

Instituto Industrial Luis. A. Huergo

Alumno: Agustín Granes

Profesor: Ana Evangelina Salvia

Carpeta Tecnología Control (Teoría)

**Sistemas de Numeración**

Un “sistema de numeración” o “Sistema numeral” es el grupo conjunto de símbolos y formas que con la ayuda de ciertas reglas (Dependiendo del sistema) consigue formar, representar y manipular entidades numéricas ya que cada sistema decide de qué manera funcionan cada uno de sus símbolos y que estructuras es posible formar. De esta manera es necesario poder distinguir a simple vista, esto se consigue según su nomenclatura o según su base lo cual es el número máximo de símbolos diferentes que se pueden usar en el sistema.

Por ejemplo, en sistema de base 10 (Decimal), se utilizan 10 símbolos diferentes: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

Existen dos tipos de clasificación de los sistemas de numeración, Sistemas aditivos y posicionales:

* **Sistemas Aditivos:** En este sistema se obtiene el valor total del número al realizar una adición consecutiva de los símbolos que este incluya. Debido a que el valor de cada símbolo es siempre el mismo esta operación no es más que una simple suma.

**Ejemplo:**

En el sistema romano, el más conocido de este grupo, posee valores representado con su abecedario: l = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1000 … En algunos sistemas el orden de los símbolos altera su positividad o negatividad, como en el caso del romano si un número es precedido por uno menor el segundo será negativo.

Como este existen muchos otros sistemas famosos, como el Maya, sin embargo, estos no son comúnmente utilizados.

* **Sistemas Posicionales:** En este sistema se obtiene el valor total de un símbolo dependiendo de su posición en el número. En este sistema cada posición representa una potencia de la base, iniciando de derecha y hacia la izquierda, el primero es el digito al que se le asigna la potencia de 0 (es decir, 1), al segundo el de la potencia 1 y de esta manera se prosigue hasta el símbolo más a la izquierda. Los símbolos se multiplican por la potencia correspondiente y luego se suman (No pueden hacer símbolos negativos).

**Ejemplo:**

El sistema decimal es el más conocido y también el usado por la población argentina, pero existen muchos otros como, el Binario, Octal, Hexadecimal, etc.

1. **Binario:** Posee los símbolos 0 y 1.
2. **Octal:** Posee los símbolos del 0 al 7.
3. **Hexadecimal:** Posee los símbolos del 0 al 9 y de la “A” a la “F”.

|  |  |  |
| --- | --- | --- |
| **Caracteriasticas** | **Sistema Aditivo** | **Sistema Posicional** |
| Definición | Suma de valores de símbolos | El valor depende de la posición del símbolo |
| Ejemplos | Sistema Romano | Decimal, Binario |
| Valor de Símbolos | Fijo | Varía según la posición |
| Operaciones | Sencillas adecuada para (Números pequeños) | Permite cálculos complejos y eficientes |

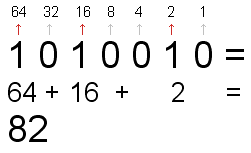
### Binario

Este sistema posee base 2 ya que utiliza solo dos valores, 0 y 1. En este se aprecia cómo los dígitos varían entre ambos símbolos cada , siendo "" la posición del símbolo, contando desde la derecha y empezando en 0.

El sistema binario es esencial para el funcionamiento de las computadoras modernas y la tecnología digital. Su simplicidad y eficacia permiten que se realicen cálculos complejos y se almacene información de manera confiable. Además, cada bit en el sistema binario representa una unidad de información, y múltiples bits se combinan para formar bytes y estructuras de datos más complejas.

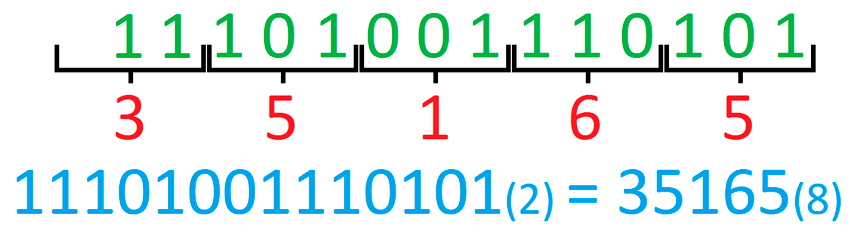
**Conversión de Binario a Decimal:**

Para convertir un número binario a decimal, se debe analizar cada bit en función de su posición, que representa una potencia de 2. Se comienza desde la derecha, asignando potencias de 2 que van desde hacia la izquierda. Por ejemplo, para el número binario 1011, se asigna el valor correspondiente a cada bit: el primer 1 está en la posición 3 ( ), el 0 en la posición 2 (), el siguiente en la posición 1 () y el ultimo 1 en la posición 0 (). Finalmente se suman todos los valores: , por lo tanto el numero binario 1011 equivale a 11 en decimal.



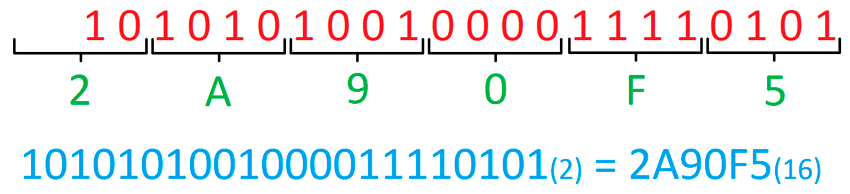
**Conversión de Binario a Octal:**

La conversión de binario a octal implica agrupar los bits en grupos de tres, comenzando desde la derecha, y luego convertir cada grupo a su equivalente en octal. Si el número de bits no es múltiplo de tres, se pueden añadir ceros a la izquierda para completar el grupo. Por ejemplo, el número binario 101101 se agrupa como 1 011 011, que se convierte en 001 011 011 al añadir ceros a la izquierda. Luego, se convierte cada grupo: 001 es 1, 011 es 3 y 011 es 3. Por lo tanto, el número binario 101101 equivale a 133 en octal. Estos grupos se forman debido que para poder formar 8 combinaciones de símbolos se necesita por lo menos 3 variables distintas.



**Conversión de Binario a Hexadecimal:**

Para convertir un número binario a hexadecimal, se agrupan los bits en grupos de cuatro, comenzando desde la derecha. Al igual que en la conversión a octal, si es necesario, se añaden ceros a la izquierda para completar el último grupo. Luego, se convierte cada grupo a su equivalente hexadecimal. Por ejemplo, para el número binario 11010110, se agrupa como 1101 0110. El grupo 1101 se convierte en 13, que en hexadecimal se representa como “D”, y el grupo 0110 se convierte en 6. Así, el número binario 11010110 equivale a D6 en hexadecimal. Los grupos formados se deben a que para poder conseguir el numero 16 o “F” en binario se necesita de 4 variables distantes.



