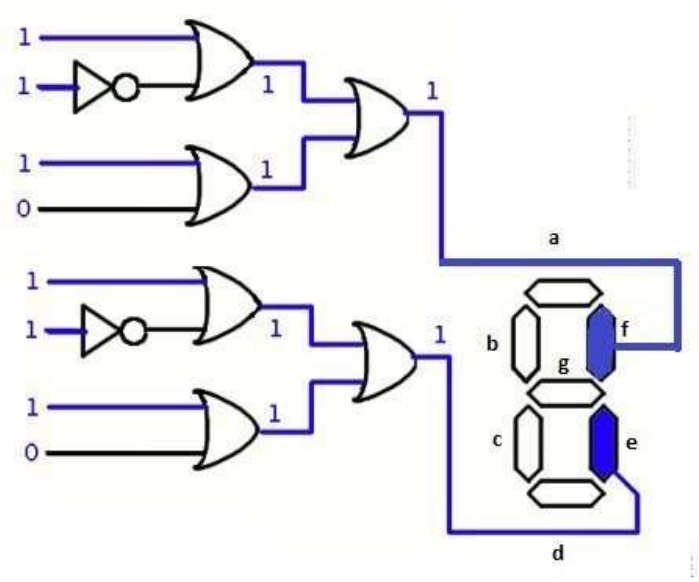
Introducción operaciones binarias



SISTEMAS DE NUMERACIÓN

En programación se utilizan diferentes sistemas de numeración. Los sistemas digitales utilizan la numeración binaria, el programador utiliza el sistema decimal y el sistema hexadecimal (y octal).

El sistema hexadecimal se utiliza por adaptación a la organización de información de los sistemas digitales.

- Sistema **decimal**: Utiliza base 10: 0 1 2 3 4 5 6 7 8 9
- Sistema **binario**: Utiliza base 2: 0 1
- Sistema **octal**: Utiliza base 8: 0 1 2 3 4 5 6 7
- Sistema **hexadecimal**: Utiliza base 16: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Los sistemas digitales utilizan paquetes de 8 *bits* como unidad básica para manejar la información. Estos paquetes se denominan ***bytes:***

1 byte = 8 bits

SISTEMAS DE NUMERACIÓN

La capacidad global de las memorias de datos se mide en bytes. Para expresar cantidades grandes se utiliza un prefijo para cada factor de 1024, o 10 bits:

- 8 bit .....1 byte
- 1024 bytes.....1 kilobyte (Kb)
- 1024 Kb .....1 Megabyte (Mb)
- 1024 Mb.....1 Gigabyte (Gb)
- 1024 Gb .....1 Terabyte (Tb)
- 1024 Tb .....1 Petabyte (Pb)
- 1024 Pb .....1 Exabyte (Eb)

La normativas IEC 80000-13 establece las unidades de manera más estricta.

Las basadas en base 10:
1kbyte=1000 bytes

Las basadas en base 2 tienen la denominación (kibi, Mebi, Gigi,…):
1KiB=1024 bytes
1MiB=1024 KiB=1024 1024 B =

El factor 1024 es equivalente a 10 bits. Es decir, cada 10 bits cambia de prefijo: $2^{10} = 1024$

El sistema hexadecimal se utiliza por comodidad, ya que su base (16) se puede expresar en potencias de 2: con un dígito podemos expresar 4 bits y con dos podemos expresar 1 byte.

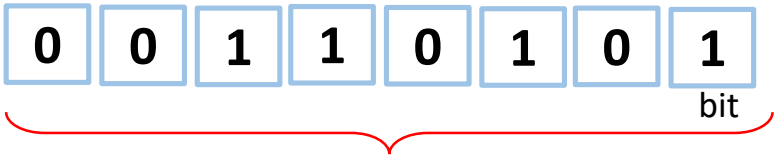
Por ejemplo:

