos resolver esta questão:



Logo, **a saída é corretamente expressa por:**

$$S = (\bar{A}.\bar{B}.\bar{C})+(A.B.C)$$

Gabarito: Letra D.

3. (CS-UFG / UFG - 2019)

Tratando-se de portas lógicas e álgebra booleana, a tabela abaixo diz respeito à operação

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

a) $X = \overline{A + B}$

b) $X = \overline{A \oplus B}$

c) $X = A + B$

d) $X = A \oplus B$

Comentários:

Verifica-se que a resposta da **tabela verdade** (variável X) apresenta valores verdadeiros (ou seja, iguais a um) somente quando as entradas (A e B) são ou ambas falsas (iguais a zero) ou ambas verdadeiras.

Tal condição é o inverso (ou a negação) da porta **OU-EXCLUSIVO** (ou **XOR**). Portanto, a alternativa correta é a letra B, por esta ser a representação de uma porta **OU-EXCLUSIVO- NEGADA** (ou **XNOR**).

Gabarito: Letra B.

4. (CESPE/ FUB - Técnico em Telecomunicações/ 2015)

A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

A tabela verdade apresentada acima representa o comportamento de um circuito combinacional com entradas A, B e C e saída S.

Com relação a essa tabela, julgue o próximo item.

Na forma soma de produtos, a expressão booliana mínima para S é dada por $S = \bar{A}\bar{B}\bar{C} + A$:

A) Certo.
B) Errado.

Comentários:

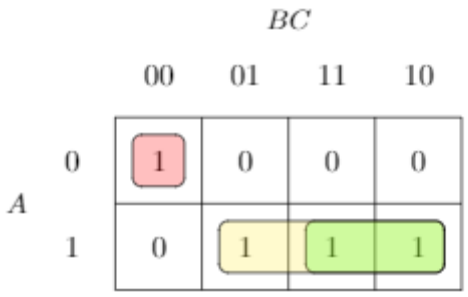
Gabarito: Errado.

Na questão para chegar a expressão booleana mínima para S representada pela tabela verdade, primeiramente encontramos a **expressão na forma canônica disjuntiva** (soma de mintermos), que é a forma mais utilizada para encontrar a expressão lógica a partir da tabela verdade.

A forma canônica disjuntiva é obtida somando os mintermos nos quais a saída é igual a 1:

$$S = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C$$

Em seguida, utilizamos o **diagrama de Karnaugh** para simplificação:



A partir do mapa, chegamos a:

$$S = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot B + A \cdot C$$

Outra maneira de encontrarmos a expressão simplificada é:

$$S = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot B\left(C + \overline{C}\right) + A \cdot \overline{B} \cdot C \quad (*)$$

$$S = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot B + A \cdot \overline{B} \cdot C$$

$$S = \overline{A} \cdot \overline{B} \cdot \overline{C} + A\left(B + \overline{B} \cdot C\right) \quad (**)$$

$$S = \overline{A} \cdot \overline{B} \cdot \overline{C} + A(B + C)$$

$$S = \overline{A} \cdot \overline{B} \cdot \overline{C} + AB + AC$$

Logo, a expressão booleana mínima para S é dada por $\boldsymbol{S = \overline{A}\overline{B}\overline{C} + AB + AC}$, e a afirmativa feita está **ERRADA.**

5. (CS-UFG / UFG - 2019)

A representação do número decimal 513 em octal é:

a) 201.

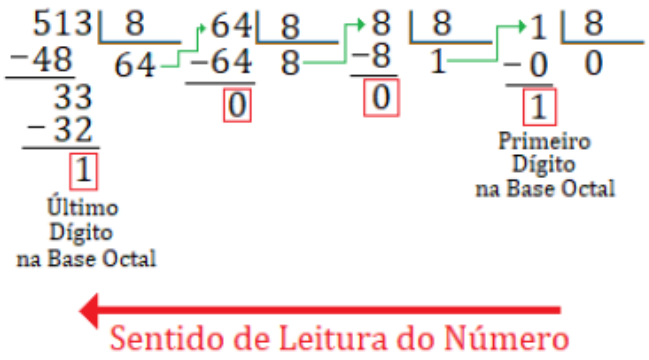
b) 0101 0001 0011.

c) 1001.

d) 0001 0000 0001.

Comentários:

Para transformar um número de uma base X para uma base Y, basta fazer divisões sucessivas do número na base X pela base Y e ir preservando o valor do resto da divisão, no modo demonstrado abaixo.



Portanto, o número 513 na base 10 equivale ao número 1001 na base octal.

Gabarito: Letra C.

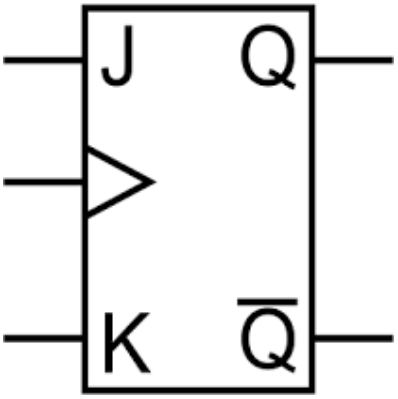
6. (CS-UFG / UFG - 2019)

No Flip-Flop tipo J-K, quando ocorre um pulso de *clock* e as entradas são J = 1 e K = 0, a saída Q

a) é 0.
b) é 1.
c) mantém o valor da saída anterior.
d) alterna o valor da saída anterior.

Comentários:

O *flip-flop* JK é um elemento de **circuitos digitais** cuja representação em circuitos é aseguinte:



Ele funciona de acordo com a seguinte **tabela verdade**:

Flip-Flop JK com Borda Positiva				Flip-Flop JK com Borda Negativa			
J	K	CLK	Q	J	K	CLK	Q
0	0	?	$Q = Q_{ANTERIOR}$ (não muda)	0	0	?	$Q = Q_{ANTERIOR}$ (não muda)
0	1	?	Q = 0	0	1	?	Q = 0
1	0	?	Q = 1	1	0	?	Q = 1
1	1	?	$Q = \sim Q_{ANTERIOR}$ (comuta)	1	1	?	$Q = \sim Q_{ANTERIOR}$ (comuta)

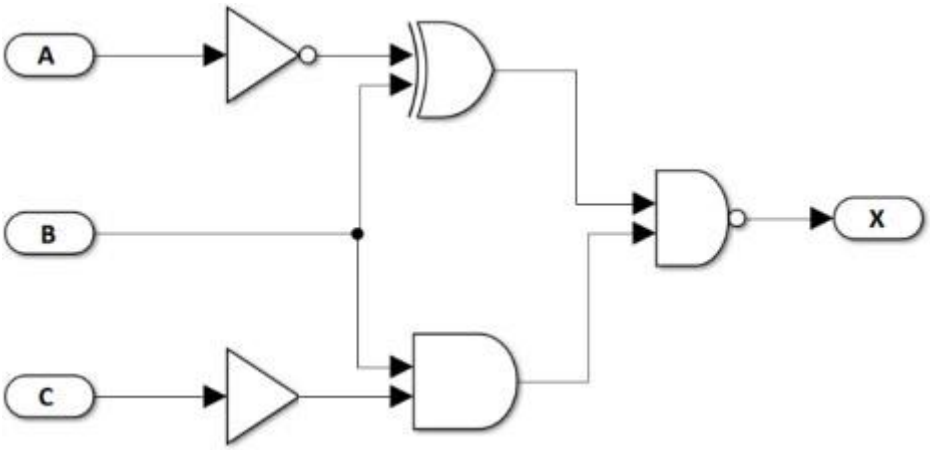
A cada pulso de *clock* o *flip-flop* verifica as entradas J e K e realiza a função descrita na sua **tabela verdade**.