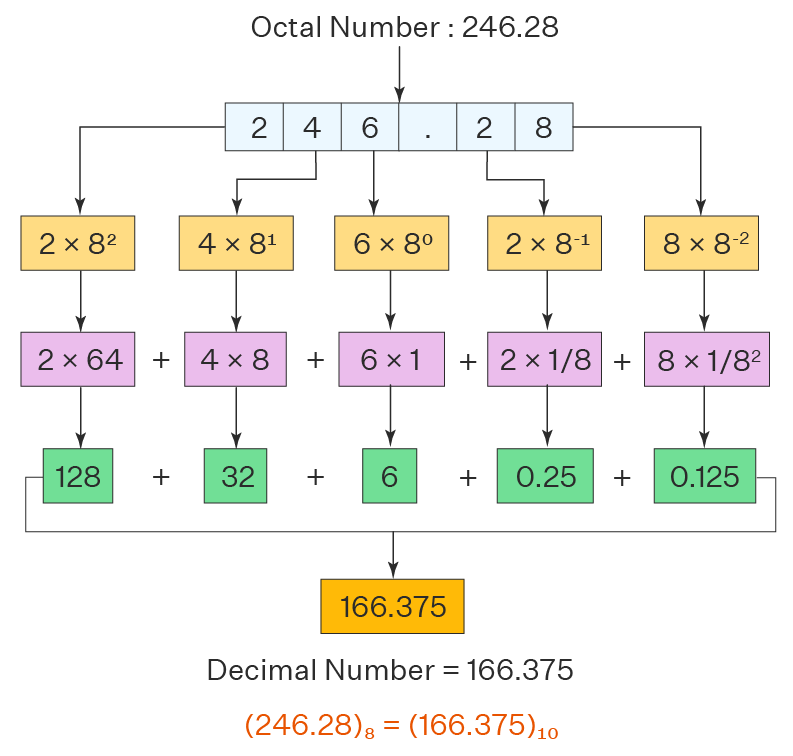
### Octal

Este sistema posee base 8 ya que utiliza solo los valores que se encuentran entre el 0 y 7, es decir, 0, 1, 2, 3, 4, 5, 6, 7. Cuando en este sistema se alcanza el símbolo 7, se procede a colocar un 1 y luego se empieza de nuevo detrás de este. Al alcanzar el número 10 en octal, se reemplaza el 1 por un 2 y se comienza de nuevo. Cada 8 números se le suma uno al que se encuentra más a la izquierda.

El sistema octal proporciona una forma compacta de representar números y es especialmente útil en contextos relacionados con la computación, donde la conversión entre sistemas binarios y octales simplifica el manejo de datos. Por ejemplo, un grupo de tres bits en binario puede ser representado como un solo dígito en octal. Aunque su uso ha disminuido con el auge del sistema hexadecimal, sigue siendo una herramienta valiosa en ciertos ámbitos de la programación y la informática, como en la representación de permisos en sistemas UNIX.

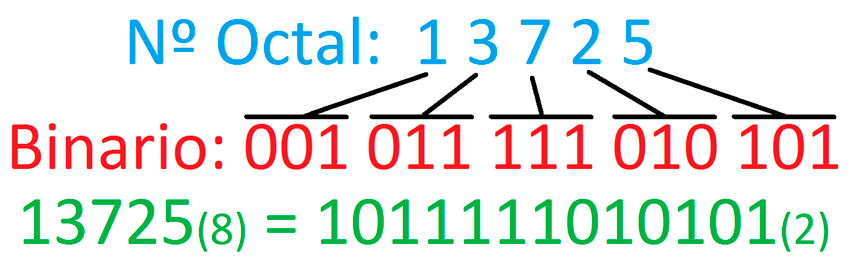
**Conversión de Octal a Decimal:**

Para convertir un número octal a decimal, se utiliza la misma lógica que en la conversión de otros sistemas de numeración. Cada dígito en un número octal representa una potencia de 8, comenzando desde la derecha con . Por ejemplo, para el número octal 237, se asigna el valor correspondiente a cada dígito: el 2 está en la posición 2 (valor ), el 3 está en la posición 1 (valor ) y el 7 está en la posición 0 (valor ). Luego, se suman todos los valores: . Por lo tanto, el número octal 237 equivale a 159 en decimal.



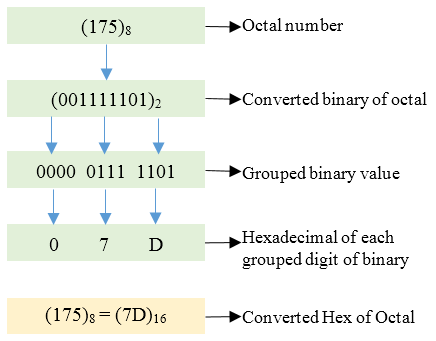
**Conversión de Octal a Binario:**

La conversión de octal a binario se realiza convirtiendo cada dígito octal a su equivalente en binario, utilizando tres bits por dígito. Por ejemplo, el dígito 0 en octal se convierte en 000, el 1 en 001, el 2 en 010, el 3 en 011, el 4 en 100, el 5 en 101, el 6 en 110 y el 7 en 111. Entonces el número octal 257. Cada dígito se convierte así: 2 se convierte en 010, 5 en 101 y 7 en 111. Por lo tanto, el número octal 257 se representa en binario como 010 101 111, que se puede simplificar a 10101111, omitiendo los ceros a la izquierda.



**Conversión de Octal a Hexadecimal:**

Para convertir un número octal a hexadecimal, primero se convierte el número octal a binario y luego ese resultado se convierte a hexadecimal. Por ejemplo, consideremos el número octal 257. Primero, convertimos cada dígito a binario, lo que nos da 010 101 111. Luego, agrupamos los bits en grupos de cuatro desde la derecha: 0010 1011 11. Al añadir ceros a la izquierda para completar el último grupo, obtenemos 0010 1011 1011. Ahora convertimos cada grupo a hexadecimal: 0010 es 2, 1011 es B y 1011 es B. Por lo tanto, el número octal 257 equivale a 2BB en hexadecimal.



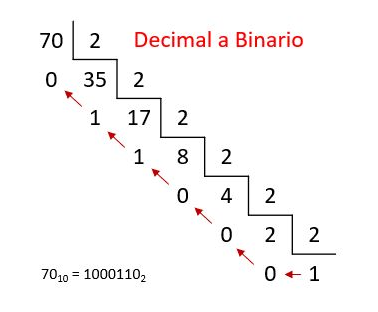
### Decimal

Este sistema posee base 10, por lo que utiliza los símbolos del 0 al 9, es decir, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Es el sistema más comúnmente utilizado en la vida cotidiana y en la mayoría de los contextos matemáticos.

El sistema decimal es más fácil de usar porque es familiar, tiene una estructura clara, permite operaciones sencillas y se apoya en herramientas que lo hacen accesible. Estas características contribuyen a su eficacia en la vida cotidiana. Además, su uso en la mayoría de las actividades diarias, como compras, mediciones y en la educación, lo convierte en un sistema natural para la mayoría de las personas. Su predominancia en la cultura y la economía refuerza su utilidad y relevancia.

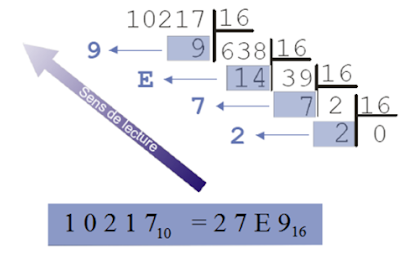
**Conversión de Decimal a Binario:**

Para convertir un número decimal a binario, se utiliza el método de la división sucesiva. Se divide el número decimal entre 2 y se anota el resto. Este proceso se repite con el cociente hasta que el cociente sea cero. Los restos obtenidos se leen en orden inverso para formar el número binario. Por ejemplo, al convertir el número decimal 13 a binario, se realizaría lo siguiente: (resto 1), (resto 0), (resto 1) y (resto 1). Leyendo los restos de abajo hacia arriba, se obtiene 1101, por lo que el número decimal 13 se representa como 1101 en binario.

****

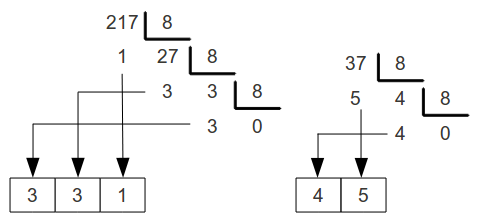
**Conversión de Decimal a Octal:**

Para convertir un número decimal a octal, se aplica un proceso similar al de la conversión a binario, pero en este caso se divide el número decimal entre 8. Se registra el resto en cada paso, que representará un dígito del número octal. Por ejemplo, para convertir el número decimal 159 a octal, se dividiría: (resto 7), (resto 3) y (resto 2). Al leer los restos de abajo hacia arriba, se obtiene 237, por lo que el número decimal 159 se representa como 237 en octal.