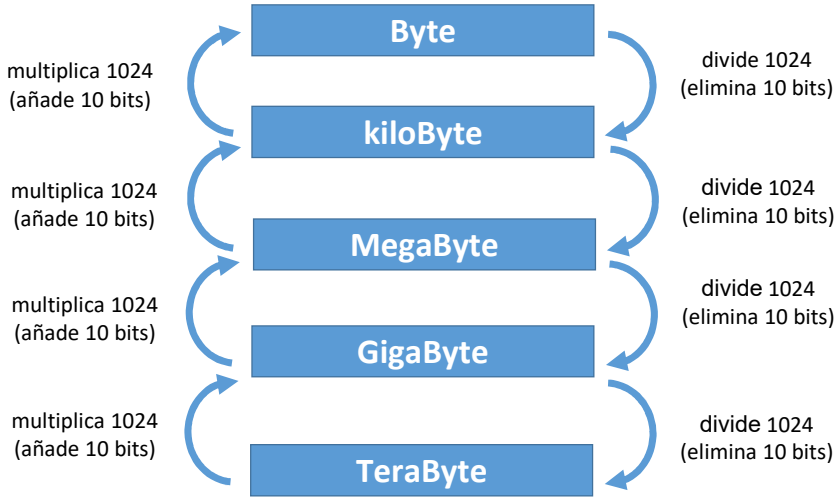
Binario: 1111 1111

Hexadecimal: FF

Decimal: 255

SISTEMAS DE NUMERACIÓN





SISTEMAS DE NUMERACIÓN

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

Decimal	Binario	Hexadecimal
16	**1**0000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14
21	10101	15
22	10110	16
23	10111	17
24	11000	18
25	11001	19
26	11010	1A
27	11011	1B
28	11100	1C
29	11101	1D
30	11110	1E
31	11111	1F

Decimal	Binario	Hexadecimal
32	10 0000	20
63	11 1111	3F
64	**1**00 0000	40
127	111 1111	7F
128	**1**000 0000	80
255	1111 1111	FF
512	**1** 0000 0000	200
1024	**1**0 0000 0000	400
2048	**1**00 0000 0000	800
32768	**1**000 0000 0000 0000	8000
65536	**1** 0000 0000 0000 0000	10000
1048576	**1** 0000 0000 0000 0000 0000	100000

SISTEMAS DE NUMERACIÓN

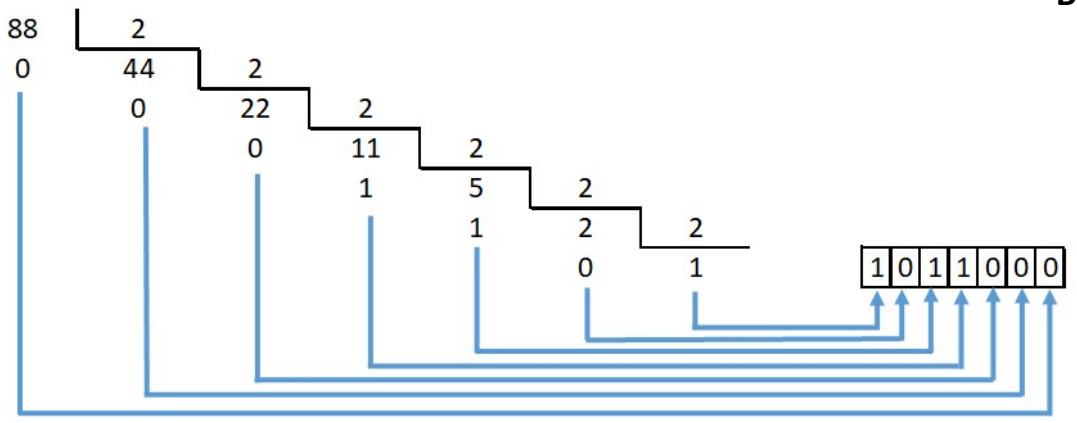
Conversión decimal a binario

Para convertir un número decimal a uno binario:

1- Dividir el número decimal sucesivamente entre dos
2- El resto de la división (0 ó 1) formará parte del dígito de número binario, de manera secuencial

El resultado del número binario toma el cociente final y los restos que van quedando en las sucesivas divisiones de derecha a izquierda:

Decimal: **88**
Binario (byte): **0101 1000**



SISTEMAS DE NUMERACIÓN

Conversión binario a decimal

Procedimiento para transformar un número binario a decimal :

- Numerar los dígitos de derecha a izquierda empezando por cero
- Multiplicar el dígito (0 ó 1) por 2 elevado al número de posición
- Sumar el resultado

Por ejemplo:

Número binario :

0 1 0 1 1 0 0 0

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

$0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$

$0 + 64 + 0 + 16 + 8 + 0 + 0 + 0 = \mathbf{88}$

SISTEMAS DE NUMERACIÓN

Conversión binario a hexadecimal

Los números se representan con dieciséis símbolos alfanuméricos: diez dígitos numéricos y seis caracteres

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F

Los caracteres A, B,C,D,E, F representan las cantidades decimales comprendidas entre 10 y 15.