Portanto, na descrição da questão, quando J = 1 e K = 0, o flip-flop JK apresenta comoresultado **Q = 1.**

Gabarito: Letra B.

7. (CS-UFG / UFG - 2019)

Considere o circuito combinacional a seguir.



A expressão para a saída X é:

a) $X=\overline{(\overline{A}\oplus B)\cdot(B\cdot C)}$.

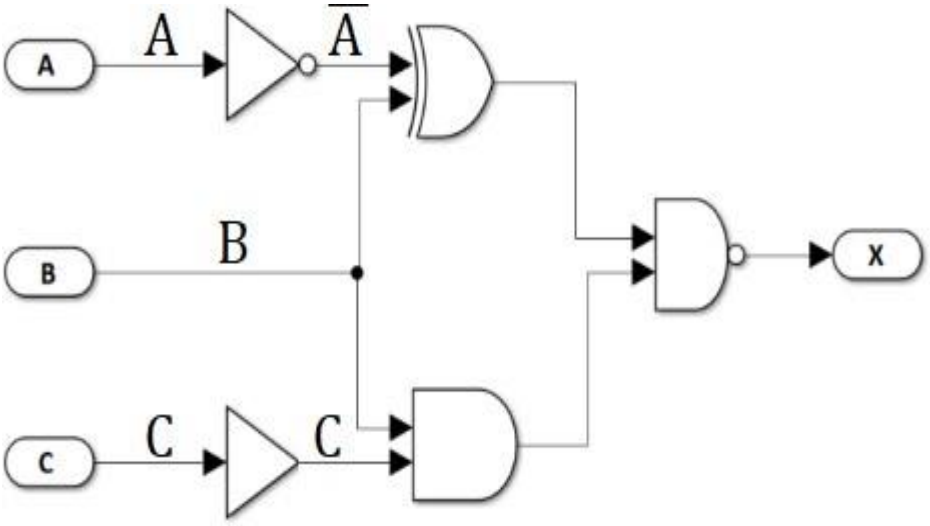
b) $X=\overline{(\overline{A}+B)\cdot(B\cdot C)}$.

c) $X=\overline{(\overline{A}\oplus B)\cdot(B\cdot \overline{C})}$.

d) $X=\overline{(\overline{A}+B)\cdot(B\cdot \overline{C})}$.

Comentários:

Vamos redesenhar o circuito e ir montando a resposta passo a passo.

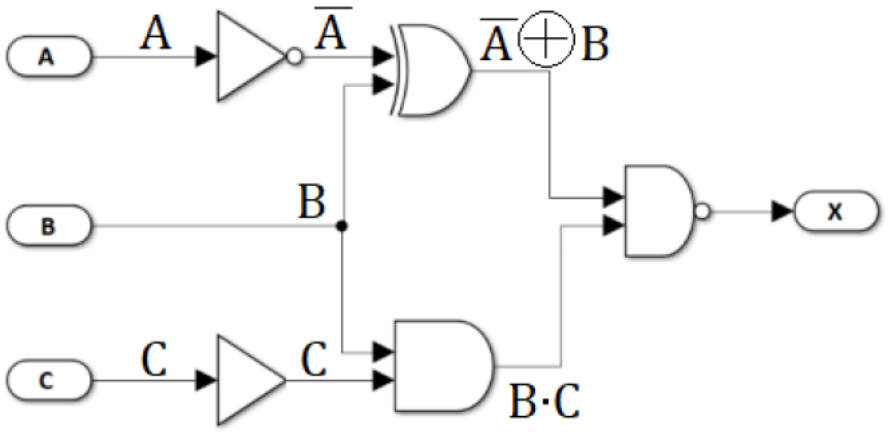


Primeiramente, o sinal A adentra o **inversor lógico** (**porta NOT**), gerando como saída $\overline{A}$.

Em segundo lugar, o sinal B entra junto com $\overline{A}$ na **porta XOR** (**ou-exclusivo**).

E, em terceiro lugar, o sinal C entra em um bloco que é apenas um buffer, ou seja, não produz efeito nenhum, so**mente cria uma interface de separação** entre o sinal C de entrada e o sinal C de saída.

Próximo passo:



Incluiu-se o sinal $\overline{A} \oplus B$ na saída da porta **XOR** e o sinal $B \cdot C$ na saída da porta **AND**.

Estes dois sinais fazem a entrada da porta **NAND** que vem em seguida.

A seguir se apresenta o **resultado final** X do circuito digital.

$$X = \overline{(\overline{A} \oplus B) \cdot B \cdot C}$$

Gabarito: Letra A.

8. (CS-UFG / UFG - 2019)

Qual das alternativas abaixo diz respeito apenas à família lógica bipolar?

a) TTL, VHDL, RTL e DTL.
b) TTL, HTL, I2C e RTL.
c) TTL, VHDL, I2C e RTL.
d) TTL, HTL, RTL e DTL.

Comentários:

Podemos classificar os transistores em **duas grandes famílias**: **Bipolar** e **MOS**.

Na **família bipolar** temos pelo menos **7 subfamílias mais conhecidas**, por assim dizer: **DTL** (*Lógica de Diodos e Transistores*), **DCTL** (*Lógica de Transistores Diretamente Acoplados*), **RTL** (*Lógica de Transistores e Resistores*), **RCTL** (*RTL com capacitores*), **HTL** (*Lógica de Alto Limiar*), **TTL** (*Lógica de Transistor-Transistor*) e **ECL** (*Lógica de Emissores Acoplados*);

Na **família MOS** temos **3 subfamílias mais conhecidas**: **PMOS** (MOSFET canal P), **NMOS** (MOSFET canal N) e **CMOS** (*Lógica MOS Complementar*).Analisando as alternativas:

Alternativa A: **Errada**. Pois **VHDL não faz parte** da família bipolar. VHDL é uma linguagem de descrição de hardware, muito utilizada em FPGAs.

Alternativa B: **Errada**. Pois I2C **não faz** parte da família bipolar. O I2C é um barramento de comunicação serial de computadores.

Alternativa C: **Errada**. Errada pois **VHDL** e **I2C não fazem parte** da família bipolar.

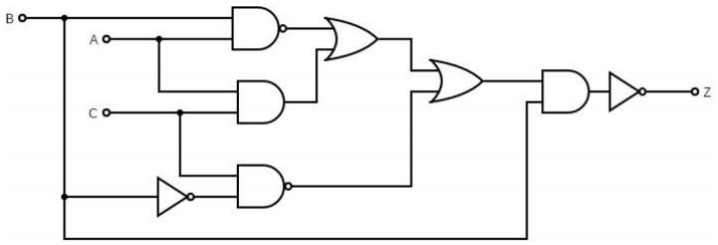
Alternativa D: **Correta**.

Portanto, o gabarito da questão é a Letra D.