**Conversión de Decimal a Hexadecimal:**

La conversión de un número decimal a hexadecimal se realiza mediante la división sucesiva entre 16. Al igual que en los casos anteriores, se registra el resto, que puede ser un número del 0 al 9 o una letra de la “A” a la “F” (donde A=10, B=11, C=12, D=13, E=14, F=15). Por ejemplo, para convertir el número decimal 255 a hexadecimal, se procede de la siguiente manera: (resto 15, que se representa como F), y (resto 15, que también se representa como F). Leyendo los restos de abajo hacia arriba, se obtiene FF. Así, el número decimal 255 se representa como FF en hexadecimal.



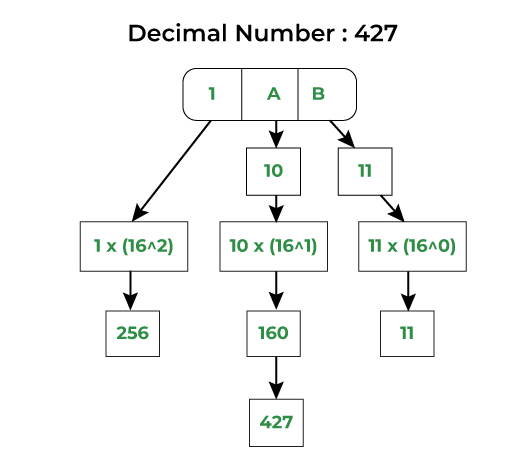
### Hexadecimal

Este sistema posee base 16, ya que utiliza solo los valores que se encuentran entre el 0 y 9, es decir, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, y las letras de la “A” a la “F”; A, B, C, D, E, F. Cuando en este sistema se alcanza el símbolo “F”, se procede a colocar un 1 y luego se empieza de nuevo detrás de este. Al alcanzar el número 1F, se reemplaza el 1 por un 2 y se vuelve a comenzar. Cada 16 números se le suma uno al que se encuentra más a la izquierda.

El sistema hexadecimal es fundamental en la computación y la programación debido a su capacidad para representar datos de manera compacta y legible. Cada dígito hexadecimal representa cuatro bits, lo que significa que es más eficiente para representar grandes valores binarios. Su relación directa con el sistema binario facilita la manipulación de información digital y permite a los programadores trabajar de manera más eficiente con direcciones de memoria y valores de datos. Es comúnmente utilizado en la programación de bajo nivel, diseño de gráficos y en lenguajes de programación como C y C++.

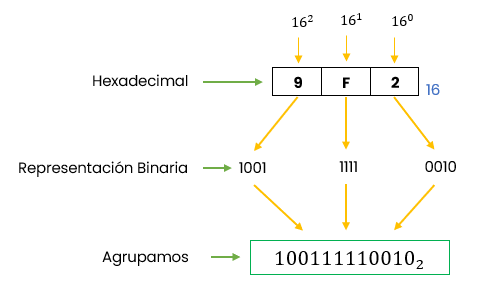
**Conversión de Hexadecimal a Decimal:**

Para convertir un número hexadecimal a decimal, se utiliza la base 16, donde cada dígito tiene un valor que corresponde a una potencia de 16. Se comienza desde la derecha, asignando potencias que van desde hacia la izquierda. Por ejemplo, para el número hexadecimal 2F3, se descompone de la siguiente manera: el 3 está en la posición 0 (valor ), la “F” (que representa 15 en decimal) está en la posición 1 (valor ) y el 2 está en la posición 2 (valor ). Al sumar estos valores, tenemos . Así, el número hexadecimal 2F3 se convierte en 755 en decimal.



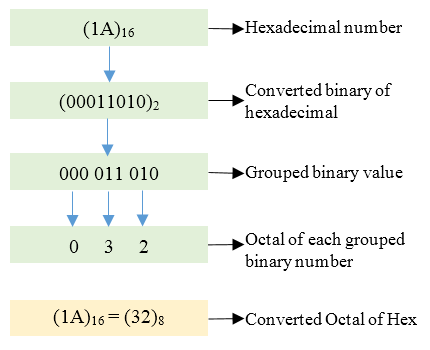
**Conversión de Hexadecimal a Binario:**

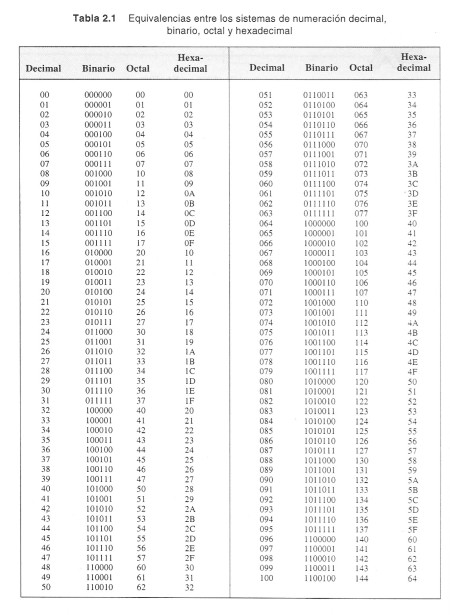
La conversión de hexadecimal a binario es bastante directa, ya que cada dígito hexadecimal se puede representar con un grupo de cuatro bits. Para convertir, simplemente se reemplaza cada dígito hexadecimal por su equivalente en binario. Por ejemplo, para el número hexadecimal 2F3, se convierte cada dígito así: el 2 es 0010, el “F” es 1111 y el 3 es 0011. Por lo tanto, el número hexadecimal 2F3 se convierte en binario como 0010 1111 0011. Para simplificar, se puede omitir los ceros a la izquierda, resultando en 1011110011.



**Conversión de Hexadecimal a Octal:**

Para convertir un número hexadecimal a octal, el método más eficiente es primero convertir el número hexadecimal a binario, y luego convertir el resultado binario a octal. Utilizando el 2F3, primero lo convertimos a binario, resultando en 0010 1111 0011. A continuación, agrupamos los bits en grupos de tres, comenzando desde la derecha: 00 101 111 001 1, que se puede completar a 000 010 111 100 011. Ahora, convertimos cada grupo de tres bits a su equivalente octal: 0000 es 0, 0010 es 2, 1111 es 7, 0100 es 4 y 1011 es 3. Por lo tanto, el número hexadecimal 2F3 se convierte en 02743 en octal.





Ejercitación

1)

a) 10111₂

1·2⁰ + 1·2¹ + 1·2² + 0·2³ + 1·2⁴ =

1 + 2 + 4 + 0 + 16 =

23₁₀

2)

a) FFE₂₁₆

1111 - 1111 - 1110 - 0010

1111111111100010₂

3)

a) 1101111101011010₂

D + F + 5 + A

4F5A₁₆

4)

a) 10AC₁₆

1 – 0000 – 1010 - 1100

1000010101100₂

5)

