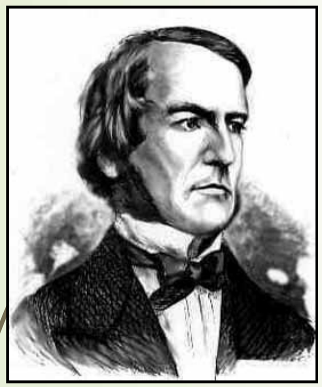
Algebra de Boole

Introducción

George Boole

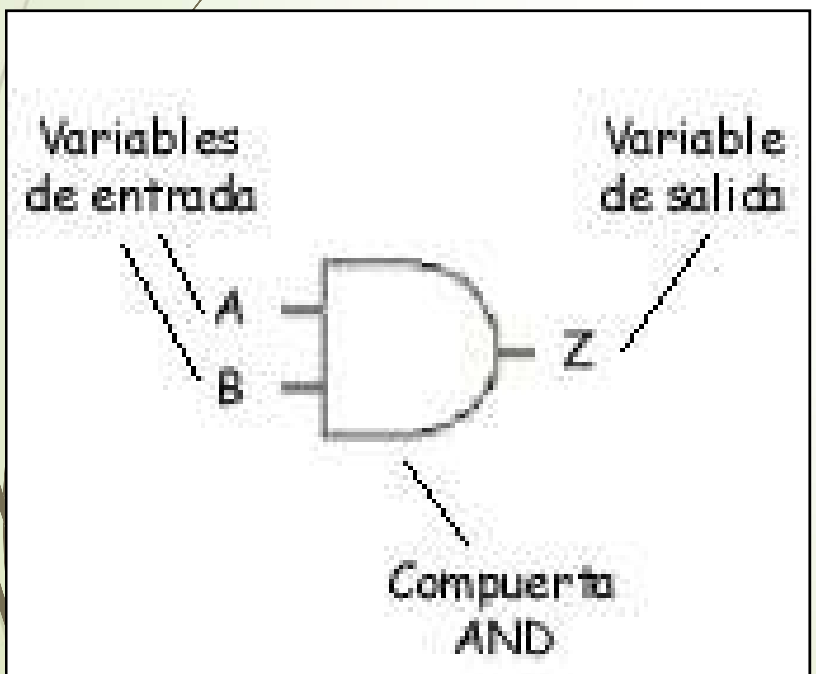


El matemático inglés George Boole nació el 2 de noviembre de 1815 en Lincoln y falleció el 8 de diciembre de 1864 en Ballintemple, Irlanda.

Boole resumió la lógica a una álgebra simple. También trabajó en ecuaciones diferenciales, el cálculo de diferencias finitas y métodos generales en probabilidad.

Compuertas Lógicas AND

Una **Compuerta AND** de dos entradas es un dispositivo electrónico que posee dos “cables en su entrada”, a las que llegan los niveles de tensión de sus dos cables (**A** y **B**) y tiene una sola salida (**Z**).



Responde a la expresión:

$$\mathbf{Z = A \cdot B}$$

Compuertas Lógicas AND

$$A \cdot B = Z$$

$$0 \cdot 0 = 0$$

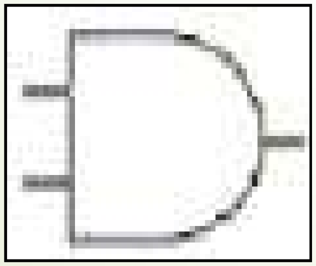
$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

0 1 0 1

0 0 1 1

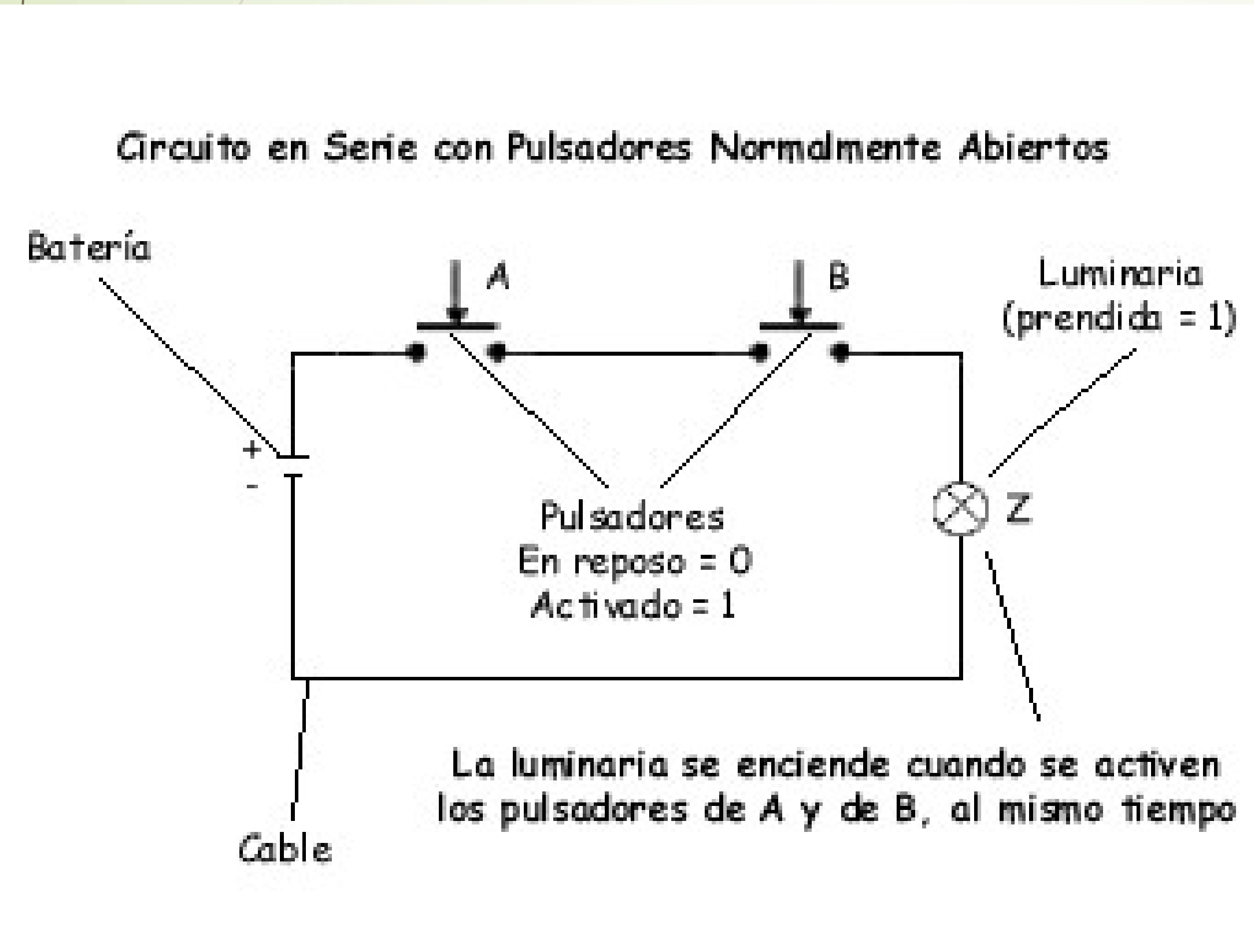


0 0 0 1

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

Circuito Lógico AND

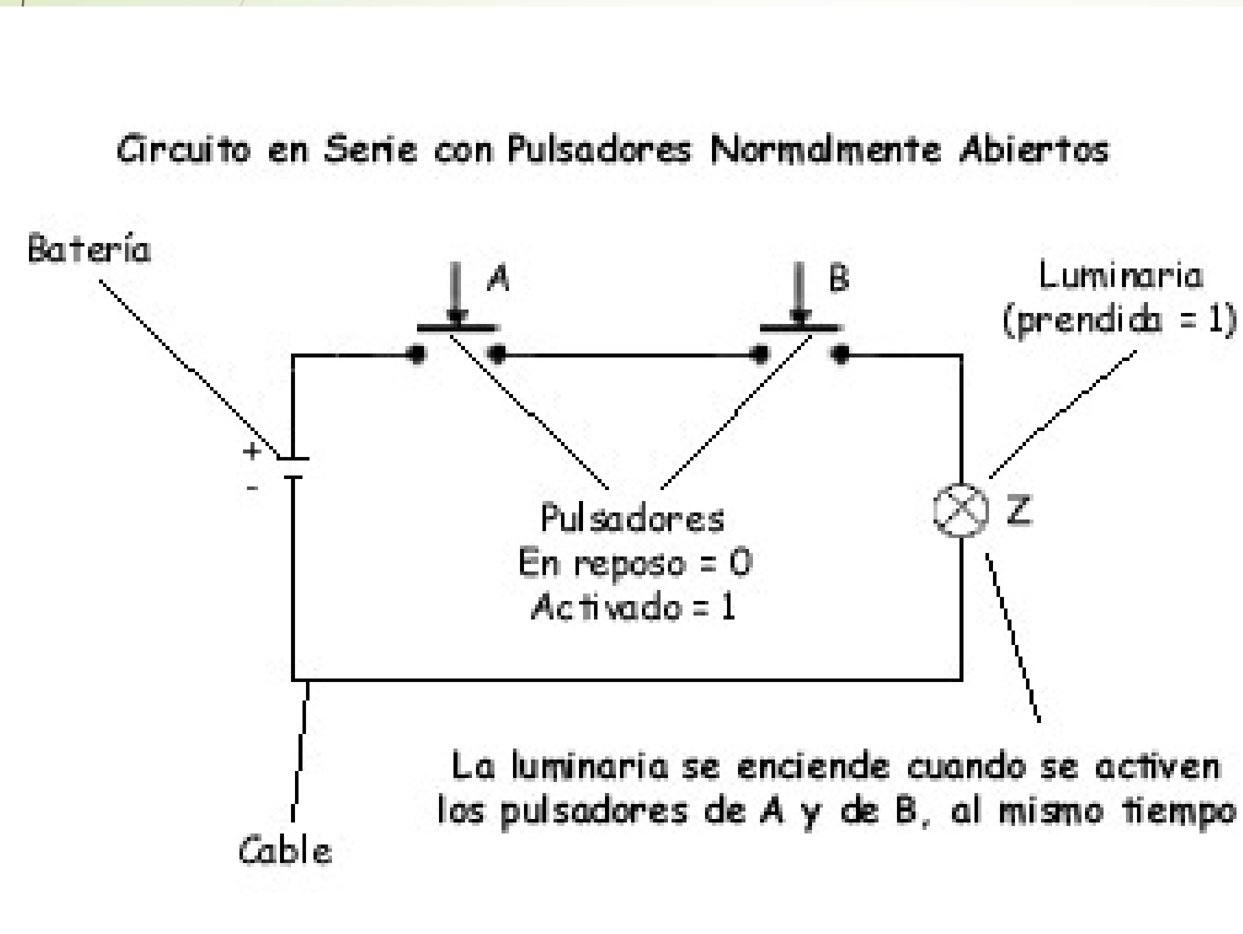
$$Z = A \cdot B$$



*También es posible representar la función lógica, su tabla de verdad y su compuerta con los pulsadores NC, formando un **"circuito lógico".***