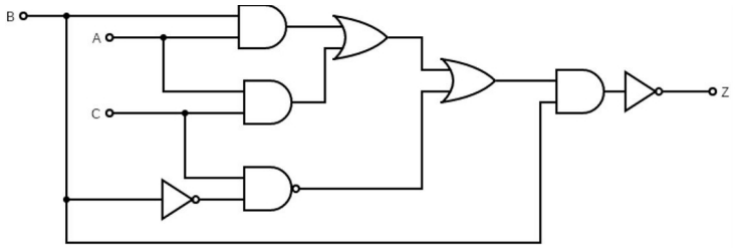
9. (FUNDEP/Prefeitura de Uberlândia - 2019)

Considerando quatro variáveis binárias A, B, C e Z, assinale a alternativa que apresenta o circuito lógico que pode ser utilizado para implementar a expressão: $Z=\overline{\left(\overline{AB}+CA+\overline{\overline{B}\,C}\right)B}$.

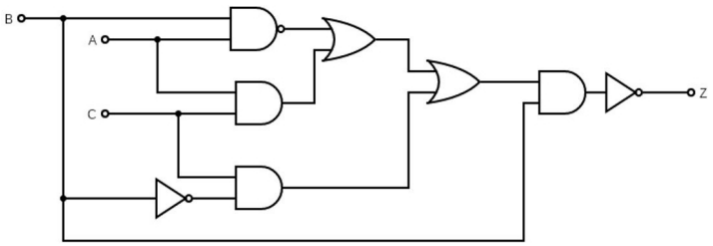
a)



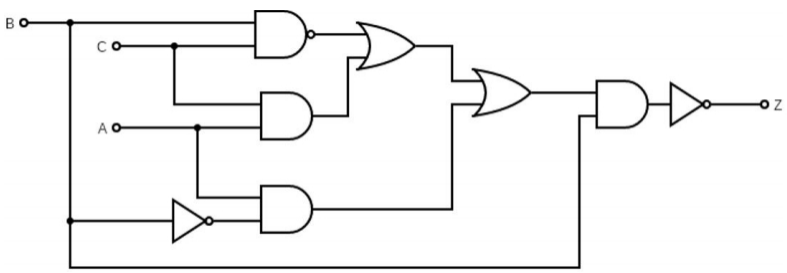
b)



c)



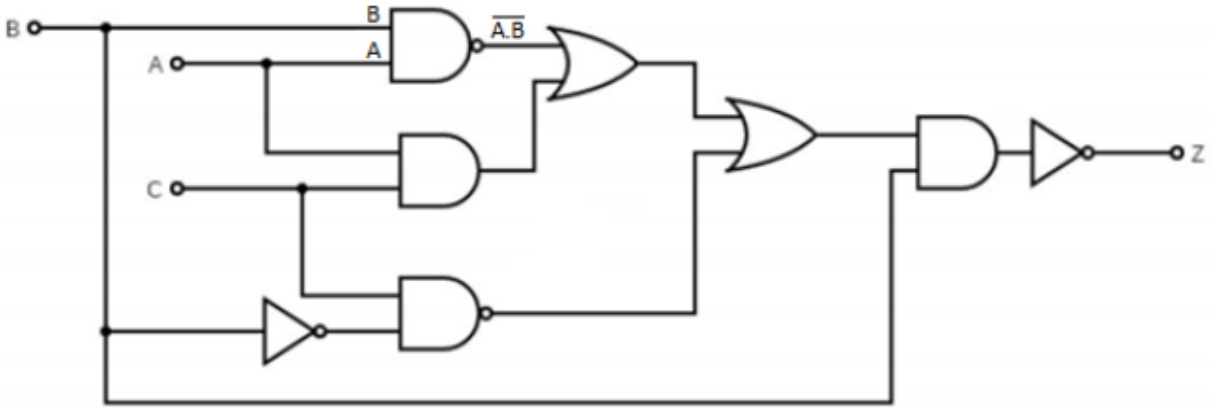
d)



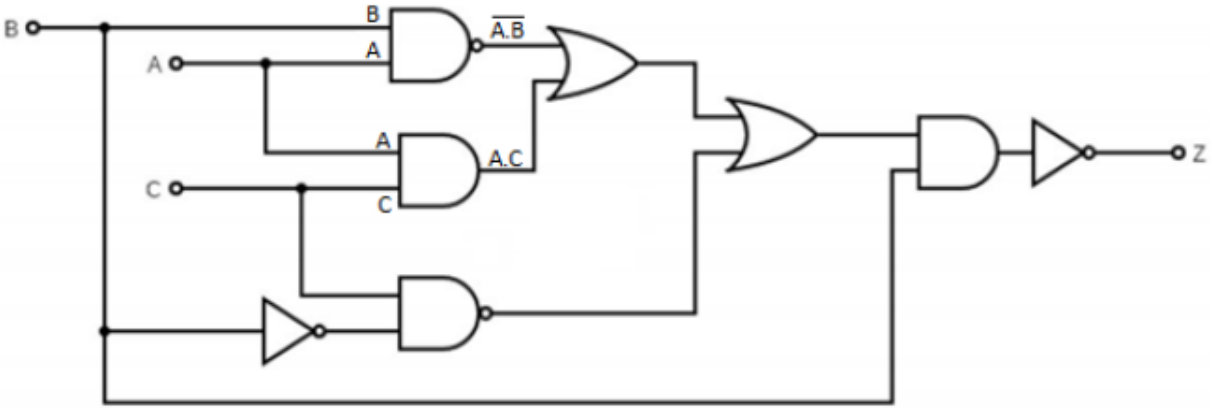
Para resolver esta questão é necessário **testar** os **circuitos lógicos** das alternativas no sentido de verificar se a **expressão lógica** que ela gera coincide com a fornecida pelo cabeçalho.

Comecemos pelo **circuito lógico** da alternativa A.

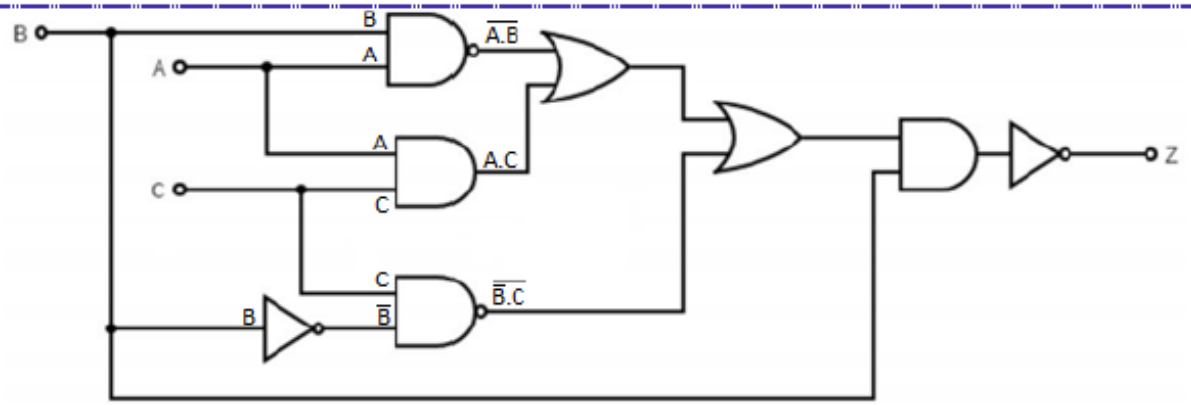
Nossa análise inicia da esquerda para a direita e de cima para baixo. A primeira **porta lógica** é uma **NAND** (ou um seja, uma função "E" negada). Nela adentram os **sinais** "B" e "A". O **resultado lógico** desta combinação é $\overline{A.B}$ (ou seja, a negação da associação entre os sinais A e B). Vamos escrever a expressão.