a) 13725₈

1 - 011 - 111 - 110 - 010 - 101

1011111110010101

5 – D – 7 – 1

17D5₁₆

b) AF2₁₆

2·16⁰ + F·16¹ + A·16² =

2 + 240 + 2560 =

2802₁₀

b) F400₁₆

1111 - 0100 - 0000 - 0000

1111010000000000₂

b) 1001100001110₂

E + 0 + 3 + 1

130E₁₆

b) F4A2₁₆

1111 – 1010 – 0001 - 0010

1111101000010010₂

b) 165₈

1 - 110 - 101

01110101

5 - 7

75₁₆

c) 347₈

7·8⁰ + 4·8¹ + 3·8² =

7 + 32 + 192 =

231₁₀

c) 3E1C₁₆

0011 - 1110 - 0001 - 1100

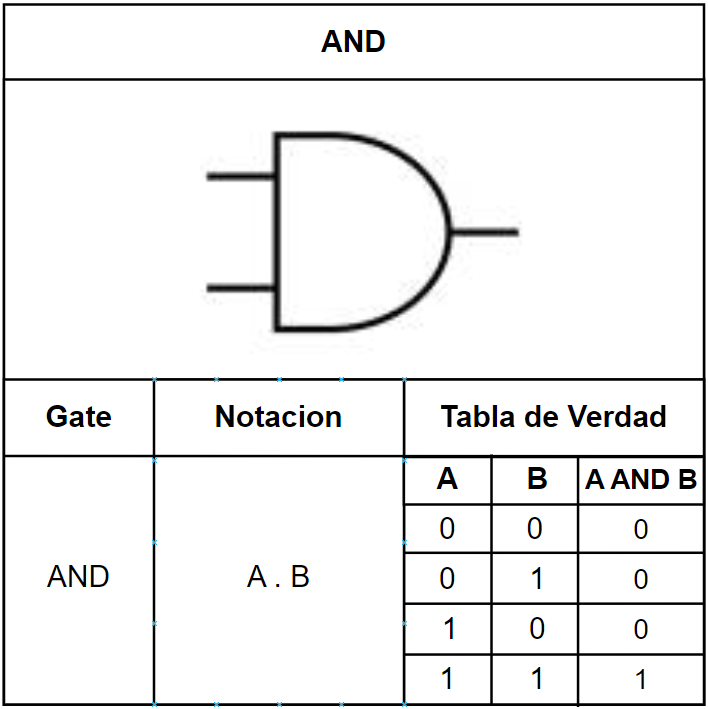
0011111000011100₂

**Lógica Booleana**

La lógica booleana es un sistema de algebra que se utiliza en matemáticas, informática y electrónica. Se basa en valores booleanos, que solo pueden ser True (1) o False (0). Esta lógica es fundamental en el diseño de circuitos digitales y en la programación. Las **compuertas** o **Gates** son los bloques básicos de cualquier circuito digital. Todos los aparatos digitales, desde el dispositivo más simple hasta el computador más sofisticado, están formados por compuertas conectadas en una gran variedad de configuraciones.

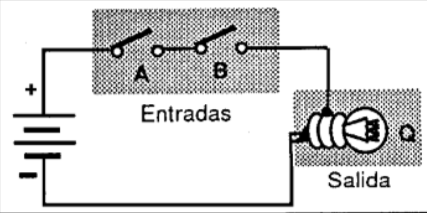
Una compuerta digital es un circuito electrónico con dos o más líneas de entrada y una línea de salida, y tiene la capacidad de tomar decisiones. La decisión de una compuerta de situar su salida en 0 o en 1 depende del estado de sus entradas y de la función lógica para la cual ha sido diseñada. Existen varios tipos de compuertas fundamentales en la lógica booleana:

### 1. Compuertas AND de dos entradas (“Y”)

Una compuerta AND de dos entradas es un dispositivo lógico que entrega una salida altacuando todas sus entradas son altasy una salida bajacuando hay un bajoen cualquiera de sus entradas.

En la figura de la derecha se muestran el símbolo lógico, la ecuación lógica y la tabla de verdad de una compuerta AND de dos entradas.

**En una compuerta se percibe como “A” a la primera entrada, “B” a la segunda y así sucesivamente. “Q” Es la letra que se le asigna a la salida. En esta compuerta la ecuación es: “Q = A . B”, sin embargo el símbolo ( . ) se puede omitir, quedando así “Q = AB”**

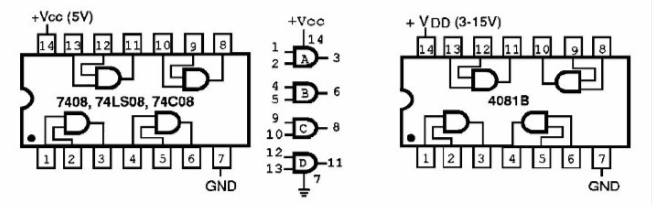
**La operación de una compuerta AND es análoga a la del circuito representado en la figura de la derecha.**

**Puesto que A y B están en serie, la lámpara Q sólo se enciende cuando ambos interruptores están cerrados y permanece apagada mientras cualquiera de los interruptores, o ambos, estén abierto.**

**Un interruptor cerrado se asimila a un nivel alto o 0 lógico y un interruptor abierto a un nivel bajo o 1 lógico.**

**Circuitos integrados con compuertas AND de dos entradas:**

Los circuitos integrados (ICs) como el **7408**, **74LS08** (familia **TTL**), **74C08**, y **4081B** (familia **CMOS**) contienen múltiples compuertas **AND** de dos entradas en un solo chip. Estos ICs son componentes comunes en el diseño de circuitos digitales, ya que permiten implementar la función lógica AND sin necesidad de construir cada compuerta individualmente con transistores o interruptores.

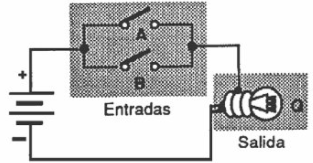
****

**Cada uno de estos dispositivos contiene cuatro compuertas AND de dos entradas, completamente independientes, en una misma cápsula de 14 pines. Todas comparten el mismo voltaje de alimentación. En este circuito el pin 7 se destina a tierra y el 14 a potencia.**

### 2. Compuertas OR de dos entradas (“O”)

### Una compuerta OR es un dispositivo digital que entrega una salida baja cuando todas sus entradas son bajas, y una salida alta cuando existe por lo menos un alto en cualquiera de sus entradas o en las dos al mismo tiempo.

### **En la figura siguiente se muestran el símbolo lógico, la ecuación lógica y la tabla de verdad de una compuerta OR de dos entradas.**

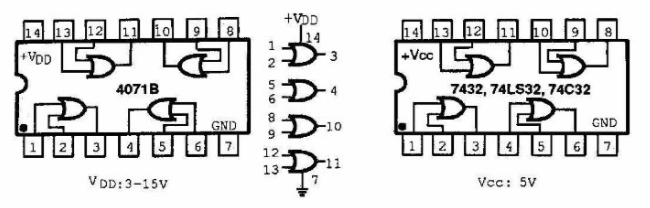
**La expresión "A + B = Q" simboliza la ecuación de esta compuerta. El signo (+) denota la función propia de una compuerta OR y no se puede omitir. Tampoco debe confundirse con el sigo "más" de la suma aritmética.**

**La operación de una compuerta OR es análoga a la del circuito siguiente.** **Los interruptores A y B representan las entradas de la compuerta y la lámpara Q su salida.**

**Debido a que los interruptores están en paralelo, la lámpara Q sólo se apagará cuando ambos interruptores A y B estén abiertos y permanecerá encendida mientras cualquiera de los interruptores, o ambos, estén cerrados.**

**Circuitos integrados con compuertas OR de dos entradas:**

**Los circuitos integrados 7432, 74LS32 (familia TTL) y 74C32 y 4071B (familia CMOS) contienen varias compuertas OR de dos entradas en un solo chip. Al igual que los ICs de compuertas AND, estos circuitos integrados son muy utilizados en la construcción de sistemas digitales porque permiten implementar la función lógica OR de manera sencilla y eficiente.**

****

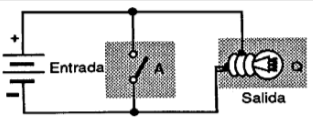
**Cada uno de estos dispositivos trae cuatro compuertas OR de dos entradas completamente independientes, en una misma cápsula de 14 pines. Todas comparten el mismo voltaje de alimentación. En este circuito el pin 7 se destina a tierra y el 14 a potencia.**

### 3. Compuertas NOT (Inversores)

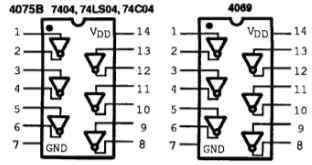
### Una compuerta NOT o inversor es un dispositivo lógico digital con una línea de entrada y una línea de salida que entrega una salida alta cuando su entrada es baja y una salida baja cuando su entrada es alta. En otras palabras, un inversor invierte, niega o complementa el nivel lógico de la señal de entrada. Es una de las compuertas más utilizadas.