Procedimiento para la conversión de binario a hexadecimal:

- Dividir el binario en grupos de 4 bits
- Convertir cada grupo a un dígito decimal. La conversión de cada dígito decimal a hexadecimal es directa y nos dará el hexadecimal pedido:

Ejemplo: **0101 1000**

Número hexadecimal: 58

Número decimal: 88

Binario: 0101 1000

SISTEMAS DE NUMERACIÓN

Conversión hexadecimal a binario

Es la conversión más sencilla, esta es la razón por la que se utilizan números hexadecimales. Se trata de convertir a binario cada dígito por separado.

Ejemplo: **9B** a decimal

9 (9 en decimal) ⟶ 1001

B (11 en decimal) ⟶ 1011

1001 1011 (binario)

El decimal es:

$$1\cdot2^7+0\cdot2^6+0\cdot2^5+1\cdot2^4+1\cdot2^3+0\cdot2^2+1\cdot2^1+1\cdot2^0=155$$

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

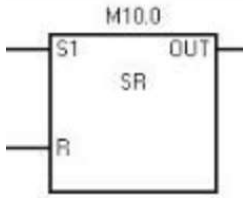
OPERACIONES LÓGICAS

Clases de registros

- **Biestables:**

Es un sistema lógico que tiene únicamente dos posibles estados. **Puede permanecer indefinidamente de manera estable en cualquiera de los dos estados**. Es necesario una acción para el paso de un estado a otro. Eléctricamente puede estar con valor alto (p.e. 5V) o con valor bajo (0V). En lógica se utilizan estados *true* o *false*, o números binarios (1, 0) . Necesita una señal para cambiar de estado.

true /cierto	1	5V
false / falso	0	0V



Biestable flip-flop RS

- **Monoestable:**

Es un sistema lógico que tiene únicamente **un posible estado**. Cuando se aplica una excitación se mantiene en un estado y retorna cuando se deja de aplicar.

- **Bit:** Es una unidad básica que indica el estado de un biestable o un monoestable
- **Registro :** Es un conjunto de biestables o bits

OPERACIONES LÓGICAS

- **Nibble:**

Es un conjunto de 4 bits. Un *nibble* se puede expresar por un dígito hexadecimal.

Por ejemplo:

El *nibble* '1010' es 'A' en hexadecimal

- **Byte:**

Es un conjunto de 8 bits. Un byte se puede expresar por dos dígitos hexadecimales:

Por ejemplo:

El *byte* '1101 0010' es 'D2' en hexadecimal

- **Word:**

Es un conjunto de 16 bits o 2 *bytes*. Un *word* se puede expresar por cuatro dígitos hexadecimales:

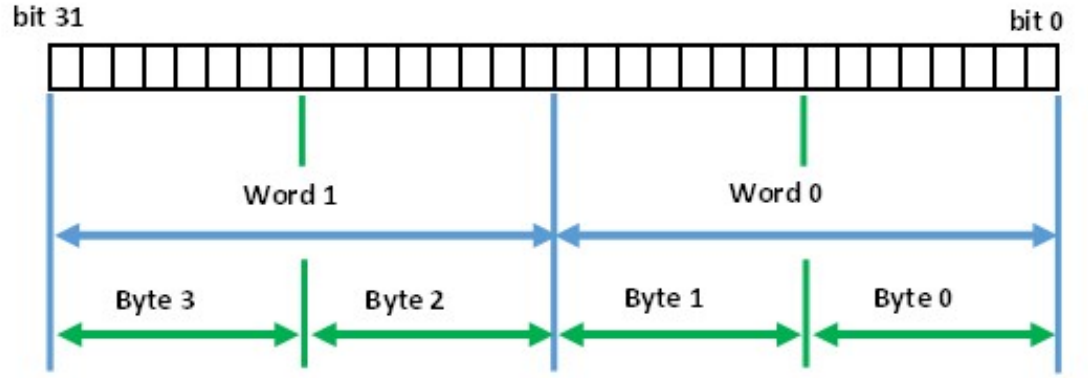
Por ejemplo:

El *word* '0100 1111 0011 1001' es '4F39' en hexadecimal

OPERACIONES LÓGICAS

- **Double Word:**

Es un conjunto de 32 bits o 4 *bytes*. Un *double word* se puede expresar por ocho dígitos hexadecimales:



OPERACIONES LÓGICAS

Álgebra de boole

Álgebra aplicada al razonamiento lógico. Esta basada en dos únicas posibilidades cierto o falso

Variable booleana: toma dos valores (cierto/falso, 0/1, conectado/desconectado)

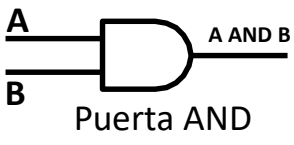
Operaciones booleanas

- Operación AND (&, ·): la salida

será 0 cuando una de las entradas es 0

será 1 cuando las dos entradas valen 1

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1



Puerta AND

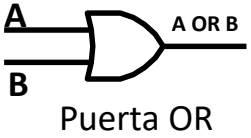
Ejemplo: Maniobra bimanual: Dos interruptores NO en serie en un circuito. Se activa el circuito si los DOS interruptores están pulsados

- Operación OR (+): la salida

será 1 cuando una de las entradas es 1

será 0 cuando las dos entradas valen 0

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



Puerta OR

Ejemplo: Dos interruptores NO en paralelo en un circuito. Se activa el circuito si UNO de los dos interruptores está pulsado.

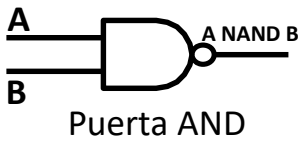
OPERACIONES LÓGICAS

- Operación NAND: la salida

será 1 cuando una de las entradas es 0

será 0 cuando las dos entradas valen 1

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0



Puerta AND

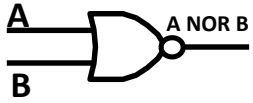
Ejemplo: Dos interruptores NC en paralelo en un circuito. Se desactiva el circuito sólo si los dos interruptores están pulsados.

- Operación NOR: la salida

será 1 cuando las dos entradas valen 0

será 0 cuando una de las entradas es 1

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0



Ejemplo: Doble pulsador de emergencia. Dos interruptores NC en serie en un circuito.
Se desactiva el circuito si se pulsa uno de los dos interruptores

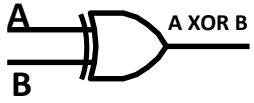
OPERACIONES LÓGICAS

- Operación XOR ($\oplus$,OR exclusiva): la salida

será 1 cuando las dos entradas son diferentes

será 0 cuando las dos entradas son iguales

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



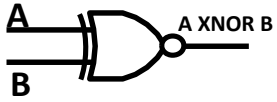
Ejemplo: Se utiliza para hacer sumas de números, bit a bit