Circuito Lógico AND

$$Z = A \cdot B$$

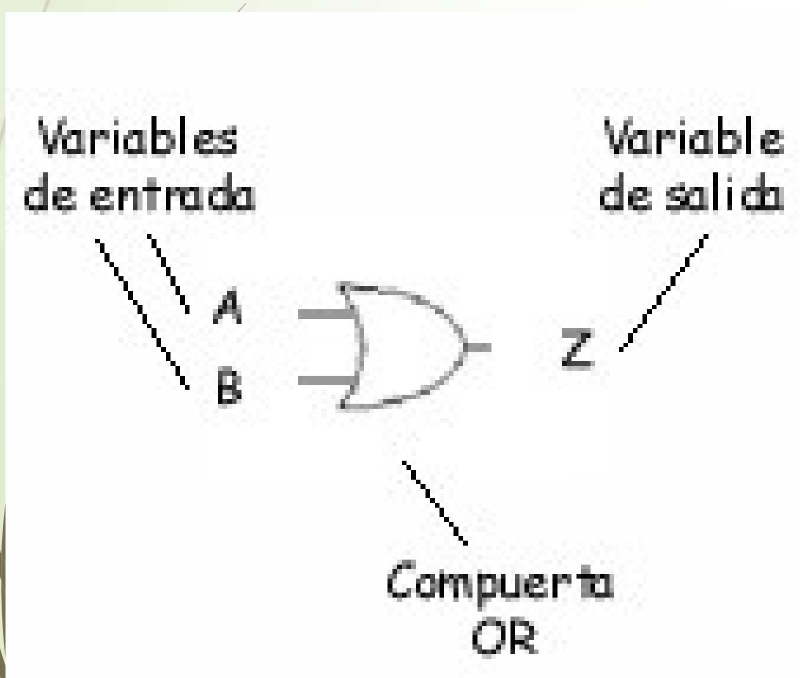


La luminaria se enciende cuando A y B son pulsados al mismo tiempo.

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

Compuertas Lógicas OR

Una **Compuerta OR** de dos entradas es un dispositivo electrónico que posee dos entradas, a las que llegan los niveles de tensión de dos cables (**A** y **B**) y una salida (**Z**).



Responde a la expresión:

Z = A + B

Compuertas Lógicas OR

A + B = Z

0 + 0 = 0

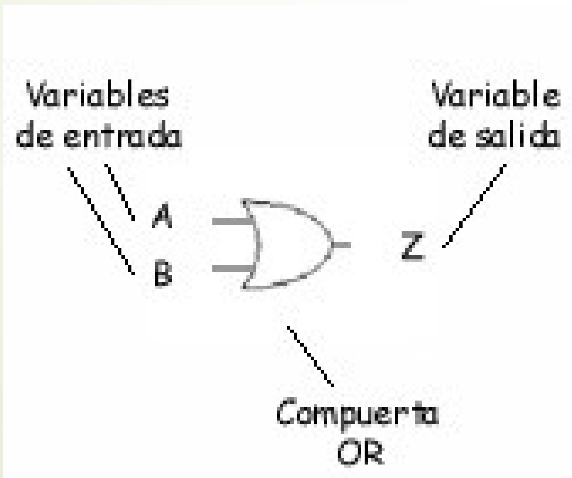
1 + 0 = 1

0 + 1 = 1

1 + 1 = 1

0 1 0 1

0 0 1 1



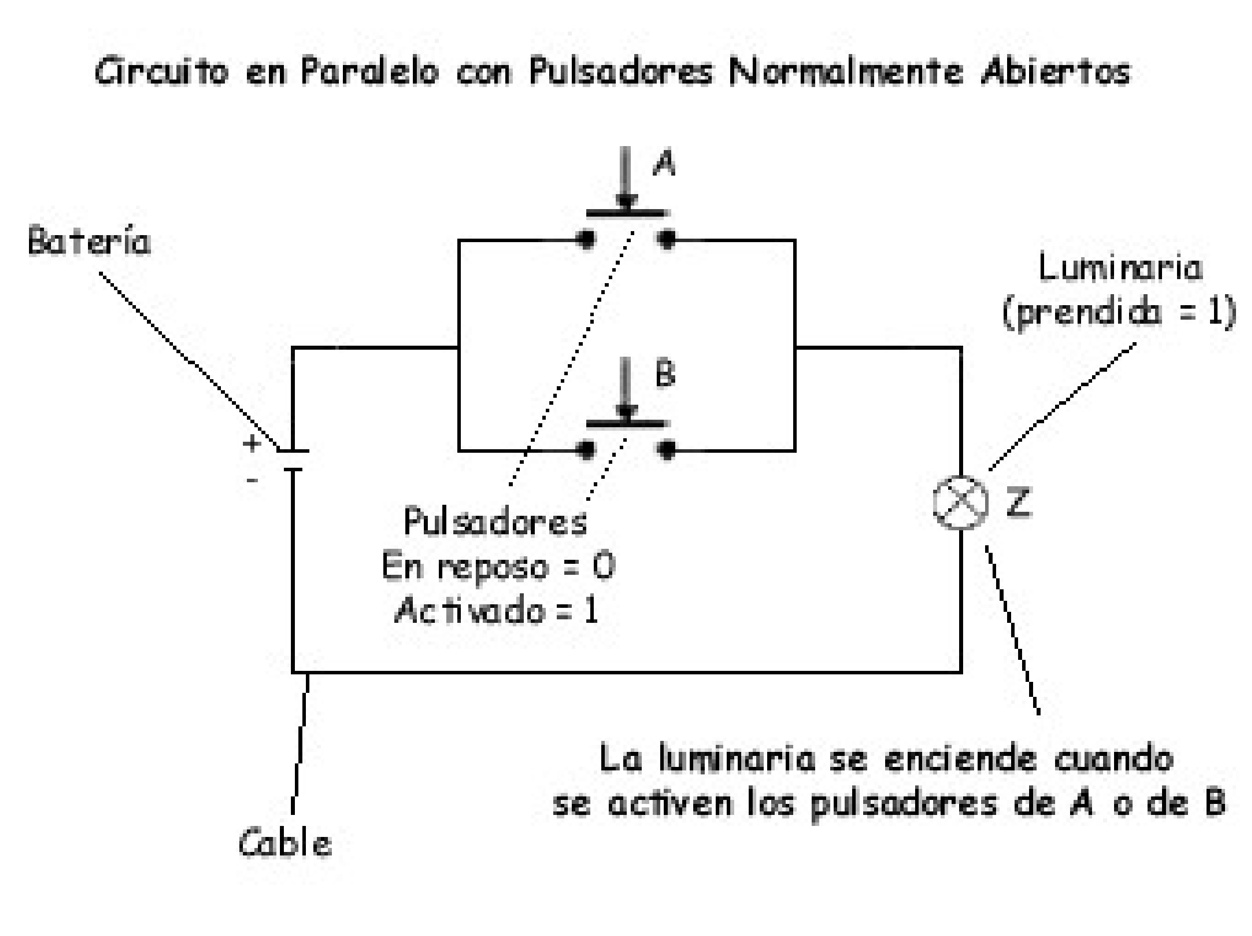
0 1 1 1

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Circuito Lógico OR

$$Z = A + B$$

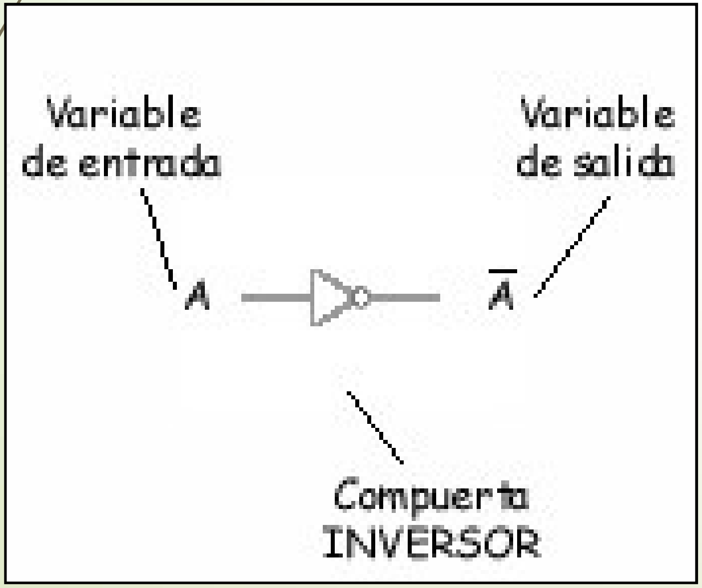
La luminaria se enciende cuando A o B son pulsados.