La ecuación lógica esta denotada por “Q = Ā“. Donde el círculo o burbuja (o) en el símbolo lógico y la barra horizontal (-) en la ecuación lógica denotan el proceso de inversión realizado por esta compuerta.

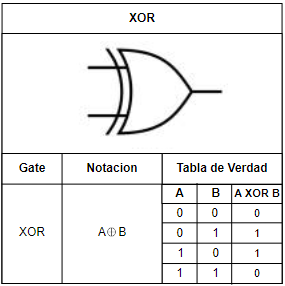
La función lógica realizada por un inversor se denomina inversión o complemento lógico. No existen inversores de dos o más entradas.

La operación de un inversor es análoga a la del circuito eléctrico mostrado en la figura. El interruptor A representa la entrada de la compuerta y la lámpara Q su salida. Debido a que el interruptor está en paralelo con la lámpara Q, esta última se encenderá cuando el interruptor A se abra y se apagará cuando el interruptor se cierre. **Sin embargo, este es solo una representación, ya que al crear este circuito la salida del interruptor debe guiar a tierra. De lo contrario se creará un corto circuito.**

**Circuitos integrados con compuertas NOT:**

 Los circuitos integrados **7404** y **74LS04** (de la familia **TTL**) y **74C04** y **4069B** (de la familia **CMOS**) contienen seis compuertas **NOT** completamente independientes en un solo encapsulado. Estos ICs son ideales para implementar funciones de inversión de señal en sistemas digitales, ya que cada inversor toma una entrada y produce su complemento.

### 4. Compuertas XOR de dos puertas (“O” Exclusivo)

****Una compuerta XOR (o "compuerta OR exclusiva") es un dispositivo digital que entrega una salida alta cuando solo una de sus entradas es alta y la otra es baja; es decir, cuando sus entradas son diferentes. En cambio, si ambas entradas son iguales (ambas altas o ambas bajas), la salida será baja. Esta propiedad hace que la compuerta XOR sea útil para detectar diferencias entre señales de entrada.

En la figura siguiente se muestran el símbolo lógico, la ecuación lógica y la tabla de verdad de una compuerta XOR de dos entradas. La expresión "A ⊕ B = Q" representa la ecuación de esta compuerta, donde el símbolo (⊕) indica la operación XOR. Este símbolo es exclusivo de la función XOR y no debe confundirse con la suma o con la función OR simple.

Ejemplos de circuitos integrados con compuertas XOR son los TTL 7486, 74LS86 y 74LS386; y los CMOS 74C86, 4030B y 4070B. Cada uno de estos dispositivos incorpora 4 compuertas XOR independientes en una misma cápsula de 14 pines.

**Compuertas negativas:**

Además de las compuertas lógicas básicas (AND, OR y NOT), existen las llamadas compuertas lógicas negativas, como la compuerta NAND, la compuerta NOR y la compuerta XNOR. Estas compuertas realizan operaciones similares a sus contrapartes directas (AND, OR y XOR), pero con una salida invertida. Las compuertas negativas son fundamentales en el diseño de sistemas digitales debido a su capacidad para simplificar circuitos y reducir la cantidad de componentes necesarios. Estas compuertas se utilizan ampliamente en la construcción de circuitos lógicos complejos y son esenciales en la arquitectura de procesadores y otros sistemas digitales avanzados.

### 1. Compuertas NAND de dos entradas (NO “Y”)

### Una compuerta NAND de dos entradas es un dispositivo lógico que opera en forma exactamente contraria a una compuerta AND, entregando una salida baja cuando todas sus entradas son altas y una salida alta mientras exista por lo menos un bajo en cualquiera de ellas.

### En la figura se muestran el símbolo lógico, la ecuación lógica y la tabla de verdad de una compuerta NAND de dos entradas. Una compuerta NAND es equivalente a una compuerta AND seguida de un inversor, por lo que su ecuación es representada por un AND, “Q = A . B” pero con la barra de negación por encima como es en la ecuación de la operación NOT.

### En la figura de la derecha puede verse gráficamente esta equivalencia. La operación de una compuerta NAND es análoga a la del circuito eléctrico mostrado en la figura de la derecha. Donde los interruptores A y B están en serie entre sí y en paralelo con la lámpara Q, esta última sólo se apaga cuando ambos interruptores están cerrados y permanece encendida mientras cualquiera de ellos esté abierto.

**Circuitos integrados con compuertas NAND de dos entradas:**

**Los circuitos integrados 7400, 74LS00 (familia TTL), 74C00 y 4011B (familia CMOS) contienen compuertas NAND de dos entradas y están empaquetados en una cápsula estándar de 14 pines. Estos dispositivos integran cuatro compuertas NAND independientes, y comparten una conexión de alimentación y una de tierra. El pin 7 se utiliza para la conexión a tierra (GND), mientras que el pin 14 se conecta a la alimentación (VCC).**

### 

### 2. Compuertas NOR de dos entradas (NO “O”)

### Una compuerta NOR de dos entradas es un dispositivo lógico que opera de forma opuesta a una compuerta OR. Esta compuerta entrega una salida alta únicamente cuando todas sus entradas están bajas; en cualquier otro caso (cuando una o ambas entradas son altas), la salida será baja.

### En la figura de la derecha se muestran el símbolo lógico, la ecuación lógica y la tabla de verdad de una compuerta NOR de dos entradas. El signo (+) y la barra (-) en la ecuación lógica y la burbuja en el símbolo OR confirman esta equivalencia.

### Una compuerta NOR es equivalente a una compuerta OR seguida de un inversor. La operación de una compuerta NOR es análoga a la del circuito eléctrico mostrado en la figura que sigue. Los interruptores A y B representan las entradas de la compuerta y la lámpara Q su salida. Debido a que los interruptores A y B están en paralelo entre sí y con la lámpara Q, esta última sólo se enciende cuando ambos interruptores están abiertos y permanece apagada mientras cualquiera de ellos, o ambos, esté cerrado.

**Circuitos integrados con compuertas NOR de dos entradas:**

### En la figura de la página siguiente se muestra la distribución de pines de los circuitos integrados CMOS 4001B y 74C02 y de los TTL 7402 y 74LS02. Cada uno de estos dispositivos contiene cuatro compuertas NOR de dos entradas completamente independientes en una cápsula de 14 pines. Estas compuertas comparten la misma conexión de alimentación y tierra: el pin 7 se utiliza para la conexión a tierra (GND), mientras que el pin 14 se conecta a la alimentación (VCC).

### 

### 3. Compuertas NXOR de dos entradas (NO “O” Exclusivo):

### Una compuerta NOR exclusiva o XNOR opera en forma exactamente opuesta a una compuerta XOR, entregando una salida baja cuando una de sus entradas es baja y la otra alta, y una salida alta cuando sus entradas son ambas altas o ambas bajas. Es decir, una compuerta XNOR indica, mediante un 1 lógico en su salida, cuándo las dos entradas tienen el mismo estado. Característica que hace ideal su utilización como verificador de igualdad en comparadores y otros circuitos aritméticos.