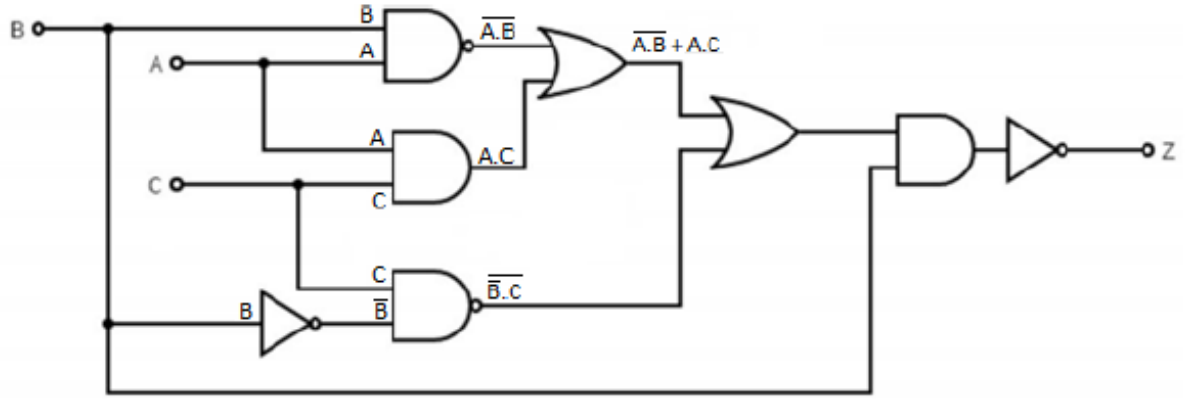
A seguir analisamos a **porta lógica** logo abaixo. Trata-se de uma porta **AND** (função "E"). Nela adentram os sinais "A" e "C". O **resultado lógico** desta combinação é $A.C$. Vamos escrever esta expressão.



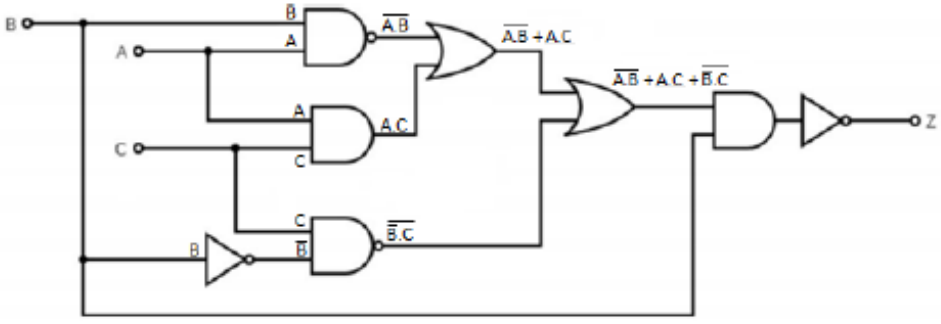
A seguir analisamos a **porta lógica** abaixo. Trata-se de uma porta **NAND**. Nela adentram os sinais "C" e "B" negado. O **resultado lógico** desta combinação é $\overline{B.C}$. Vamos escrever esta expressão.



Subindo novamente e indo para a próxima **porta lógica**. Trata-se de uma porta **OR** (função "OU"). Nela adentram os sinais $\overline{A.B}$ e $A.C$. O **resultado lógico** desta combinação é $\overline{A.B}+A.C$. Vamos escrever esta expressão.



A seguir, a próxima **porta lógica** também é uma **OR**, com sinais de entrada dados por $\overline{A.B}+A.C$ e por $\overline{B.C}$. O **resultado lógico** desta expressão é $\overline{A.B}+A.C+\overline{B.C}$. Vamos escrever esta expressão.



Por último, o sinal $\overline{A.B}+A.C+\overline{B.C}$ é combinado com o sinal "B" em uma **porta lógica AND** associada a uma **porta lógica NOT**, o que gera uma **porta lógica NAND**, cujo resultado é a expressão Z que a questão busca. Sendo assim, tem-se que $Z=\overline{(\overline{A.B}+A.C+\overline{B.C}).B}$, que bate com a expressão do cabeçalho e é o gabarito da questão. Portanto, a **alternativa correta é a letra A**.

10.(FUNDEP/Prefeitura de Uberlândia - 2019)

Em relação aos sistemas digitais, a conversão decimal binário é de fundamental importância. Assinale a alternativa que apresenta o equivalente binário ao número decimal 35.

a) 100001

b) 100111

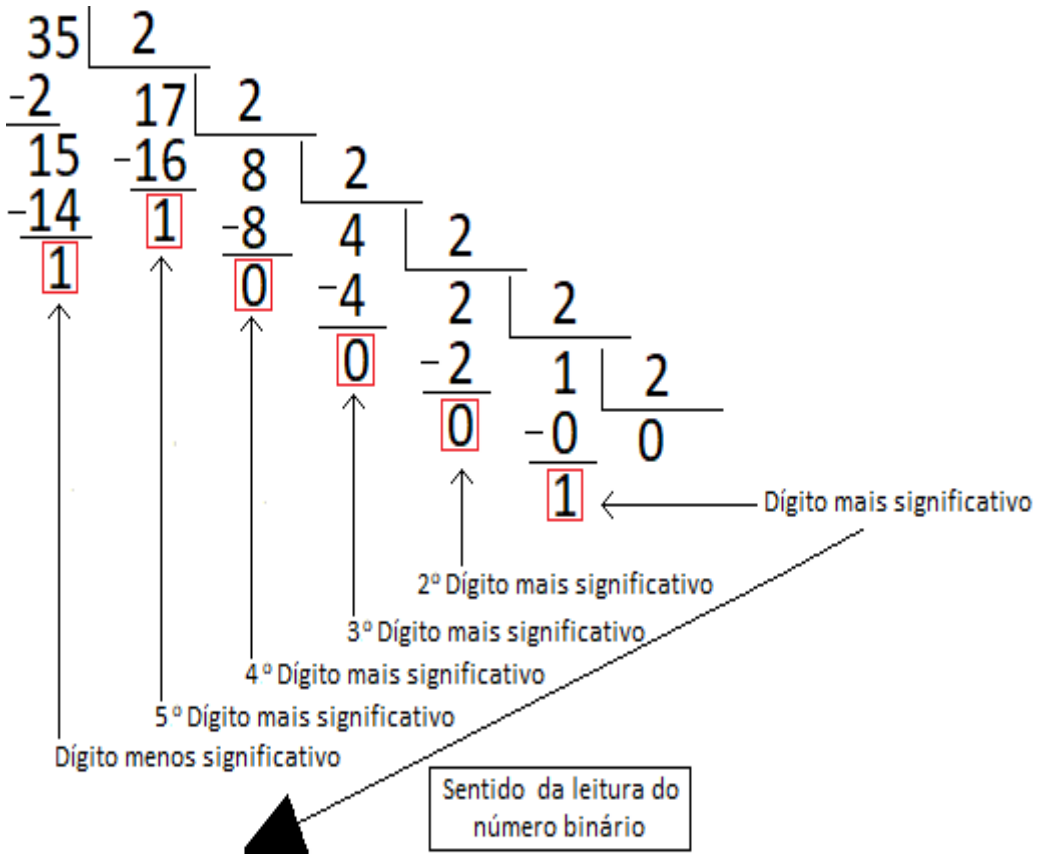
c) 110001

d) 100011

Comentários:

Para se fazer a conversão de um **número decimal** para um **número binário** é necessáriofazer uma **sucessão de divisões por 2.** O resultado é então lido ao contrário, conforme

segue.