A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Compuertas Lógicas Inversor

Una **Compuerta INVERSOR** es un dispositivo electrónico que enciende el cable que está en su salida, si el cable que está en su entrada se encuentra apagado, y viceversa. Puede decirse que uno es la negación del otro.



Responde a la expresión:

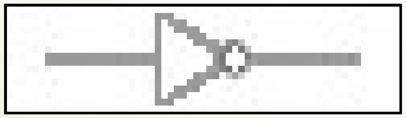
$$Z = \overline{A}$$

Compuertas Lógicas Inversor

$$\overline{A} = Z$$

0 = 1

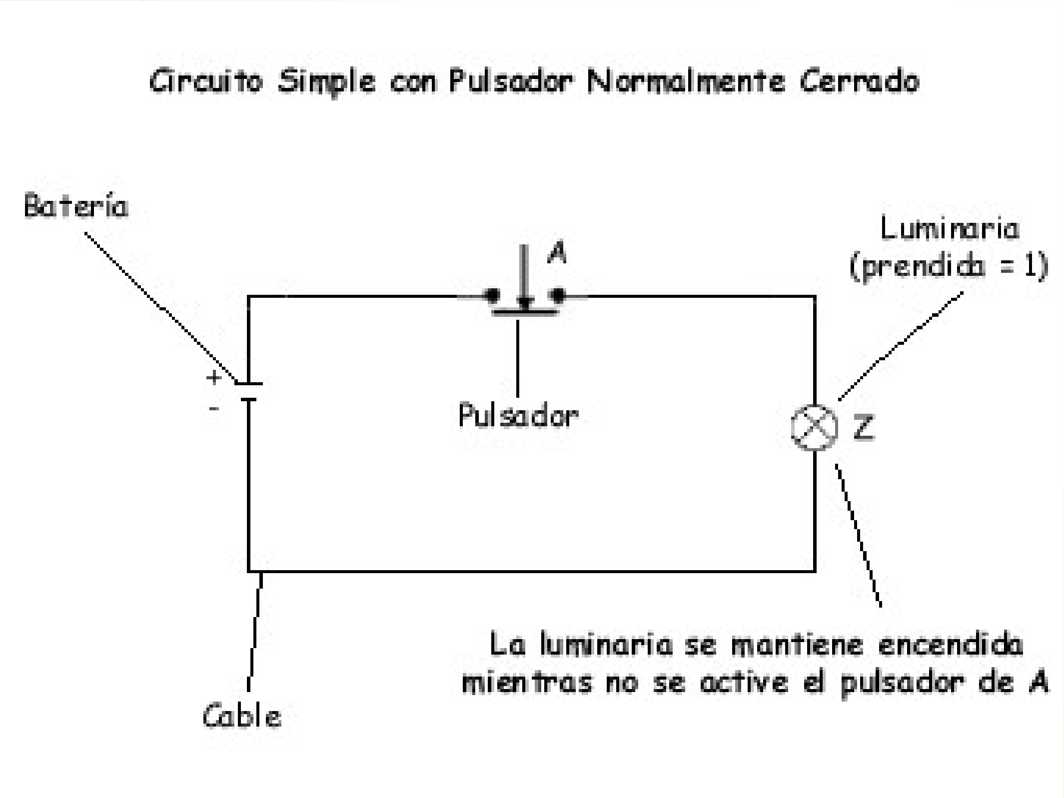
1 = 0

0 1  0 1

A	Z
0	1
1	0

Circuito Lógico Inversor

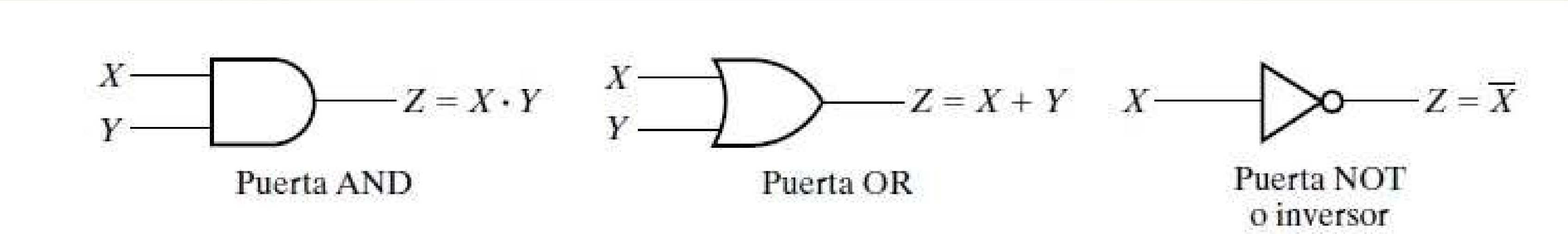
$$Z = \overline{A}$$

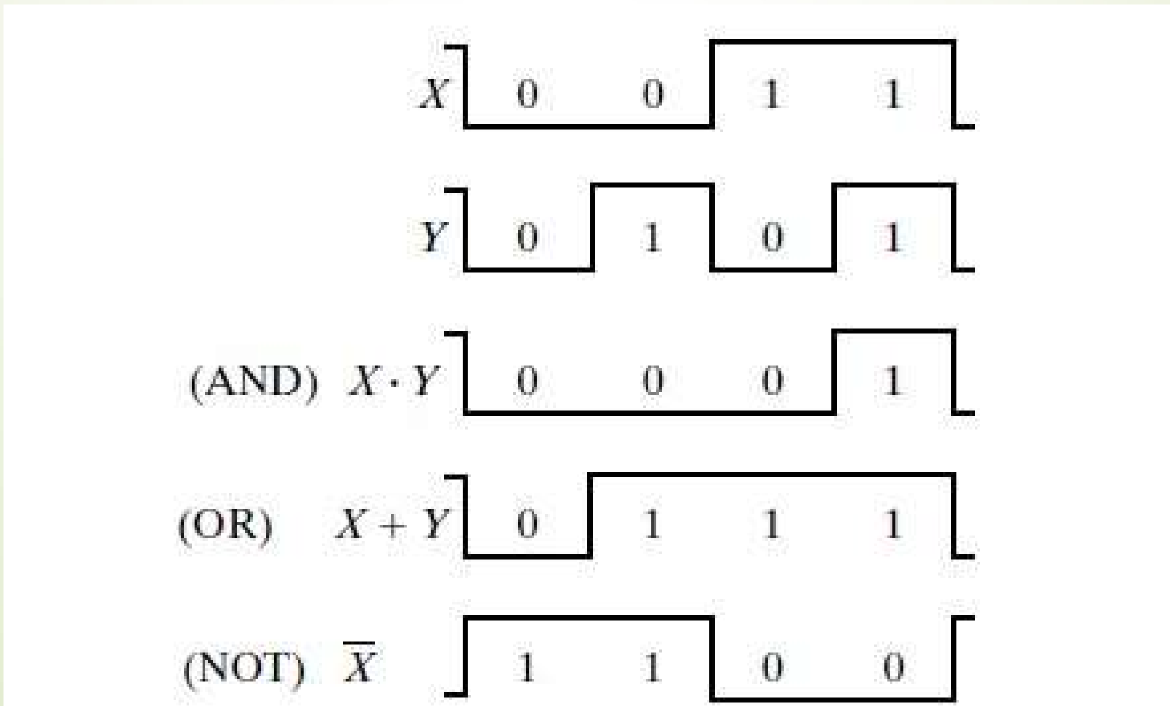


La luminaria se activará si A toma el valor 0 y viceversa.

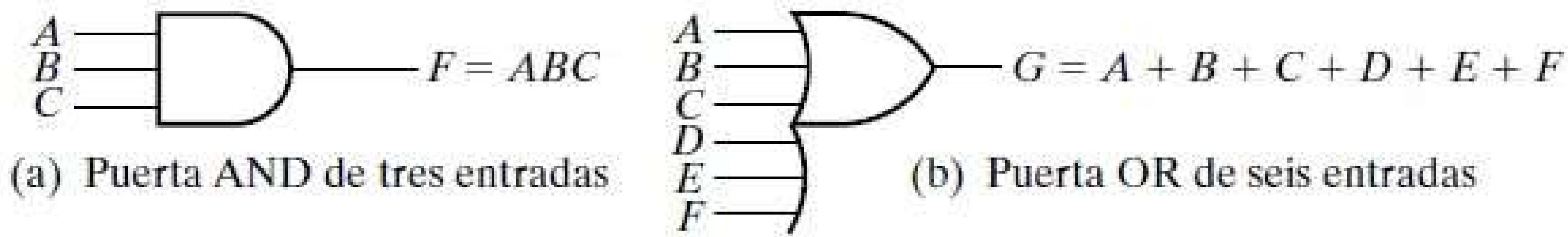
A	Z
0	1
1	0

Resumen puertas Lógicas Primarias





Puertas con mas de 2 entradas



(a) Puerta AND de tres entradas

(b) Puerta OR de seis entradas

Leyes del Algebra de Boole

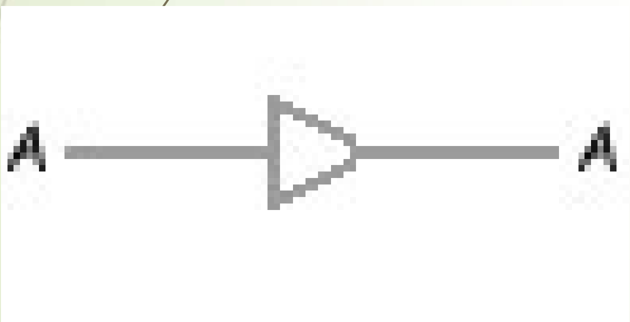
Identidades básicas del Álgebra de Boole

1.	$X + 0 = X$	2.	$X \cdot 1\ X$	
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	
7.	$X + \bar{X} = 1$	8.	$X \cdot \bar{X} = 0$	
9.	$\bar{\bar{X}} = X$			
10.	$X + Y = Y + X$	11.	$XY = YX$	Conmutativa
12.	$X + (Y + Z) = (X + Y) + Z$	13.	$X(YZ) = (XY)Z$	Asociativa
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributiva
16.	$\overline{X + Y} = \bar{X} \cdot \bar{Y}$	17.	$\overline{X \cdot Y} = \bar{X} + \bar{Y}$	De DeMorgan

En la Tabla se enumeran las identidades básicas del Álgebra de Boole.