- Operación NXOR: la salida

será 1 cuando las dos entradas son iguales

será 0 cuando las dos entradas son diferentes

A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

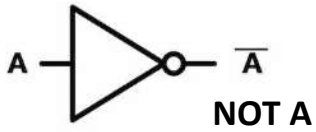


- Operación NOT ($\bar{A}$) : la salida

será 1 cuando la entrada es 0

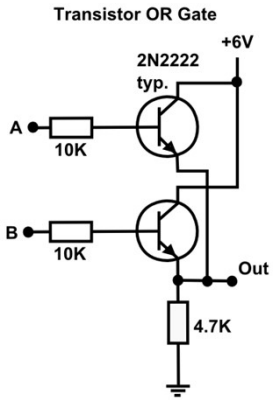
será 0 cuando la entrada es 1

A	NOT A
0	1
1	0

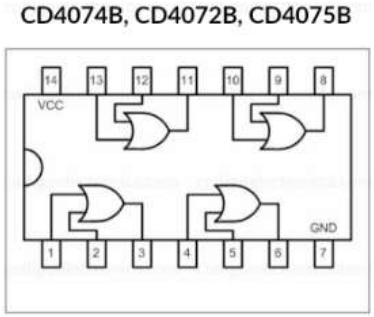


OPERACIONES LÓGICAS

Implementación de una puerta OR mediante transistores BJT

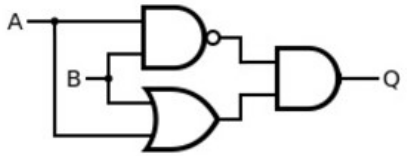


Chips comerciales que contienen puertas OR



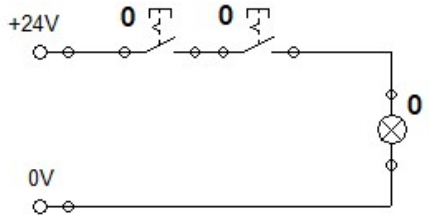
Obtención de puertas OR exclusivas mediante combinaciones de puertas básicas AND, OR. Por ejemplo:

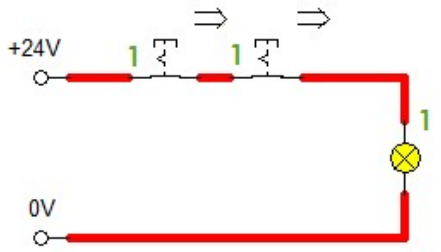
$$A \oplus B = \bar{A} \cdot B + \bar{B} \cdot A$$
$$A \oplus B = \overline{(A \cdot B)} \cdot (A + B)$$



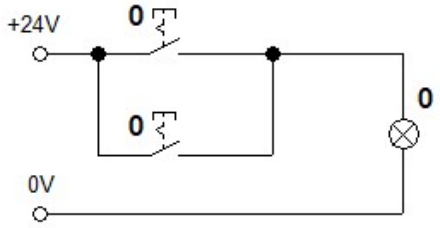
OPERACIONES LÓGICAS

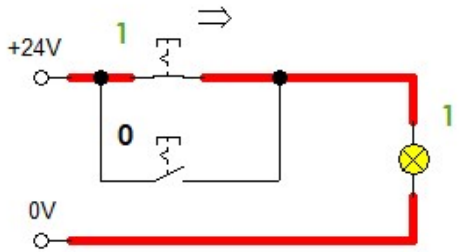
Implementación de puertas mediante interruptores





A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

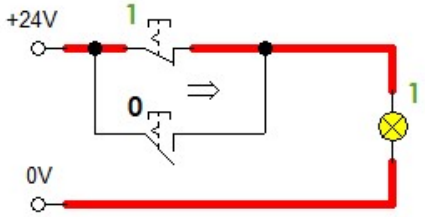


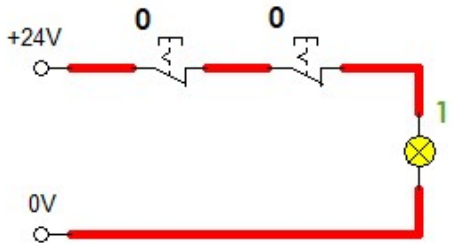


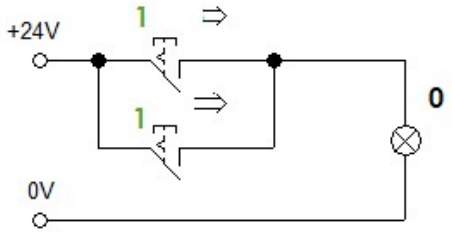
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