Portanto, o **número 35 em decimal é representado em binário pelo número 100011**.

Assim, a **alternativa correta é a letra D**.

11. (FUNDEP/Prefeitura de Uberlândia - 2019)

O sistema de base 16, ou hexadecimal, utiliza 16 símbolos para a representação numérica. Assinale a alternativa que apresenta a representação decimal para o número hexadecimal 5D3B.

a) 32

b) 23866

c) 23867

d) 512

Comentários:

A conversão de um **número hexadecimal** para **decimal** pode ser da seguinte maneira.

$$(5D3B)_{16}=(5\times16^3+D\times16^2+3\times16^1+B\times16^0)_{10}$$

Ok, mas multiplicar "D" e "B"??? Como assim?? Bom, você lembra que os números hexadecimais são [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F], certo? E que "A" em hexadecimal equivale a 10 em decimal, "B" em hexadecimal equivale a vale 11 em decimal, e assim pordiante até "F", que em decimal equivale a 15.

Sendo assim, substituímos os valores das letras hexadecimais pelos seus correspondentes decimais:

$$(5D3B)_{16}=(5\times16^3+13\times16^2+3\times16^1+11\times16^0)_{10}$$

E agora efetuamos as multiplicações e somas, obtendo $(5D3B)_{16}=(23867)_{10}$. Ou seja, **alternativa correta é a letra C**.

12. (FGV/Prefeitura de Salvador - 2019)

Considere a tabela verdade de um dado circuito digital, a seguir.

a	b	c	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

A saída Y desse circuito é:

a) $Y = bc + a\bar{c} + \bar{a}c + \bar{a}b$

b) $Y = b\bar{c} + a\bar{c} + \bar{a}c + \bar{a}b$

c) $Y = b\bar{c} + a\bar{c} + \bar{a}c + ac$

d) $Y = b\bar{c} + a\bar{c} + bc + \bar{a}b$

e) $Y = b\bar{c} + a\bar{c} + \bar{a}\bar{c} + \bar{a}b$

Comentários:

Desenhando o **Mapa de Karnaugh** desta tabela verdade de **3 bits** e associando os termos que podem ser associados entre si. Lembrando que os **termos são agrupáveis sempre em potências de 2** (ou seja, 2^0, 2^1, 2^2, ...) e **sempre com vizinhos na horizontal ou vertical**.

a \ bc	00	01	11	10
0	0	1	1	1
1	1	0	0	1

E a expressão que ele representa fica sendo: $Y = \overbrace{\bar{a}\,c}^{Vermelho} + \underbrace{\bar{a}\,b}_{Verde} + \overbrace{b\,\bar{c}}^{Azul} + \underbrace{a\,\bar{c}}_{Roxo}$. Logo, a alternativa correta é a **letra B**.

13. (IBFC/Prefeitura de Cabo de Santo Agostinho - 2019)

A capacidade de armazenar informações binárias por um período de tempo é uma função imprescindível para os sistemas digitais. Sobre o dispositivo responsável por tal armazenamento em circuitos digitais, assinale a alternativa correta.

a) Portas lógicas
b) Comparador
c) Multiplexador
d) Registrador

Comentários:

Analisemos as alternativas.

Alternativa A: **Errado**. Portas lógicas sozinhas somente implementam suas funções intrínsecas. Então está **errado**.

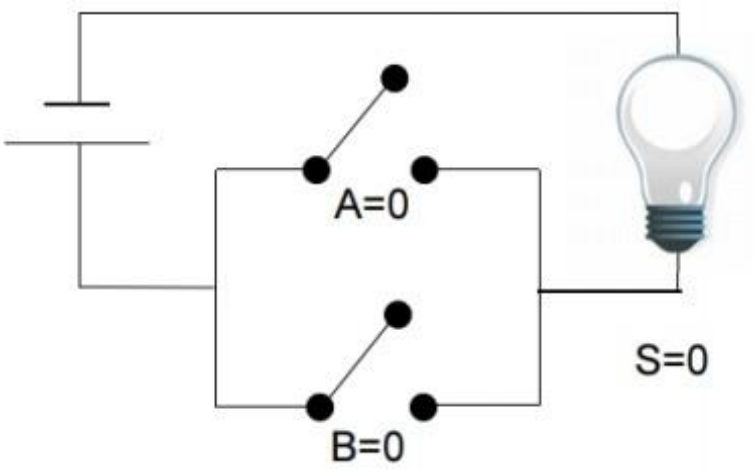
Alternativa B: **Errado**. O comparador simplesmente compara dois valores e emite um sinalque representa maior, igual ou menor. Portanto está **errado**.

Alternativa C: **Errado**. O multiplexador combina diversos sinais em um único canal. Servepara mandar diversos dados através de uma única saída. Então está **errado**.

Alternativa D: **Correto**. O registrador é um circuito formado por **flip-flops**. É nestes **flip-flops**

14. (IBFC/Prefeitura de Cabo de Santo Agostinho - 2019)

A lâmpada ilustrada na figura abaixo deve ser acesa e, para executar essa função, utiliza-se a porta lógica OU. Para tanto, tem-se as chaves idênticas A e B que se encontram inicialmente na posição aberta (binário 0). Sabendo que a posição fechada é representada pelo binário 1, analise as afirmativas abaixo e dê valores de Verdadeiro (V) ou Falso (F).



( ) Se A = 0 e B = 1 a lâmpada irá

acender.

( ) Se A = 1 e B = 0 a lâmpada irá

apagar.

( ) Se A = 1 e B = 1 a lâmpada não irá acender, pois ocorrerá um curto-circuito.

Assinale a alternativa que apresenta a sequência correta de cima para baixo.

a) F, F, F
b) V, F, F
c) F, V, V
d) V, F, V

Comentários:

Assumindo que a fonte possui capacidade para acender a lâmpada, nota-se que na condição mostrada na figura acima (ambas as chaves abertas, ou seja A = 0 e B = 0), a lâmpada obviamente **não acende**, uma vez que não forma um circuito fechado.

Se a **chave B fechar** (ou seja, mudar do estado 0 para o estado 1), com a chave A permanecendo aberta (estado 0), então a lâmpada se acenderá. Portanto, a primeira afirmativa é verdadeira.

Se a **chave A fechar** (ou seja, mudar do estado 0 para o estado 1), com a chave B permanecendo aberta, então a lâmpada também se acenderá. Portanto, a segunda afirmativa é falsa.

Se **ambas as chaves fecharem** (ou seja, mudarem do estado 0 para o estado 1), então a lâmpada também se acenderá. Portanto, a terceira afirmativa também é falsa.

Ou seja, o circuito comporta-se como uma **porta lógica OU**, com a saída da porta sendo representada pelo acendimento (estado 1) ou não acendimento da lâmpada (estado 0).

Portanto, o gabarito é a letra B.