Otras Compuertas Lógicas: Seguidor

Una **Compuerta SEGUIDOR** es un dispositivo electrónico que actúa como buffer: mantiene en la salida, el valor que se encuentra a la entrada.



Responde a la expresión:

Z = A

Compuertas Lógicas Seguidor

A = Z

0 = 0

1 = 1

0 1

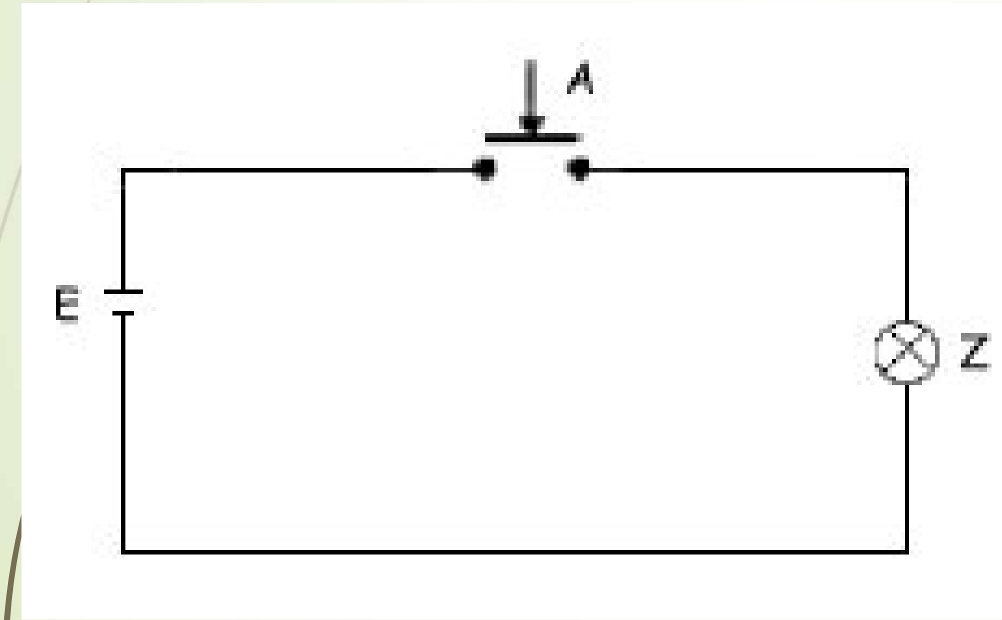
0 1

A	Z
0	0
1	1

Circuito Lógico Seguidor

Z = A

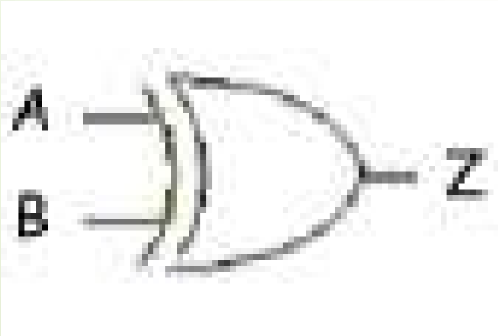
La luminaria se enciende cuando A es pulsado.



A	Z
0	0
1	1

Compuertas Lógicas XOR

Una **compuerta XOR** u ***OR excluyente*** de dos entradas es un dispositivo electrónico que presenta dos entradas, a las que llegan los estados de las dos variables (**A** $\oplus$ **B**), y una salida, que genera en el cable (**Z**).



Responde a la expresión:

$$Z = \overline{A} \cdot B + \overline{B} \cdot A$$

Compuertas Lógicas XOR

$$A \oplus B = Z$$

$0 \oplus 0$

$0 \oplus 1$

$1 \oplus 0$

$1 \oplus 1$

$$Z = \overline{A} \cdot B + \overline{B} \cdot A$$

$1 \cdot 0 + 1 \cdot 0$

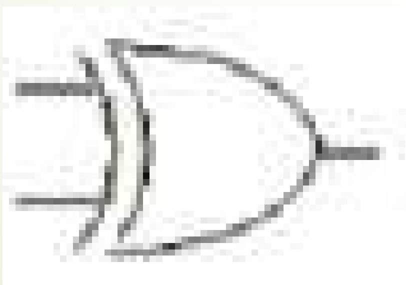
$1 \cdot 1 + 0 \cdot 0$

$0 \cdot 0 + 1 \cdot 1$

$0 \cdot 1 + 0 \cdot 1$

0 1 0 1

0 0 1 1



0 1 1 0

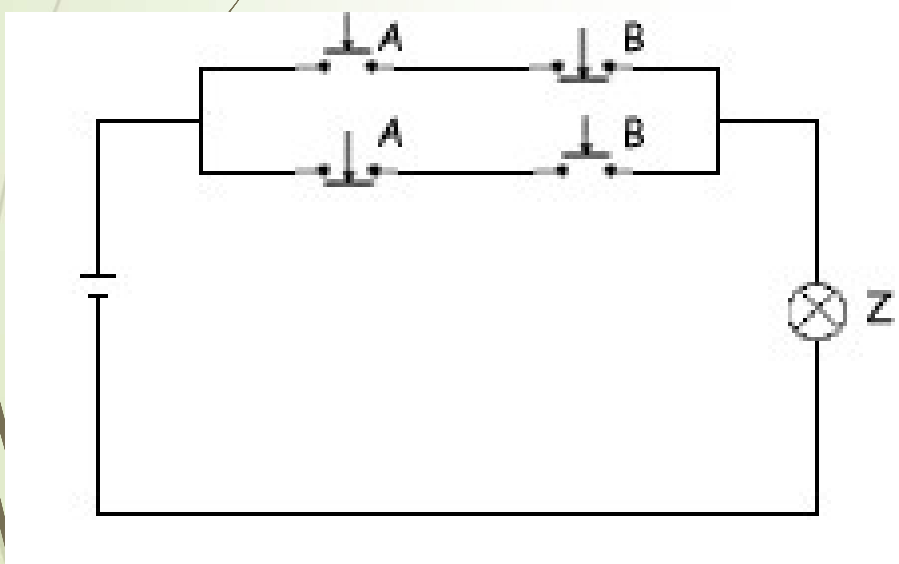
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Circuito Lógico XOR

Cuando ambos se activan al mismo tiempo, Z vale 0, c.c. vale 1.

$$Z = \overline{A} \cdot B + \overline{B} \cdot A$$

Z se activará si A o B se activan (o no), pero no al mismo tiempo



A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Compuertas Derivadas (NAND)

Una compuerta **NAND** resulta de invertir la salida de una compuerta **AND**.

Compuerta AND $\longrightarrow$ $Z = A \cdot B$

Invertimos la salida (NAND) $\longrightarrow$ $\overline{Z} = A \cdot B$

Negamos de ambos lados (porque se requiere Z) $\longrightarrow$ $\overline{\overline{Z}} = \overline{A \cdot B}$

Por ley de doble neg. $\longrightarrow$ $Z = \overline{A \cdot B}$

Por ley de Morgan $\longrightarrow$ $\boxed{Z = \overline{A} + \overline{B}}$

Expresión Booleana

Compuertas Lógicas (NAND)

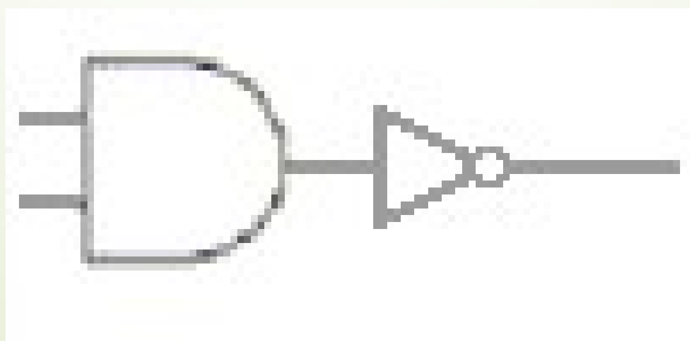
$$\overline{A}+\overline{B}=Z$$

$$\overline{0}+\overline{0}=1$$
$$\overline{0}+\overline{1}=1$$
$$\overline{1}+\overline{0}=1$$
$$\overline{1}+\overline{1}=0$$

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

0 1 0 1

0 0 1 1



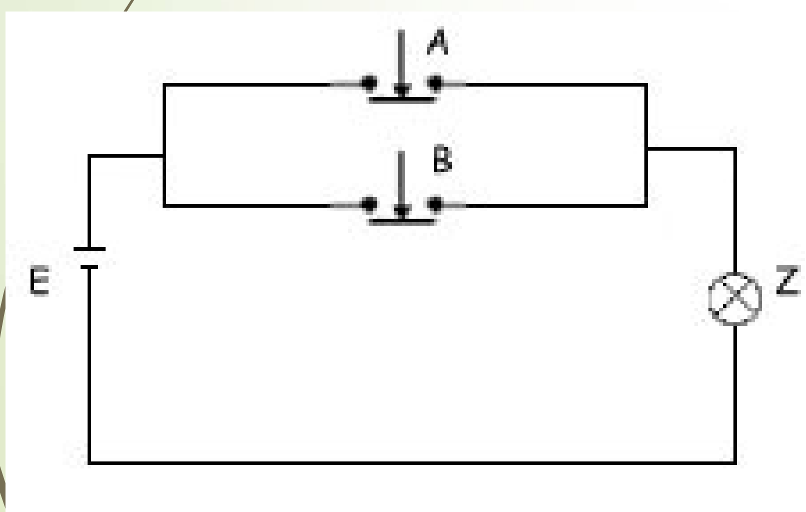
1 1 1 0

Circuito Lógico (NAND)

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

Esto coincide que, cuando A y B son iguales a 1, haciendo que Z sea igual a 0.