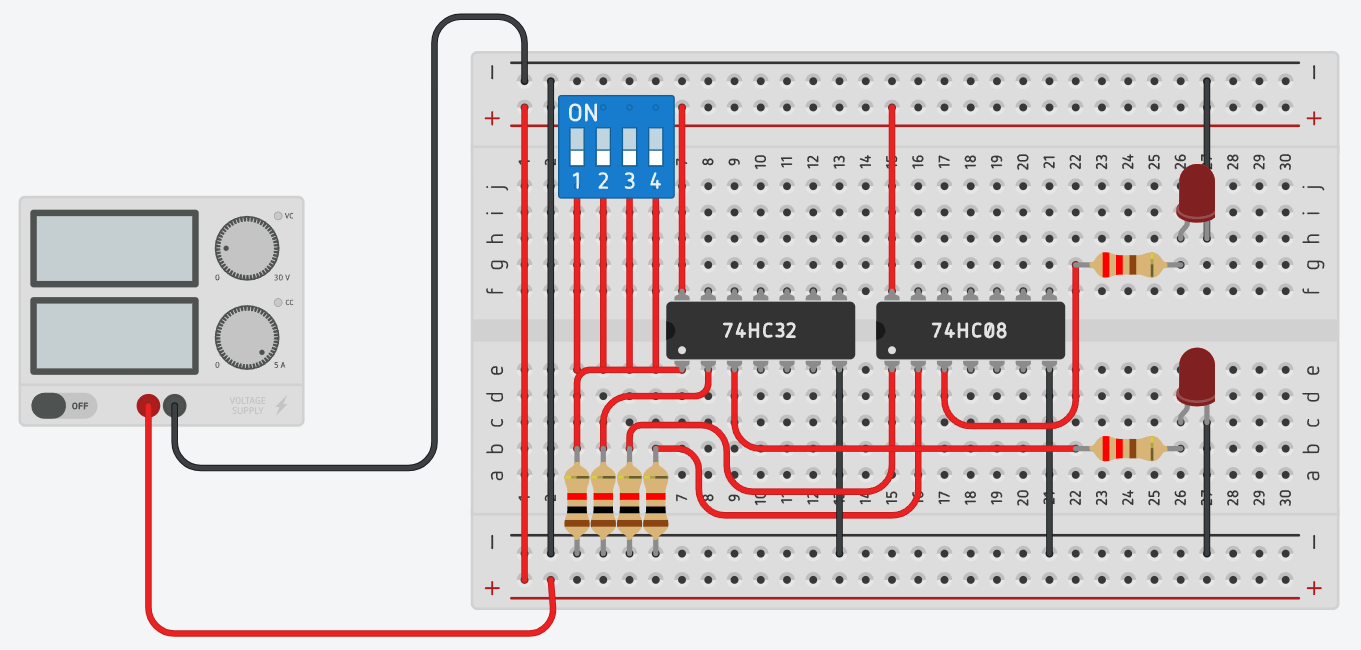
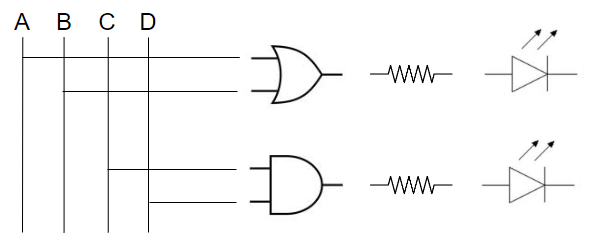
En la figura se muestran el símbolo lógico y la tabla funcional de una compuerta XNOR. una compuerta XNOR es igual a una compuerta XOR seguida de un inversor, por lo que se representa su ecuación igual que una operación XOR, “Q = A ⊕ B”, con la barra de inversión.

Ejercitación

TP B1

Crear un circuito con 4 líneas de corriente asociadas a 4 variable, “A”, “B”, ”C”, ”D”.

Las primeras 2 deberán utilizarse en un circuito integrado con la función de OR lógico y las otras 2 deberán utilizarse con la Función de AND lógico

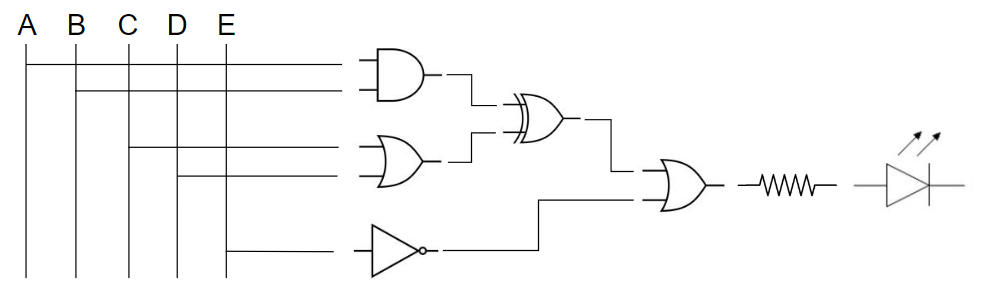


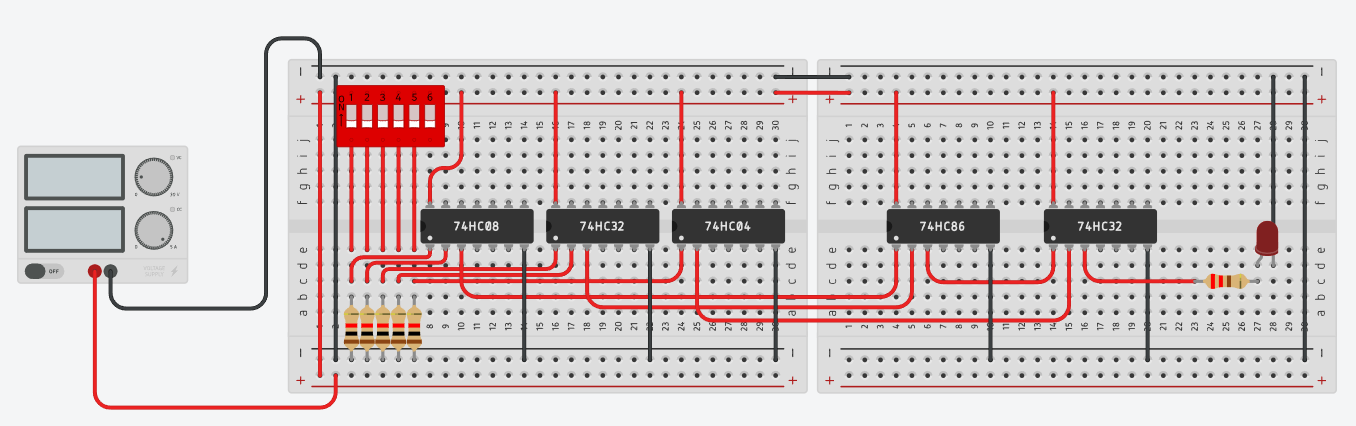
|  |  |  |
| --- | --- | --- |
| A - B - C - D | A + B | C . D |
| 0 - 0 - 0 - 0 | 0 | 0 |
| 0 - 0 - 0 - 1 | 0 | 0 |
| 0 - 0 - 1 - 0 | 0 | 0 |
| 0 - 0 - 1 - 1 | 0 | 1 |
| 0 - 1 - 0 - 0 | 1 | 0 |
| 0 - 1 - 0 - 1 | 1 | 0 |
| 0 - 1 - 1 - 0 | 1 | 0 |
| 0 - 1 - 1 - 1 | 1 | 1 |
| 1 - 0 - 0 - 0 | 1 | 0 |
| 1 - 0 - 0 - 1 | 1 | 0 |
| 1 - 0 - 1 - 0 | 1 | 0 |
| 1 - 0 - 1 - 1 | 1 | 1 |
| 1 - 1 - 0 - 0 | 1 | 0 |
| 1 - 1 - 0 - 1 | 1 | 0 |
| 1 - 1 - 1 - 0 | 1 | 0 |
| 1 - 1 - 1 - 1 | 1 | 1 |

TP B2

Crear un circuito con 5 líneas de corriente asociadas a 5 variable, “A”, “B”, ”C”, ”D”, “E”.

Para este circuito se deberá usar “AND” lógico, dos “OR” lógicos, un “NOT” lógico y un “XOR” lógico, sin tener en cuenta el desperdicio de integrados.