15. (IBFC/Prefeitura de Cabo de Santo Agostinho - 2019)

As mais diversas áreas têm apresentado avanços tecnológicos significantes nas últimas décadas e muito disso se faz devido aos sistemas digitais. Sobre as vantagens dos sistemas digitais em comparação com os analógicos, assinale a alternativa incorreta.

a) Menor necessidade de utilização de conversores de sinais A/D e D/A.
b) Os sistemas digitais possuem maior capacidade de armazenamento de informações.
c) Ruídos ou pequenas flutuações de tensão afetam menos os circuitos digitais.
d) Os circuitos digitais são mais fáceis de serem projetados.

Comentários

Analisemos as afirmativas.

A afirmativa "A" está **errada**, uma vez que os **circuitos digitais** **essencialmente** trabalham com a conversão de sinais de digital para analógico e de analógico para digital.

A afirmativa "B" está **correta**, uma vez que é possível armazenar muito mais dados em uma mídia digital (*por exemplo, um cartão de memória)* do que em uma mídia analógica (*por exemplo, uma folha de papel*).

A afirmativa "C" está **correta**, pois os **circuitos digitais** assumem **estágios** pré-definidos assim que os níveis de tensão ultrapassam determinados **patamares**, ou seja, são menos propensos a erros devido a ruídos e pequenas flutuações que os circuitos analógicos.

A afirmativa "D" **também está correta**, uma vez que a elaboração de **circuitos digitais** podeser feita através do uso de **técnicas computacionais**, ou até mesmo manuais (como **Mapas de Karnaugh**). Também diversos blocos já se encontram prontos para o uso, necessitando apenas o encadeamento de diversos blocos para se obter o resultado desejado. Também é possível se efetuar a programação de um chip via linguagem de programação, como Assembly ou C, por exemplo.

Portanto, o gabarito é a letra A.

16.(FUNDEP/SAAE de Itabira-MG - 2019)

Analise o mapa de Karnaugh a seguir.

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	1	0	0	1
$\overline{A}B$	0	1	1	0
AB	0	1	1	0
$A\overline{B}$	0	0	0	0

A equação simplificada a partir desse mapa é:

a) $S=\overline{A}\cdot\overline{C}+\overline{A}\cdot\overline{D}\cdot C$.

b) $S=A\cdot\overline{B}\cdot\overline{C}+\overline{B}\cdot D$.

c) $S=A\cdot B\cdot D+B\cdot\overline{C}\cdot\overline{D}$.

d) $S=\overline{A}\cdot\overline{B}\cdot\overline{D}+B\cdot D$.

Comentários:

Agrupando os termos no **Mapa de Karnaugh**, temos:

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	1	0	0	1
$\overline{A}B$	0	1	1	0
AB	0	1	1	0
$A\overline{B}$	0	0	0	0

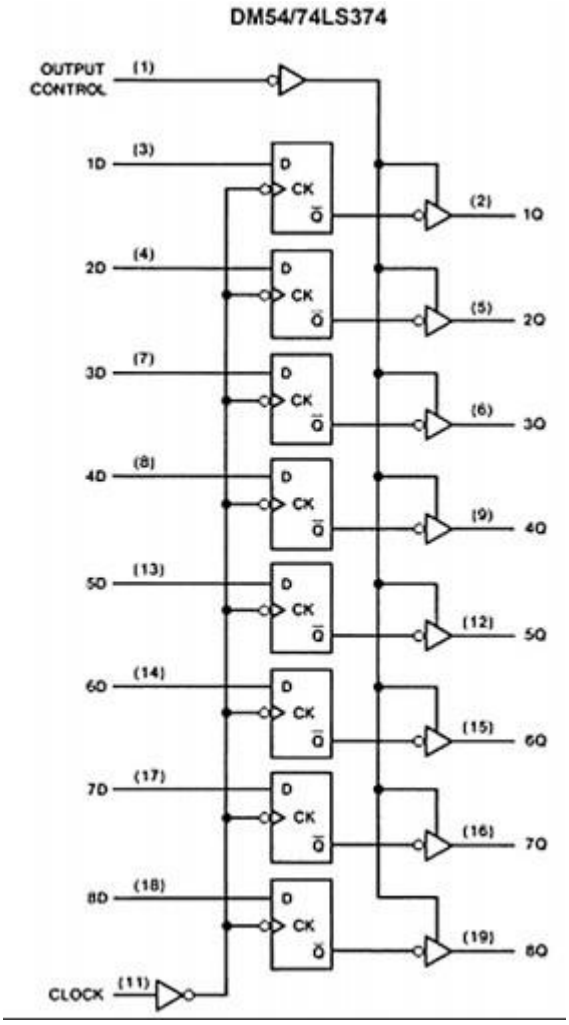
Esta expressão pode ser resumida então em:

$$S=B\cdot D+\overline{A}\cdot\overline{B}\cdot\overline{D}$$

Portanto, **o gabarito é a letra D.**

17. (UFMG/UFMG - 2019)

Analise a figura que mostra o diagrama interno do circuito integrado 74LS374:



A alternativa que indica a função lógica CORRETA do circuito é:

a) Registrador de deslocamento com entrada paralela e saída paralela.
b) Registrador de entrada paralela e saída paralela com coletor aberto.
c) Registrador de entrada paralela e saída paralela com três estados.
d) Contador de 8 bits síncrono com saída em coletor aberto.

Comentários:

Não se trata de um contador, uma vez que os flip-flops são independentes entre si.

Portanto, descarta-se a alternativa D.

Não se trata de um registrador de deslocamento, uma vez que os **flip-flops** não estão ligadosem série, e sim em paralelo. **Portanto, descarta-se a alternativa A**.

Trata-se de um registrador de entrada paralela e saída paralela, porém a saída não é com coletor aberto, uma vez que há um sinal de controle para a saída. **Portanto, descarta-se a alternativa B**.

Devido ao fato de o *buffer de saída* do circuito integrado possuir uma segunda entrada, pode-se dizer que o circuito integrado possui saída paralela com 3 estados: **1, 0 ou Z (alta impedância)**. Uma saída Z pode ser considerada como desconectada do circuito. Isto é útil quando se precisa que um bit específico seja lido por uma varredura feita por outro dispositivo, por exemplo. A saída em Z é diferente da saída 0, uma vez que 0 não significa ausência de tensão, e sim apenas uma convenção para um nível lógico baixo.

Portanto, o gabarito é a letra C.

LISTA DE QUESTÕES

1. (UFCG / UFCG - 2019)

Um técnico de laboratório precisa montar um circuito digital de 3 entradas e 1 saída que obedeça a Tabela da Verdade apresentada na tabela abaixo.

Na tabela, A, B e C representam as entradas do circuito digital e Y representa a saída do circuito digital.

É correto afirmar que a expressão lógica de Y é:

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

a) $Y=\overline{C}+A\cdot B$

b) $Y=\overline{A\cdot B}+C$

c) $Y=\overline{C}\cdot(\overline{A+B})$

d) $Y=\overline{C+A\cdot B}$

e) $Y=\overline{C}+\overline{A\cdot B}$

2. (VUNESP/TJM-SP – 2021)

Considere a seguinte tabela-verdade, com três entradas (A, B, C) e uma saída S.