Z será igual a 0 sólo si A y B se activan en conjunto.



A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

$$\rightarrow \overline{0.0} = \overline{0} = 1$$

$$\rightarrow \overline{0.1} = \overline{0} = 1$$

$$\rightarrow \overline{1.0} = \overline{0} = 1$$

$$\rightarrow \overline{1.1} = \overline{1} = 0$$

Compuertas Derivadas

Compuerta “NOR”

Una compuerta **NOR** resulta de invertir la salida de una compuerta **OR**.

Compuerta OR $\longrightarrow$ $Z = A + B$

Invertimos la salida (NOR) $\longrightarrow$ $\overline{Z} = A + B$

Negamos de ambos lados $\longrightarrow$ $\overline{\overline{Z}} = \overline{A + B}$

Por ley de doble neg. $\longrightarrow$ $Z = \overline{A + B}$

Por ley de Morgan $\longrightarrow$ $\boxed{Z = \overline{A} \cdot \overline{B}}$

Expresión Booleana

Compuertas Lógicas (NOR)

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

$$\overline{0}\cdot\overline{0}=1$$

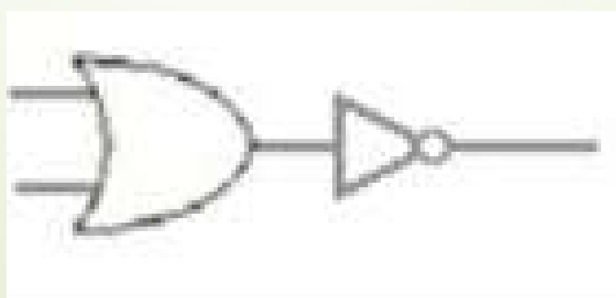
$$\overline{0}\cdot\overline{1}=0$$

$$\overline{1}\cdot\overline{0}=0$$

$$\overline{1}\cdot\overline{1}=0$$

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

0 1 0 1
0 0 1 1



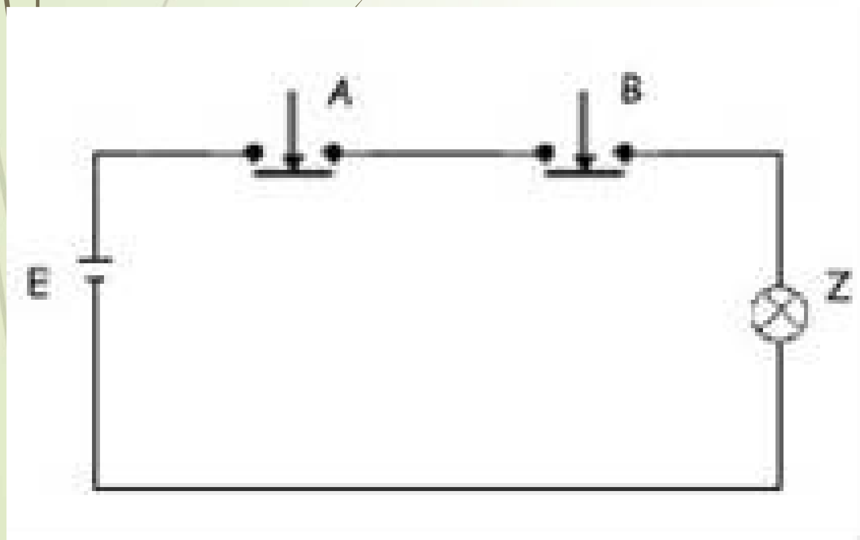
1 0 0 0

Circuito Lógico (NOR)

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

Esto es cuando A y B son iguales a 0, haciendo que Z sea igual a 1.

Z será igual a 1 si A o B no se presionan (activan) en ningún momento



A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

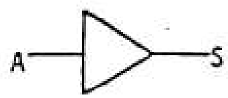
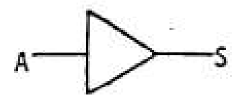
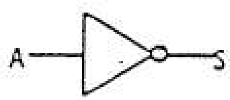
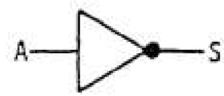
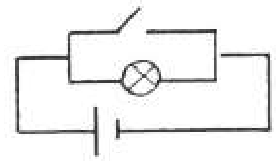
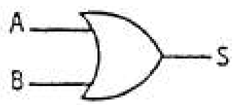
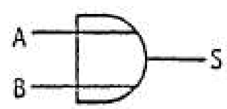
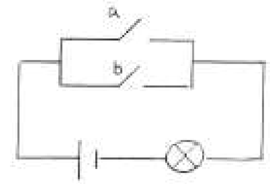
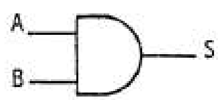
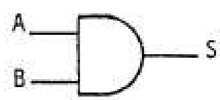
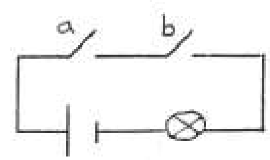
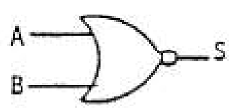
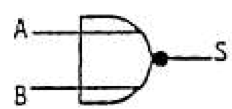
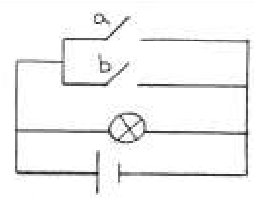
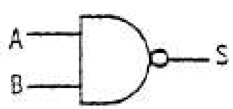
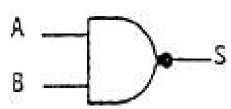
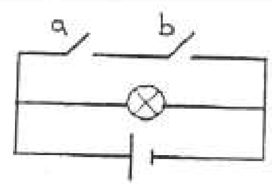
$\rightarrow \overline{0+0} = \overline{0} = 1$

$\rightarrow \overline{0+1} = \overline{1} = 0$

$\rightarrow \overline{1+0} = \overline{1} = 0$

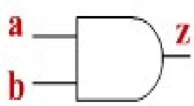
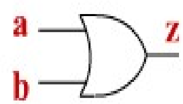
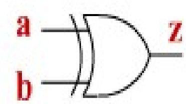
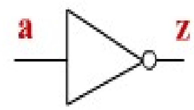
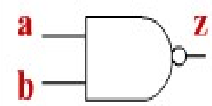
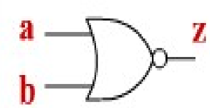
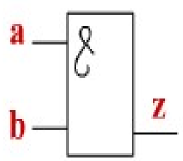
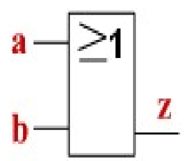
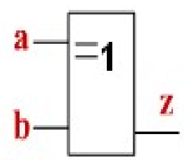
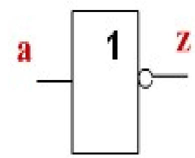
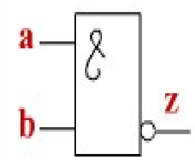
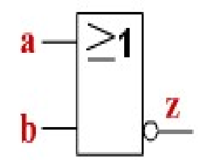
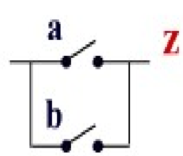
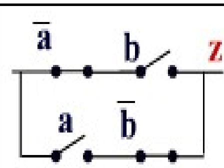
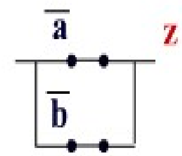
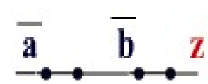
$\rightarrow \overline{1+1} = \overline{1} = 0$

Resumen Compuertas Lógicas

Puerta **NO INVERT** (no inversor) o **IGUALDAD**			$S = A$		***E*** ***S*** 0 0 1 1	el acoplamiento de circuitos sin inversión de la señal de entrada
Puerta **INVERT** (inversor) o **NOT**			$S = \bar{A}$		***E*** ***S*** 0 1 1 0	Invierte el estado de la señal aplicada a su entrada
Puerta **OR** (suma lógica)			$S=A+B$		***B*** ***A*** ***S*** 0 0 0 0 1 1 1 0 1 1 1 1	La salida es "0" cuando todas las entradas son "0"
Puerta **AND** (producto lógico)			$S=A*B$		***B*** ***A*** ***S*** 0 0 0 0 1 0 1 0 0 1 1 1	La salida es "1" cuando todas las entradas son "1"
Puerta **NOR** (suma negada)			$S=\overline{A+B}$		***B*** ***A*** ***S*** 0 0 1 0 1 0 1 0 0 1 1 0	La salida es "1" solo cuando todas las entradas son "0"
Puerta **NAND** (producto lógico negado)			$S=\overline{A*B}$		***B*** ***A*** ***S*** 0 0 1 0 1 1 1 0 1 1 1 0	La salida es "0" cuando todas las entradas son "1"

Resumen Compuertas Lógicas

FUNCIONES LÓGICAS BÁSICAS

NOMRE	AND - Y	OR - O	XOR O-exclusiva	NOT Inversor	NAND	NOR
SÍMBOLO						
SÍMBOLO						
TABLA DE VERDAD	a b z 0 0 0 0 1 0 1 0 0 1 1 1	a b z 0 0 0 0 1 1 1 0 1 1 1 1	a b z 0 0 0 0 1 1 1 0 1 1 1 0	a z 0 1 1 0	a b z 0 0 1 0 1 1 1 0 1 1 1 0	a b z 0 0 1 0 1 0 1 0 0 1 1 0
EQUIVALENTE EN CONTACTOS						
AXIOMA	$z = a . b$	$z = a + b$	$z = \bar{a} . b + a . \bar{b}$	$z = \bar{a}$	$z = \overline{a . b}$	$z = \overline{a + b}$

Mapas de Karnaugh – Simplificación de funciones booleanas

¿Qué son los Mapas de Karnaugh?