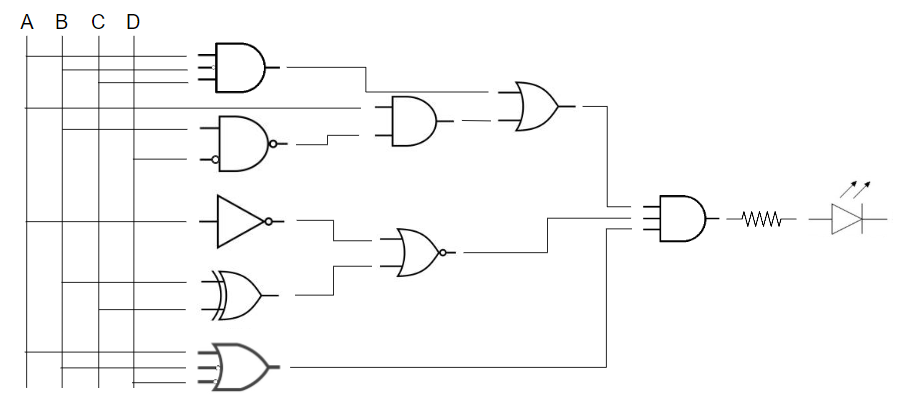


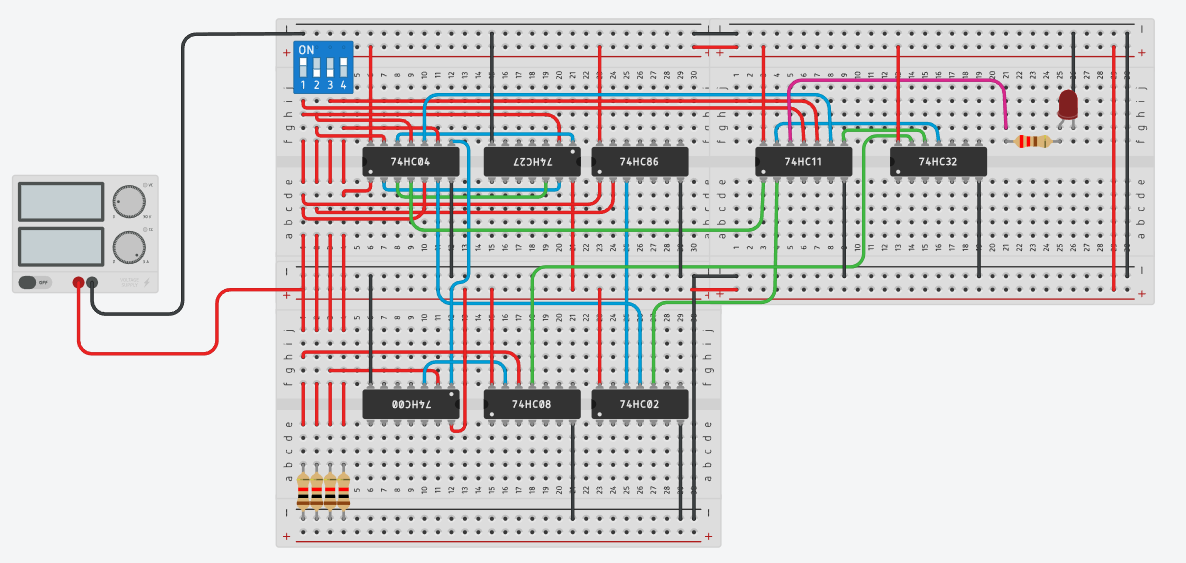
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A - B - C - D - E | A . B | C + D | Ē | (A . B) Ꚛ (C + D) | (A . B) Ꚛ (C + D) + Ē |
| 0 - 0 - 0 - 0 - 0 | 0 | 0 | 1 | 0 | 1 |
| 0 - 0 - 0 - 0 - 1 | 0 | 0 | 0 | 0 | 0 |
| 0 - 0 - 0 - 1 - 0 | 0 | 1 | 1 | 1 | 1 |
| 0 - 0 - 0 - 1 - 1 | 0 | 1 | 0 | 1 | 1 |
| 0 - 0 - 1 - 0 - 0 | 0 | 1 | 1 | 1 | 1 |
| 0 - 0 - 1 - 0 - 1 | 0 | 1 | 0 | 1 | 1 |
| 0 - 0 - 1 - 1 - 0 | 0 | 1 | 1 | 1 | 1 |
| 0 - 0 - 1 - 1 - 1 | 0 | 1 | 0 | 1 | 1 |
| 0 - 1 - 0 - 0 - 0 | 0 | 0 | 1 | 0 | 1 |
| 0 - 1 - 0 - 0 - 1 | 0 | 0 | 0 | 0 | 0 |
| 0 - 1 - 0 - 1 - 0 | 0 | 1 | 1 | 1 | 1 |
| 0 - 1 - 0 - 1 - 1 | 0 | 1 | 0 | 1 | 1 |
| 0 - 1 - 1 - 0 - 0 | 0 | 1 | 1 | 1 | 1 |
| 0 - 1 - 1 - 0 - 1 | 0 | 1 | 0 | 1 | 1 |
| 0 - 1 - 1 - 1 - 0 | 0 | 1 | 1 | 1 | 1 |
| 0 - 1 - 1 - 1 - 1 | 0 | 1 | 0 | 1 | 1 |
| 1 - 0 - 0 - 0 - 0 | 0 | 0 | 1 | 0 | 1 |
| 1 - 0 - 0 - 0 - 1 | 0 | 0 | 0 | 0 | 0 |
| 1 - 0 - 0 - 1 - 0 | 0 | 1 | 1 | 1 | 1 |
| 1 - 0 - 0 - 1 - 1 | 0 | 1 | 0 | 1 | 1 |
| 1 - 0 - 1 - 0 - 0 | 0 | 1 | 1 | 1 | 1 |
| 1 - 0 - 1 - 0 - 1 | 0 | 1 | 0 | 1 | 1 |
| 1 - 0 - 1 - 1 - 0 | 0 | 1 | 1 | 1 | 1 |
| 1 - 0 - 1 - 1 - 1 | 0 | 1 | 0 | 1 | 1 |
| 1 - 1 - 0 - 0 - 0 | 1 | 0 | 1 | 1 | 1 |
| 1 - 1 - 0 - 0 - 1 | 1 | 0 | 0 | 1 | 1 |
| 1 - 1 - 0 - 1 - 0 | 1 | 1 | 1 | 0 | 1 |
| 1 - 1 - 0 - 1 - 1 | 1 | 1 | 0 | 0 | 0 |
| 1 - 1 - 1 - 0 - 0 | 1 | 1 | 1 | 0 | 1 |
| 1 - 1 - 1 - 0 - 1 | 1 | 1 | 0 | 0 | 0 |
| 1 - 1 - 1 - 1 - 0 | 1 | 1 | 1 | 0 | 1 |
| 1 - 1 - 1 - 1 - 1 | 1 | 1 | 0 | 0 | 0 |

TP B3

Crear un circuito con 4 líneas de corriente asociadas a 4 variable, “A”, “B”, ”C”, ”D”.

Para este circuito se tiene como objetivo gastar la menor cantidad posible de integrados con el fin de reducir costos.