A	B	C	S
0	0	0	**1**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

A saída ***S*** é corretamente expressa em:

Alternativas

A) $A.B.C$

B) $\overline{A}.\overline{B}.\overline{C}$

C) $A \oplus B \oplus C$

D) $(\overline{A}.\overline{B}.\overline{C}) + (A.B.C)$

E) $(\overline{A} + \overline{B} + \overline{C}) \bullet (A + B + C)$

3. (CS-UFG / UFG - 2019)

Tratando-se de portas lógicas e álgebra booleana, a tabela abaixo diz respeito à operação

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

a) $X=\overline{A+B}$

b) $X=\overline{A \oplus B}$

c) $X=A+B$

d) $X=A \oplus B$

4. (CESPE/ FUB - Técnico em Telecomunicações/ 2015)

A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

A tabela verdade apresentada acima representa o comportamento de um circuito combinacional com entradas A, B e C e saída S.

Com relação a essa tabela, julgue o próximo item.

Na forma soma de produtos, a expressão booliana mínima para S é dada por $S = \bar{A}\bar{B}\bar{C} + A$:

A) Certo.
B) Errado.

5. (CS-UFG / UFG - 2019)

A representação do número decimal 513 em octal é:

a) 201.

b) 0101 0001 0011.

c) 1001.

d) 0001 0000 0001.

6. (CS-UFG / UFG - 2019)

No Flip-Flop tipo J-K, quando ocorre um pulso de *clock* e as entradas são J = 1 e K = 0, a saídaQ

a) é 0.

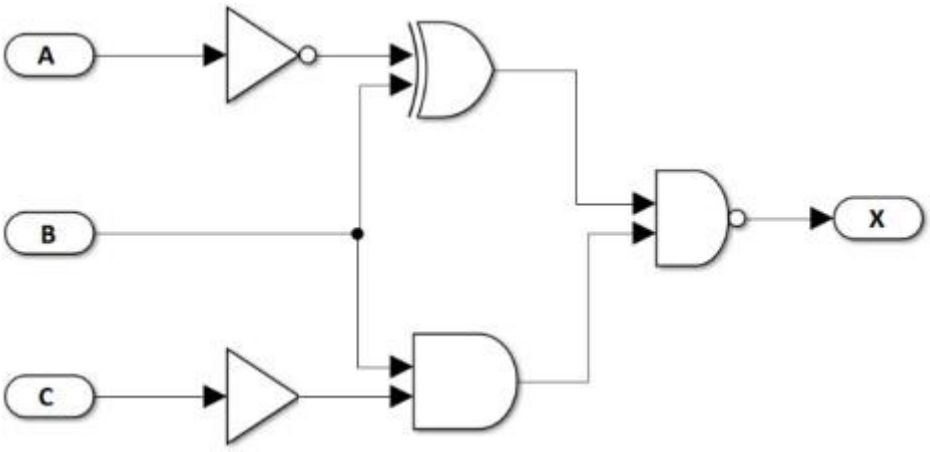
b) é 1.

c) mantém o valor da saída anterior.

d) alterna o valor da saída anterior.

7. (CS-UFG / UFG - 2019)

Considere o circuito combinacional a seguir.



A expressão para a saída X é:

a) $X=\overline{(\overline{A}\oplus B)\cdot(B\cdot C)}$.

b) $X=\overline{(\overline{A}+B)\cdot(B\cdot C)}$.

c) $X=\overline{(\overline{A}\oplus B)\cdot(B\cdot \overline{C})}$.

d) $X=\overline{(\overline{A}+B)\cdot(B\cdot \overline{C})}$.

8. (CS-UFG / UFG - 2019)

Qual das alternativas abaixo diz respeito apenas à família lógica bipolar?

a) TTL, VHDL, RTL e DTL.

b) TTL, HTL, I2C e RTL.

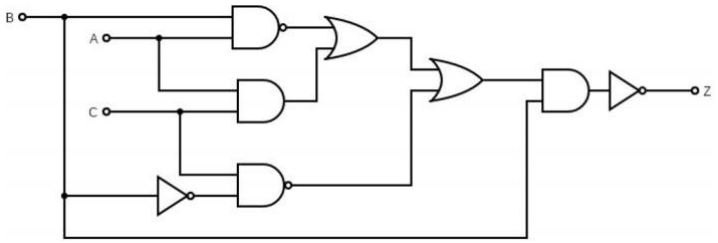
c) TTL, VHDL, I2C e RTL.

d) TTL, HTL, RTL e DTL.

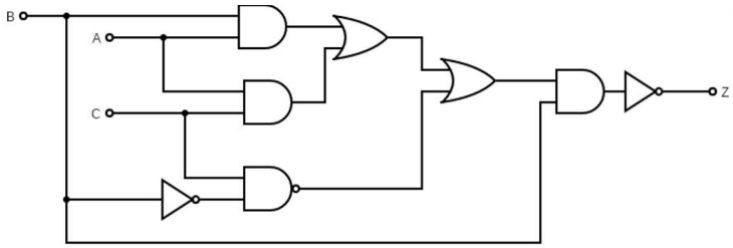
9. (FUNDEP/Prefeitura de Uberlândia - 2019)

Considerando quatro variáveis binárias A, B, C e Z, assinale a alternativa que apresenta o circuito lógico que pode ser utilizado para implementar a expressão: $Z=\overline{(\overline{AB}+CA+\overline{\overline{B}C})B}$.

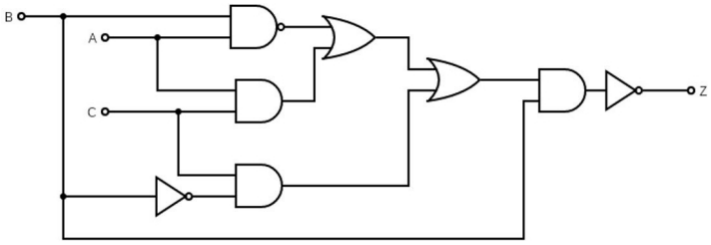
a)



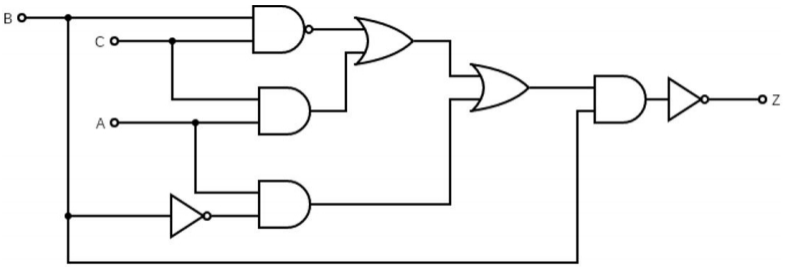
b)



c)



d)



10.(FUNDEP/Prefeitura de Uberlândia - 2019)

Em relação aos sistemas digitais, a conversão decimal binário é de fundamental importância. Assinale a alternativa que apresenta o equivalente binário ao número decimal 35.

a) 100001

b) 100111

c) 110001

d) 100011

11. (FUNDEP/Prefeitura de Uberlândia - 2019)

O sistema de base 16, ou hexadecimal, utiliza 16 símbolos para a representação numérica. Assinale a alternativa que apresenta a representação decimal para o número hexadecimal 5D3B.

a) 32

b) 23866

c) 23867

d) 512

12. (FGV/Prefeitura de Salvador - 2019)

Considere a tabela verdade de um dado circuito digital, a seguir.

a	b	c	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

A saída Y desse circuito é:

a) $Y = bc + a\bar{c} + \bar{a}c + \bar{a}b$

b) $Y = b\bar{c} + a\bar{c} + \bar{a}c + \bar{a}b$

c) $Y = b\bar{c} + a\bar{c} + \bar{a}c + ac$

d) $Y = b\bar{c} + a\bar{c} + bc + \bar{a}b$

e) $Y = b\bar{c} + a\bar{c} + \bar{a}\bar{c} + \bar{a}b$

13. (IBFC/Prefeitura de Cabo de Santo Agostinho - 2019)