- Los **Mapas de Karnaugh** son una herramienta muy utilizada para la simplificación de circuitos lógicos. Cuando se tiene una **función lógica** con su tabla de verdad y se desea implementar esa función de la manera más económica posible se utiliza este método.

Ejemplo 1:

- Se tiene la siguiente tabla de verdad para tres variables. Se desarrolla la función lógica basada en ella. (primera forma canónica).
- Ver que en la fórmula se incluyen solamente las variables (A, B, C) cuando F es igual a "1". Si A en la tabla de verdad es "0" se pone la negación de A, si B = "1" se pone B, Si C = "0" se pone negación de C, etc.

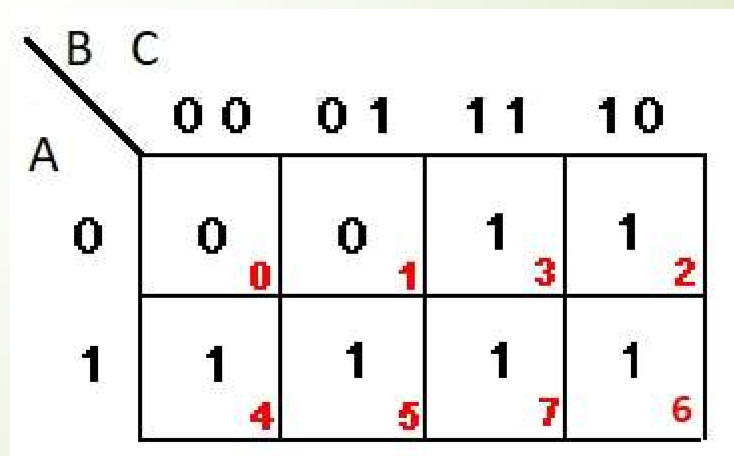
	A	B	C	F	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	1	$\rightarrow \bar{A}\,B\,\bar{C}$
3	0	1	1	1	$\rightarrow \bar{A}\,B\,C$
4	1	0	0	1	$\rightarrow A\,\bar{B}\,\bar{C}$
5	1	0	1	1	$\rightarrow A\,\bar{B}\,C$
6	1	1	0	1	$\rightarrow A\,B\,\bar{C}$
7	1	1	1	1	$\rightarrow A\,B\,C$

- La función lógica es:

$$F = \bar{A}\,B\,\bar{C} + \bar{A}\,B\,C + A\,\overline{B\,C} + A\,\bar{B}\,C + A\,B\,\bar{C} + A\,B\,C$$

Ejemplo 1: (cont.)

- Una vez obtenida la función lógica, se implementa el mapa de Karnaugh. Este tiene 8 casillas que corresponden a 2^n, donde n = 3 (número de variables (A, B, C)).
- La primera fila corresponde a A = 0. La segunda fila corresponde a A = 1. La primera columna corresponde a BC = 00 (B=0 y C=0).
- La segunda columna corresponde a BC = 01 (B=0 y C=1). La tercera columna corresponde a BC = 11 (B=1 y C=1). La cuarta columna corresponde a BC = 10 (B=1 y C=0).



Ejemplo: (cont.)

- En el **mapa de Karnaugh** se han puesto “1” en las casillas que corresponden a los valores de F = “1” en la tabla de verdad. Tomar en cuenta la numeración de las filas de la tabla de verdad y la numeración de las casillas en el **mapa de Karnaugh.**

A \ B C	0 0	0 1	1 1	1 0
0	0	0	1	1
1	1	1	1	1

- Para proceder con la simplificación, se crean grupos de “1”s que tengan 1, 2, 4, 8, 16, etc. (solo potencias de 2). Los “1”s deben estar adyacentes (no en diagonal) y mientras más “1”s tenga el grupo, mejor.
- **La función mejor simplificada es aquella que tiene el menor número de grupos con el mayor número de “1” s en cada grupo**

Ejemplo: (cont.)

- Se puede ver del cuadro que hay dos grupos cada uno de cuatro "1"s (se permite compartir casillas entre los grupos). La nueva expresión de la función boolena simplificada se deduce del **mapa de Karnaugh**.

A \ BC	00	01	11	10
0	0	0	1	1
1	1	1	1	1

- Para el primer grupo (rojo): la simplificación da B (los "1"s de la tercera y cuarta columna corresponden a B sin negar)
- Para el segundo grupo (azul): la simplificación da A (los "1"s están en la fila inferior que corresponde a A sin negar).

Entonces el resultado es final corresponde a:

$$F = B + A \quad ó \quad F = A + B$$

Ejemplo 2: (cont.)

- La tabla de verdad siguiente:

	A	B	C	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

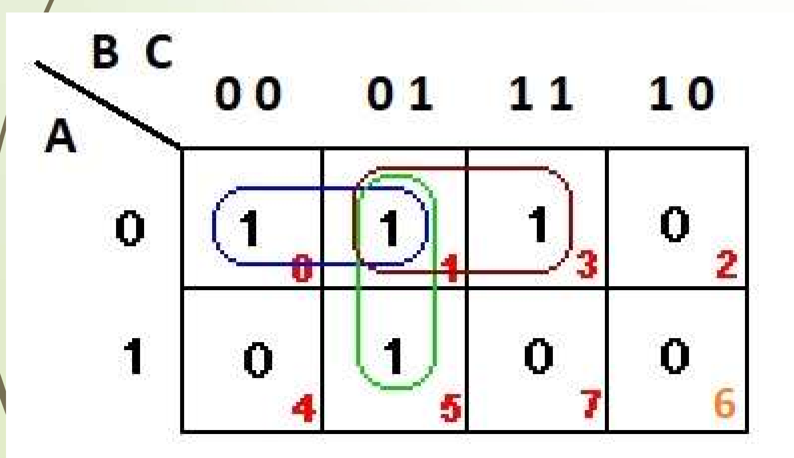
- Presenta como resultado la función booleana:

$$F = \overline{A\,B\,C} + \overline{A\,B}\,C + \overline{A}\,B\,C + A\,\overline{B}\,C$$

- Se ve claramente que la función es un reflejo del contenido de la tabla de verdad cuando F = “1”, Con esta ecuación se crea el **mapa de Karnaugh** y se escogen los grupos.

Ejemplo 2:

- Se lograron hacer 3 grupos de dos “1”s cada uno. Se puede ver que no es posible hacer grupos de 3, porque 3 no es potencia de 2. Se observa que hay una casilla que es compartida por los tres grupos.



- La función simplificada es:

$$F = \overline{A\,B} + \overline{A}\,C + \overline{B}\,C$$

- El grupo en azul representa a:

$$\overline{A\,B}$$

- El grupo café es:

$$\overline{A}\,C$$

- Y, finalmente, el grupo verde corresponde a:

$$\overline{B}\,C$$

Sistemas Numéricos Digitales

La Abstracción Digital

- La mayoría de las variables son **continuas**
  - Voltaje en un cable
  - Frecuencia de oscilador
  - Posición de un objeto
- La abstracción digital considera un **conjunto discreto** de valores (0, 1, 2, …)

Disciplina Digital : Valores Binarios

- **Dos valores discretos:**
  - 1’s y 0’s
  - 1, VERDADERO (true), ALTO (high)
  - 0, FALSO (falso), BAJO (low)
- **1 y 0:** niveles de voltaje, engranajes giratorios, niveles de fluido, etc.
- Los circuitos digitales usan niveles de **voltaje** para representar 1 y 0
- ***Bit***: Dígito *binario*

Sistemas numéricos

El sistema numérico decimal se emplea en la aritmética cotidiana para representar números mediante cadenas de dígitos. Dependiente de su posición en la cadena, cada dígito tiene un valor asociado a un entero como potencia en base 10. Por ejemplo, el número decimal 724.5 se interpreta de manera que representa 7 centenas, más 2 decenas, más 4 unidades y más 5 décimas. Las centenas, decenas, unidades y décimas son potencias de 10, dependiendo de la posición de los dígitos. El valor del número se calcula de la forma siguiente:

$$724.5 = 7 \times 10^2 + 2 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1}$$

La convención es escribir solamente los dígitos y deducir las potencias de 10 según su posición. En general, un número decimal con n dígitos a la izquierda del punto decimal y m dígitos a la derecha del punto decimal es representado por una cadena de coeficientes:

$$A_{n-1}A_{n-2}\ldots A_1A_0.A_{-1}A_{-2}\ldots A_{-m+1}A_{-m}$$

Sistemas numéricos

Cada coeficiente A_i es uno de los 10 dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9). El valor de subíndice i determina la posición del coeficiente y, asimismo el peso $10i$ con que hay que multiplicar el coeficiente.