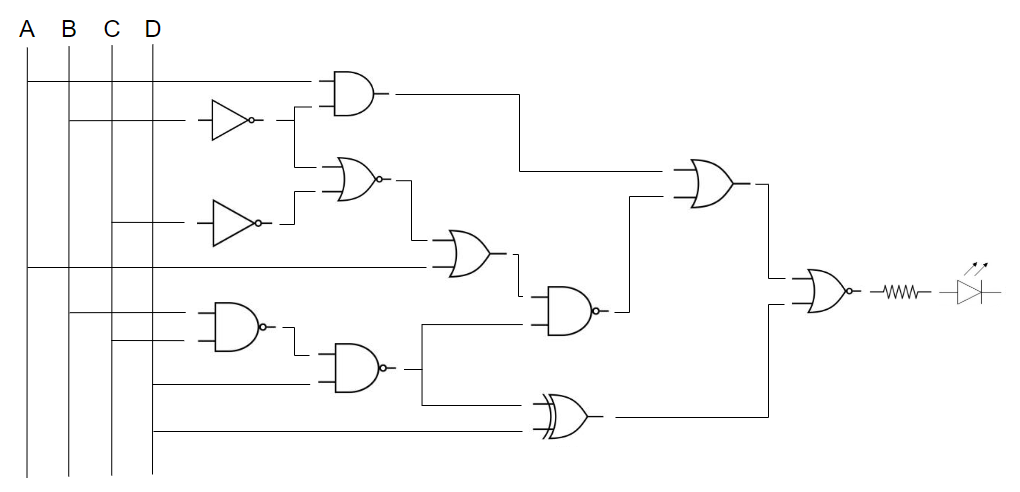


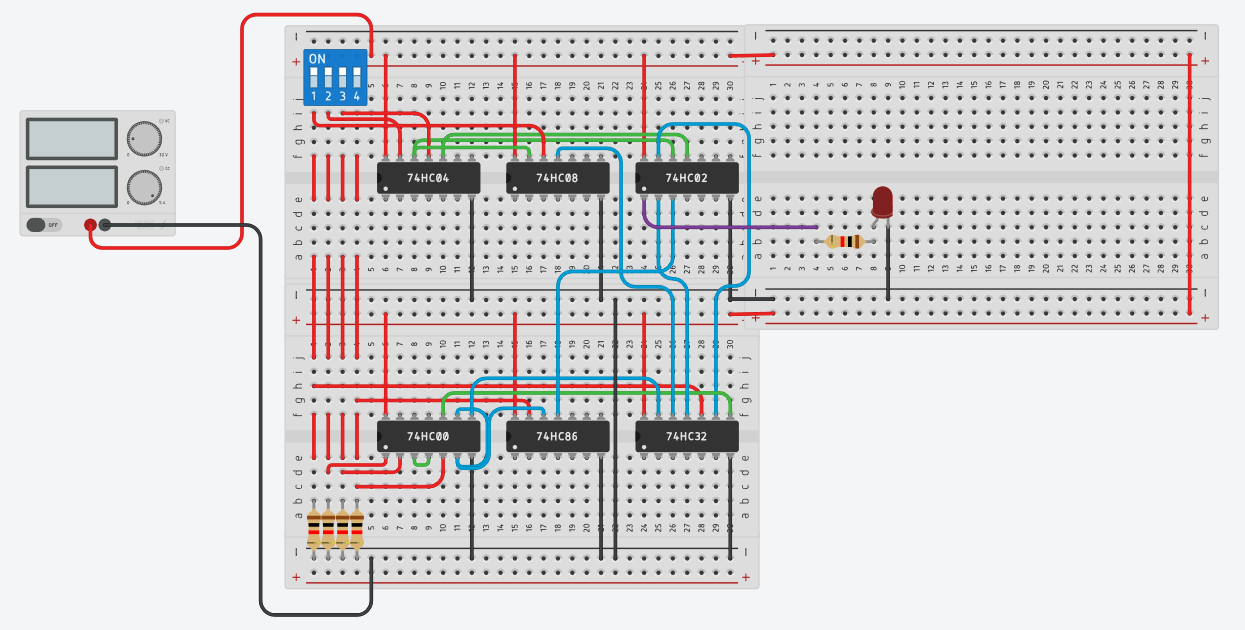
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [(A.B.C) + (A.C.D)] . [A+(BꚚC)] . [A+B+C] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| A + (B Ꚛ C) | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| (A . B . C ) +  A . (C . D) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| A . (C . D) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| A + B + D | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| B Ꚛ C | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| A | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C . D | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| A . B . C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| A - B - C - D | 0 - 0 - 0 - 0 | 0 - 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 - 0 - 0 - 0 | 1 - 0 - 0 - 1 | 1 - 0 - 1 - 0 | 1 - 0 - 1 - 1 | 1 - 1 - 0 - 0 | 1 - 1 - 0 - 1 | 1 - 1 - 1 - 0 | 1 - 1 - 1 - 1 |

TP B4

Crear un circuito con 4 líneas de corriente asociadas a 4 variable, “A”, “B”, ”C”, ”D”.

Para este circuito se tiene como objetivo gastar la menor cantidad posible de integrados con el fin de reducir costos.





|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (A + (B + C)) . (B . C) . D+ (A . B)  + ((B . C) . D Ꚛ D) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| ((B . C) . D) Ꚛ D | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| (A + (B + C)) . (B . C) . D + (A . B) | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| (A + (B + C)) . (B . C) . D | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| (B . C) . D | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| A + (B + C) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| B + C | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| B . C | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| A . B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| A - B - C - D | 0 - 0 - 0 - 0 | 0 - 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 - 0 - 0 - 0 | 1 - 0 - 0 - 1 | 1 - 0 - 1 - 0 | 1 - 0 - 1 - 1 | 1 - 1 - 0 - 0 | 1 - 1 - 0 - 1 | 1 - 1 - 1 - 0 | 1 - 1 - 1 - 1 |

**Teoremas y Propiedades de la lógica de Boole**

Dentro del álgebra de Boole, existen varias propiedades y teoremas que permiten simplificar y manipular expresiones booleanas para obtener resultados de forma más eficiente. Estas propiedades y teoremas son reglas básicas que establecen cómo interactúan entre sí los valores binarios cuando se combinan a través de operaciones. Al aplicar estas reglas, es posible reducir la complejidad de una expresión booleana, haciéndola más fácil de comprender, diseñar o implementar en un circuito lógico.

|  |  |  |
| --- | --- | --- |
| **NOMBRE** | **EXPRESIÓN** | **VISUALIZACIÓN** |
| Elemento Neutro, AND |  |  |
| Elemento Neutro, OR |  |  |
| Idem Potencia |  |  |
| Elemento Inverso |  |  |
| Doble inversión |  |  |
| Conmutativa |  |  |
| Asociativa |  |  |
| Distributiva, AND |  |  |
| Distributiva, OR |  |  |
| Absorción |  |  |
| De Morgan |  |  |

TP B5

Replicar el Circuito N° 3 pero buscando simplificar al máximo el circuito utilizando todas las propiedades del algebra de Boole posibles.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [(A.B.C) + (A.(C+D)] . [A.(BꚚC)] . [A+B+D] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| A . (B Ꚛ C) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| (A . B . C ) +  A . (C + D) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| A . (C + D) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| A + B + D | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| B Ꚛ C | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| C + D | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| A . B . C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| A - B - C - D | 0 - 0 - 0 - 0 | 0 - 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 - 0 - 0 - 0 | 1 - 0 - 0 - 1 | 1 - 0 - 1 - 0 | 1 - 0 - 1 - 1 | 1 - 1 - 0 - 0 | 1 - 1 - 0 - 1 | 1 - 1 - 1 - 0 | 1 - 1 - 1 - 1 |

TP B6

Replicar el Circuito N° 4 pero buscando simplificar al máximo el circuito utilizando todas las propiedades del algebra de Boole posibles.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [(A + (B . C)) . (B . C) + D . (A . B)]  . [((B . C) + D) Ꚛ D] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| ((B . C )+ D) Ꚛ D | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| (A + (B . C)) + (B . C) . D + (A . B) | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| (A + (B . C))  + ((B . C) . D) | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| (B . C )+ D | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| A + (B . C) | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| B . C | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| B . C | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| A . B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| A - B - C - D | 0 - 0 - 0 - 0 | 0 - 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 - 0 - 0 - 0 | 1 - 0 - 0 - 1 | 1 - 0 - 1 - 0 | 1 - 0 - 1 - 1 | 1 - 1 - 0 - 0 | 1 - 1 - 0 - 1 | 1 - 1 - 1 - 0 | 1 - 1 - 1 - 1 |