A capacidade de armazenar informações binárias por um período de tempo é uma função imprescindível para os sistemas digitais. Sobre o dispositivo responsável por tal armazenamento em circuitos digitais, assinale a alternativa correta.

a) Portas lógicas

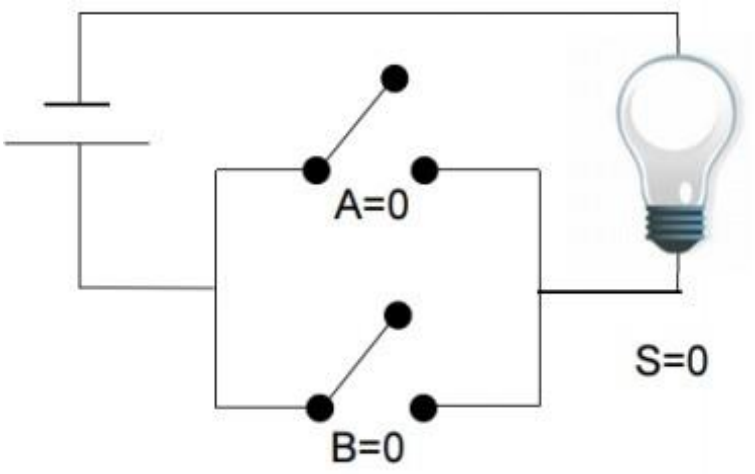
b) Comparador

c) Multiplexador

d) Registrador

14. (IBFC/Prefeitura de Cabo de Santo Agostinho - 2019)

A lâmpada ilustrada na figura abaixo deve ser acesa e, para executar essa função, utiliza-se a porta lógica OU. Para tanto, tem-se as chaves idênticas A e B que se encontram inicialmente na posição aberta (binário 0). Sabendo que a posição fechada é representada pelo binário 1, analise as afirmativas abaixo e dê valores de Verdadeiro (V) ou Falso (F).



( ) Se A = 0 e B = 1 a lâmpada irá

acender.

( ) Se A = 1 e B = 0 a lâmpada irá

apagar.

( ) Se A = 1 e B = 1 a lâmpada não irá acender, pois ocorrerá um curto-circuito.

Assinale a alternativa que apresenta a sequência correta de cima para baixo.

a) F, F, F
b) V, F, F
c) F, V, V
d) V, F, V

15. (IBFC/Prefeitura de Cabo de Santo Agostinho - 2019)

As mais diversas áreas têm apresentado avanços tecnológicos significantes nas últimas décadas e muito disso se faz devido aos sistemas digitais. Sobre as vantagens dos sistemas digitais em comparação com os analógicos, assinale a alternativa incorreta.

a) Menor necessidade de utilização de conversores de sinais A/D e D/A.

b) Os sistemas digitais possuem maior capacidade de armazenamento de informações.

c) Ruídos ou pequenas flutuações de tensão afetam menos os circuitos digitais.

d) Os circuitos digitais são mais fáceis de serem projetados.

16. (FUNDEP/SAAE de Itabira-MG - 2019)

Analise o mapa de Karnaugh a seguir.

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	1	0	0	1
$\overline{A}B$	0	1	1	0
AB	0	1	1	0
$A\overline{B}$	0	0	0	0

A equação simplificada a partir desse mapa é:

a) $S=\overline{A}\cdot\overline{C}+\overline{A}\cdot\overline{D}\cdot C$.

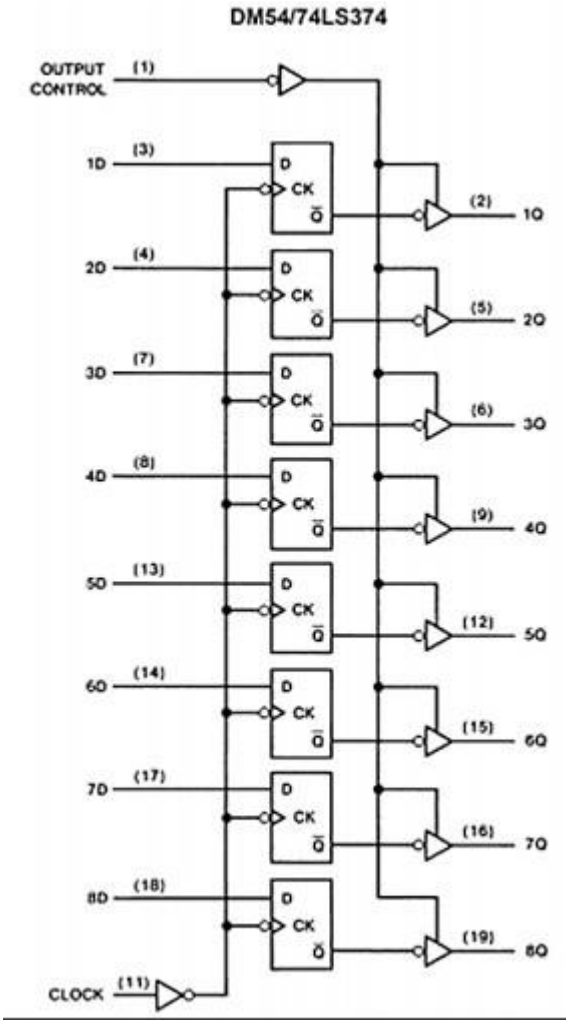
b) $S=A\cdot\overline{B}\cdot\overline{C}+\overline{B}\cdot D$.

c) $S=A\cdot B\cdot D+B\cdot\overline{C}\cdot\overline{D}$.

d) $S=\overline{A}\cdot\overline{B}\cdot\overline{D}+B\cdot D$.

17. (UFMG/UFMG - 2019)

Analise a figura que mostra o diagrama interno do circuito integrado 74LS374:



A alternativa que indica a função lógica CORRETA do circuito é:

a) Registrador de deslocamento com entrada paralela e saída paralela.
b) Registrador de entrada paralela e saída paralela com coletor aberto.
c) Registrador de entrada paralela e saída paralela com três estados.
d) Contador de 8 bits síncrono com saída em coletor aberto.

GABARITO

1. D
2. D
3. B
4. Errado.
5. C
6. B
7. A
8. D
9. A
10. D
11. C
12. B
13. D
14. B
15. A
16. D
17. C

REFERÊNCIAS BIBLIOGRÁFICAS

MALVINO, A.; BATES, D. **Eletrônica:** Volume I: 8ª Edição. Porto Alegre: Editora McGraw Hill, 2016.

MENDONÇA, A. ZELENOVSKY, R. **Eletrônica Digital: Curso Prático e Exercícios:** 2ª Edição. Rio de Janeiro: Editora MZ, 2008.

NILSSON, J. W.; RIEDEL, S. A. **Circuitos Elétricos:** 6ª Edição. Rio de Janeiro: Editora LTC, 2003.

SEDRA, A. S.; SMITH, K. C. **Microeletrônica:** 5ª Edição. São Paulo: Editora Pearson Prentice Hall, 2007.

TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais: Princípios e Aplicações:** 8ª Edição. São Paulo: Editora Pearson Prentice Hall, 2003.

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.