Al sistema numérico decimal se llama base 10, porque se multiplican los coeficientes por potencias de 10 y el sistema usa 10 dígitos diferentes. En general, un número en base r contiene r dígitos, 0, 1, 2, ..., $r - 1$, y se expresa como una potencia de r según la fórmula general

$$A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \dots + A_1r^1 + A_0r^0$$
$$+ A_{-1}r^{-1} + A_{-2}r^{-2} + \dots + A_{-m+1}r^{-m+1} + A_{-m}r^{-m}$$

$$(312.4)_5 = 3 \times 5^2 + 1 \times 5^1 + 2 \times 5^0 + 4 \times 5^{-1}$$
$$= 75 + 5 + 2 + 0.8 = (82.8)_{10}$$

Sistemas numéricos

- Números decimales

1000's column
100's column
10's column
1's column

$5374_{10} =$

- Números binarios

8's column
4's column
2's column
1's column

$1101_2 =$

Sistemas numéricos

- Números decimales

1000's column
100's column
10's column
1's column

$$5374_{10} = \underbrace{5 \times 10^3}_{\text{five thousands}} + \underbrace{3 \times 10^2}_{\text{three hundreds}} + \underbrace{7 \times 10^1}_{\text{seven tens}} + \underbrace{4 \times 10^0}_{\text{four ones}}$$

- Números binarios

8's column
4's column
2's column
1's column

$$1101_2 = \underbrace{1 \times 2^3}_{\text{one eight}} + \underbrace{1 \times 2^2}_{\text{one four}} + \underbrace{0 \times 2^1}_{\text{no two}} + \underbrace{1 \times 2^0}_{\text{one one}} = 13_{10}$$

Potencias de dos

- $2^0 =$
- $2^1 =$
- $2^2 =$
- $2^3 =$
- $2^4 =$
- $2^5 =$
- $2^6 =$
- $2^7 =$
- $2^8 =$
- $2^9 =$
- $2^{10} =$
- $2^{11} =$
- $2^{12} =$
- $2^{13} =$
- $2^{14} =$
- $2^{15} =$

Potencias de dos

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$

Conversión de números

- Conversión binaria a decimal:
  - Convertir 10011_2 a decimal
- Conversión decimal a binaria:
  - Convertir 47_{10} a binaria

Conversión numérica

- Conversión binaria a decimal:
  - Convertir 10011_2 a decimal
  - $16\times1 + 8\times0 + 4\times0 + 2\times1 + 1\times1 = 19_{10}$

- Conversión decimal a binaria:
  - Convertir 47_{10} a binario
  - $32\times1 + 16\times0 + 8\times1 + 4\times1 + 2\times1 + 1\times1 = 101111_2$

Rango y valores binarios

- Números decimales de N-dígitos
  - ¿Cuantos valores distintos?
  - ¿Rango?
  - Ejemplo: numero decimal de 3-dígitos:

- Números binarios de N-bits
  - ¿Cuantos valores distintos?
  - Rango:
  - Ejemplo: numero binario de 3-bits:

Valores Binarios y Rango

- Numero decimales de N-dígitos
  - ¿Cuantos valores distintos? **10^N**
  - ¿Rango? **$[0, 10^N - 1]$**
  - Ejemplo: numero decimal de 3-dígitos:
    - **$10^3 = 1000$ valores posibles**
    - **Rango: [0, 999]**
- *Numero binario de N*-bit
  - ¿Cuantos valores distintos? **2^N**
  - Rango: **$[0, 2^N - 1]$**
  - Ejemplo: numero binario de 3-bits:
    - **$2^3 = 8$ valores posibles**
    - **Rango: $[0, 7] = [000_2$ al $111_2]$**

Números Octal y Hexadecimales

Como hemos mencionado anteriormente, todos las computadoras y sistemas digitales usan la representación binaria. Los sistemas de numeración octal (en base 8) y hexadecimal (en base 16) son útiles para representar cantidades binarias indirectamente porque poseen la propiedad de que sus bases son de potencia a 2. Ya que $2^3 = 8$ y $2^4 = 16$, cada dígito octal corresponde a tres dígitos binarios y cada dígito hexadecimal corresponde a cuatro dígitos binarios.

La representación más compacta de números binarios en octal o hexadecimal es mucho más conveniente para las personas que usar cadenas de bits en binario que son tres o cuatro veces más largas. Así, la mayoría de los manuales de computadoras usan números octales o hexadecimales para especificar cantidades binarias. Un grupo de 15 bits, por ejemplo, puede ser representado en el sistema octal con solamente cinco dígitos. Un grupo de 16 bits se puede representar en hexadecimal con cuatro dígitos. La elección entre una representación octal o hexadecimal de números binarios es arbitraria, aunque la hexadecimal tiende a ser la más usada, ya que los bits aparecen frecuentemente en grupos de tamaño divisible por cuatro.

$$(127.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = (87.5)_{10}$$

$$(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (46687)_{10}$$

Números Octal y Hexadecimales

Digito Hex	Equivalente Decimal	Equivalente Binario
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
A	10	
B	11	
C	12	
D	13	
E	14	
F	15	

Números Octal y Hexadecimales

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

Conversión Hexadecimal a Binario

- Conversión Hexadecimal a binario:
  - Convertir $4AF_{16}$ (también se escribe como 0x4AF) a binario

- Conversión Hexadecimal a decimal:
  - Convertir 0x4AF a decimal

Conversión hexadecimal a binaria

- Conversión hexadecimal a binario:
  - Convertir $4AF_{16}$ (también se escribe como 0x4AF) a binario

$$0100\ 1010\ 1111_2$$

- Conversión hexadecimal a decimal
  - Convertir $4AF_{16}$ a decimal

$$16^2 \times 4 + 16^1 \times 10 + 16^0 \times 15 = 1199_{10}$$

Tabla resumen

Números con diferentes bases

Decimal (base 10)	Binario (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	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

Bits, Bytes, Nibbles…

- Bits

10010110

most significant bit (Msb)

least significant bit (Lsb)

- Bytes & Nibbles

byte

10010110

nibble

- Bytes

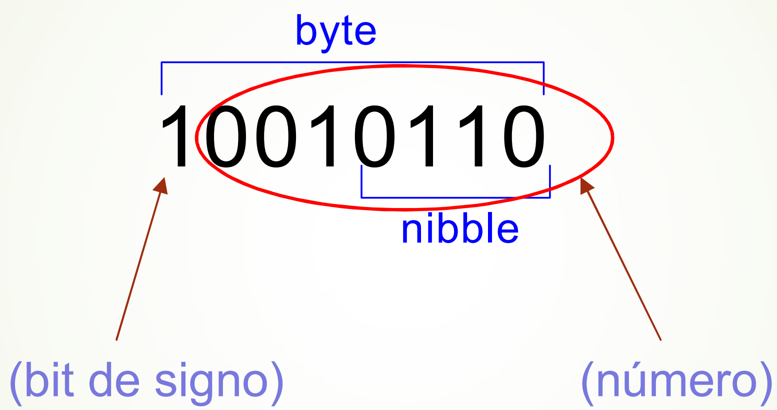
CEBF9AD7

most significant byte

least significant byte

Características del Bytes

- Bytes (8 bits)



- ¿Por qué o qué utilidad tiene esto?

Importantes Potencias de Dos

Recordemos que:

- 2^{10} = 1 kilo $\approx$ 1000 (1024)
- 2^{20} = 1 mega $\approx$ 1 millón (1.048.576)
- 2^{30} = 1 giga $\approx$ Mil millones (1.073.741.824)

¿Recuerda a qué correspondería el Tera?

Estimando una potencia de Dos

- ¿Cual es el valor de 2^{24}?

- ¿Cuantos valores distintos puede representar una variable de 32-bit?

Estimando Potencias de Dos

- ¿Cual es el valor de 2^{24}?

  $2^4 \times 2^{20} \approx$ 16 millones

- ¿Cuantos valores distintos puede representar una variable de 32-bit?

  $2^2 \times 2^{30} \approx$ 4 mil millones

Ejemplo: De decimal a Binario

Convierta el número decimal 41 a binario:

γ

$41/2 = 20 + \mathbf{1}/2$	**Resto = 1**	Dígito menos significativo
$20/2 = 10$	**= 0**	
$10/2 = 5$	**= 0**	
$5/2 = 2 + \mathbf{1}/2$	**= 1**	
$2/2 = 1$	**= 0**	
$1/2 = 0 + \mathbf{1}/2$	**= 1**	Dígito más significativo

$$(41)_{10} = (\mathbf{101001})_2$$

Por supuesto, se puede convertir el número decimal mediante la suma de potencias de dos:

$$(41)_{10} = 32 + 8 + 1 = (101001)_2$$

Ejemplo: De Decimal a Binario (con decimales)

Convierta el número decimal 0.6875 a binario:

$0.6875 \times 2 = \mathbf{1}.3750$	**Entero** $= \mathbf{1}$	Dígito más significativo
$0.3750 \times 2 = \mathbf{0}.7500$	$= \mathbf{0}$	
$0.7500 \times 2 = \mathbf{1}.5000$	$= \mathbf{1}$	
$0.5000 \times 2 = \mathbf{1}.0000$	$= \mathbf{1}$	Dígito menos significativo

$(0.6875)_{10} = (0.\mathbf{1011})_2$

Ejemplo: De decimal a Octal

$153/8 = 19 + \mathbf{1}/8$	**Resto = 1**	Dígito menos significativo
$19/8 = 2 + \mathbf{3}/8$	**= 3**	
$2/8 = 0 + \mathbf{2}/8$	**= 2**	Dígito más significativo

$$(153)_{10} = (\mathbf{231})_8$$

Ejemplo: De decimal a Octal (con decimales)

Convierta el número decimal 0.513 a una fracción octal de tres dígitos:

$0.513 \times 8 = \mathbf{4}.104$	**Entero = 4**	Dígito más significativo
$0.104 \times 8 = \mathbf{0}.832$	**= 0**	
$0.832 \times 8 = \mathbf{6}.656$	**= 6**	
$0.656 \times 8 = \mathbf{5}.248$	**= 5**	Dígito menos significativo

$$(0.513)_{10} = (0.\mathbf{407})_8 .$$

La conversión de números decimales con partes enteras y fraccionarias se realiza convirtiendo cada parte por separado y después combinando los dos resultados. Usando los resultados de los Ejemplos 1-3 y 1-6, obtenemos

$$(153.513)_{10} = (231.407)_8$$

Rangos de los números

En las computadoras digitales, el rango de los números que se pueden representar está basado en el número de bits disponibles en la estructura del hardware que almacena y procesa la información. El número de bits en estas estructuras son normalmente potencias de dos, como 8, 16, 32 y 64. Como el número de bits está predeterminado por las estructuras, la adición de ceros al principio y al final es necesario para representar los números, así el rango de números que pueden ser representados está también predeterminado.

Por ejemplo, para una computadora que procesa enteros sin signo de 16 bits, el número 537 está representado como 0000001000011001. El rango de enteros que pueden ser manejados por esta representación va de 0 a $2^{16} - 1$, eso es de 0 a 65 535. Si la misma computadora procesa fracciones sin signo de 16 bits con el punto binario a la izquierda del dígito más significativo, entonces el número 0.375 está representado por 0.0110000000000000. El rango de fracciones que se puede representar es de 0 a $(2^{16} - 1)/2^{16}$, o de 0.0 a 0.9999847412.

Arquitectura de Computadores

Fundamentos
Operaciones aritméticas en binaria y otros

Suma

- Decimal

$$\begin{array}{r} 3734 \\ +\ \ 5168 \\ \hline \end{array}$$

- Binaria

$$\begin{array}{r} 1011 \\ +\ \ 0011 \\ \hline \end{array}$$

Suma

- Decimal

$$\begin{array}{r} \color{blue}{11} \;\; \color{blue}{\leftarrow \text{carries}} \\ 3734 \\ +\;\; 5168 \\ \hline 8902 \end{array}$$

- Binaria

$$\begin{array}{r} \color{blue}{11} \;\; \color{blue}{\leftarrow \text{carries}} \\ 1011 \\ +\;\; 0011 \\ \hline 1110 \end{array}$$

Ejemplos de Suma Binaria

- Sume los siguientes números binarios de 4-bit (nibble)

$$\begin{array}{r} 1001 \\ +\ 0101 \\ \hline \end{array}$$

- Sume los siguientes números binarios de 4-bit

$$\begin{array}{r} 1011 \\ +\ 0110 \\ \hline \end{array}$$

Ejemplos de Suma Binaria

- Sume los siguientes números binarios de 4-bit

$$\begin{array}{r} 1 \\ 1001 \\ +\ 0101 \\ \hline 1110 \end{array}$$

- Sume los siguientes números binarios de 4-bit

$$\begin{array}{r} 111 \\ 1011 \\ +\ 0110 \\ \hline 10001 \end{array}$$

Desbordamiento!

Desbordamiento

- Los sistemas digitales operan sobre un **numero fijo de bits**
- Desbordamiento (**Overflow**): cuando el resultad es demasiado grande para calzar en los bits disponibles
- Vea el ejemplo previo de 11 + 6

Números Binarios con Signo

- Números con Signo/Magnitud
- Números en complemento de dos

Números con Signo/Magnitud

- 1 bit de signo, *N*-1 bits para magnitud
- Bit de signo es el mas significativo, el bit mas a la izquierda
  - Numero positivo: bit signo = 0
  - Numero negativo: bit signo = 1

$$A : \{a_{N-1}, a_{N-2}, \cdots a_2, a_1, a_0\}$$

$$A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} a_i 2^i$$

- Ejemplo, representación ± 6 con sign/mag de 4 bits:

  +6 =

  - 6 =

- Rango de numero con signo/magnitud de N-bit:

Números con Signo/Magnitud

- 1 bit de signo, N-1 bits para magnitud
- Bit de signo es el mas significativo, el bit mas a la izquierda
  - Numero positivo: bit signo = 0
  - Numero negativo: bit signo = 1

$$A : \{a_{N-1}, a_{N-2}, \ldots, a_2, a_1, a_0\}$$

$$A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} a_i 2^i$$

- Ejemplo, representación ± 6 con sign/mag de 4 bits:

  +6 = **0110**

  - 6 = **1110**

- Rango de numero con signo/magnitud de N-bit: **$[-(2^{N-1}-1), 2^{N-1}-1]$**

Números con Signo/Magnitud

- 2 Problemas:
  - Suma no funciona, por ejemplo -6 + 6:

$$
\begin{array}{r}
1110 \\
+\,0110 \\
\hline
10100
\end{array}
\text{ (error!)}
$$

  - Dos representaciones del 0 (± 0):

    1000

    0000

Números complemento de dos

- No tenemos los problemas de los números con signo/magnitud:
  - Suma funciona
  - Una sola representación para el 0

Números en Complemento de Dos

- Máximo valor negativo tiene el valor de -2^{N-1}

$$A = a_{n-1}\left(-2^{n-1}\right) + \sum_{i=0}^{n-2} a_i 2^i$$

- El mayor numero positivo de 4-bit:
- El numero mas negativo de 4-bit:
- El bit mas significativo aun indica el signo (1 = negativo, 0 = positivo)
- Rango de un numero de *N*-bit en complemento de dos:

Números en Complemento de Dos

- Máximo valor negativo tiene el valor de -2^{N-1}

$$A = a_{n-1}\left(-2^{n-1}\right) + \sum_{i=0}^{n-2} a_i 2^i \qquad \text{(C2)}$$

- El mayor numero positivo de 4-bit: **0111**
- El numero mas negativo de 4-bit: **1000**
- El bit mas significativo aun indica el signo (1 = negativo, 0 = positivo)
- Rango de un numero de N-bit en complemento de dos: $[-(2^{N-1}), 2^{N-1}-1]$

Tomando el complemento de dos

- Invierta el signo del numero en complemento de dos

  Método:

  1. Invertir los bits
  2. Sume 1

- Ejemplo: Invertir el signo de $3_{10} = 0011_2$

Tomando complemento de dos

- Invertir el signo del numero en complemento de dos
- Método:
  1. Invertir los bits
  2. Sume 1
- Ejemplo: Invierta el signo de $3_{10} = 0011_2$

1. 1100

2. + 1

$1101 = -3_{10}$

- Compruebe con la formula (C2)

Ejemplos de Complemento de Dos

- Tome el complemento de dos de $6_{10} = 0110_2$

- ¿Cual es el valor decimal de 1001_2?

Ejemplos de Complemento de Dos

- Tome el complemento de dos de $6_{10} = 0110_2$
  1. 1001
  2. \+ 1

  $1010_2 = -6_{10}$

- ¿Cual es el valor decimal del numero en complemento de dos 1001_2?
  1. 0110
  2. \+ 1

  El $0111_2 = 7_{10}$, luego $1001_2 = -7_{10}$

Suma de Complemento de dos

- Sume 6 + (-6) con números en complemento de dos

$$\begin{array}{r} 0110 \\ +\ \ 1010 \\ \hline \end{array}$$

- Sume -2 + 3 con numero en complemento de dos

$$\begin{array}{r} 1110 \\ +\ \ 0011 \\ \hline \end{array}$$

Suma de Complementos de Dos

- Sume 6 + (-6) con números en complemento de dos

$$\begin{array}{r} {\color{blue}111} \\ 0110 \\ +\quad 1010 \\ \hline {\color{gray}1}0000 \end{array}$$

- Sume -2 + 3 con números en complemento de dos

$$\begin{array}{r} {\color{blue}111} \\ 1110 \\ +\quad 0011 \\ \hline {\color{gray}1}0001 \end{array}$$

Multiplicación Binaria

Multiplicando:	1011
Multiplicador:	× 101
	1011
	0000
	1011
Producto:	110111

Multiplicación Octal

Realice la multiplicación $(762)_8 \times (45)_8$:

Octal
7 6 2
4 5
4 6 7 2
3 7 1 0
4 3 7 7 2

Octal		Decimal		Octal
5×2	=	$10 = 8+2$	=	**12**
$5 \times 6 + 1$	=	$31 = 24 + 7$	=	**37**
$5 \times 7 + 3$	=	$38 = 32 + 6$	=	**46**
4×2	=	$8 = 8 + 0$	=	**10**
$4 \times 6 + 1$	=	$25 = 24 + 1$	=	**31**
$4 \times 7 + 3$	=	$31 = 24 + 7$	=	**37**

Código ASCII

Código ASCII para caracteres

El código estándar para caracteres alfanuméricos se llama ASCII (Código estandarizado americano para intercambio de información, *American Standard Code for Information Interchange*). Usa siete bits para codificar 128 caracteres, según se muestra en la Tabla 1-5. Los siete bits del código se indican como B_1 hasta B_7, donde B_7 es el bit más significativo. Note que los tres bits más significativos del código determinan la columna y los cuatro bits menos significantes la fila de la tabla. La letra A, por ejemplo, es representada en ASCII por 1000001 (columna 100, fila 0001). El código ASCII contiene 94 caracteres que pueden ser imprimidos y 34 caracteres no imprimibles usados para varias funciones de control. Los caracteres imprimibles consisten en 26 letras mayúsculas, 26 letras minúsculas, 10 cifras y 32 caracteres especiales imprimibles como %, @, y $.

Código ASCII

American Standard Code for Information Interchange (ASCII)

	$B_7B_6B_5$							
$B_4B_3B_2B_1$	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Código ASCII

Caracteres de control:

NULL	NULL	DLE	Data link escape
SOH	Inicio del cabecero	DC1	Control de dispositivo 1
STX	Inicio del texto	DC2	Control de dispositivo 2
ETX	Fin del texto	DC3	Control de dispositivo 3
EOT	Fin de la transmisión	DC4	Control de dispositivo 4
ENQ	Petición	NAK	Acknowledge negativo
ACK	Confirmación	SYN	Espera Síncrona
BEL	Timbre	ETB	Fin del bloque de transmisión
BS	Retroceso	CAN	Cancelar
HT	Tab. horizontal	EM	Fin del medio
LF	Line feed	SUB	Sustituir
VT	Tab. vertical	ESC	Escape
FF	Form feed	FS	Separador de fichero
CR	Retorno de carro	GS	Separador de grupo
SO	Desplazamiento hacia fuera	RS	Separador de registro
SI	Desplazamiento hacia dentro	US	Separador de unidad
SP	Espacio	DEL	Borrar