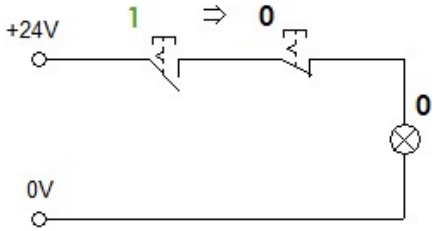
OPERACIONES LÓGICAS

Implementación de puertas mediante interruptores







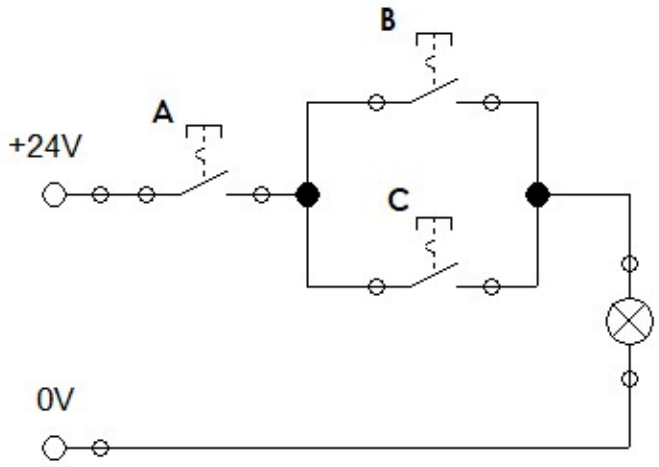


A	B	A NAND B
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

A	B	A NOR B
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

OPERACIONES LÓGICAS

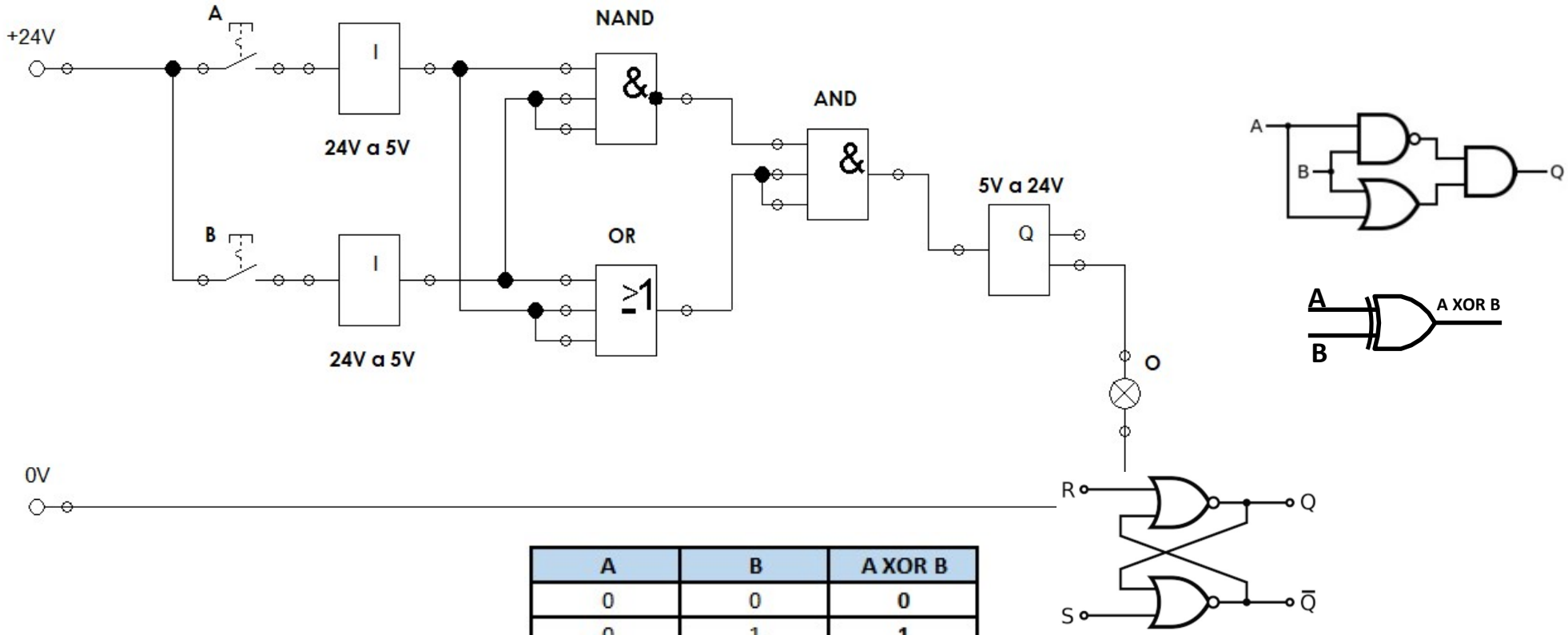
Implementación de puertas mediante interruptores. Tabla de la verdad



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

OPERACIONES LÓGICAS

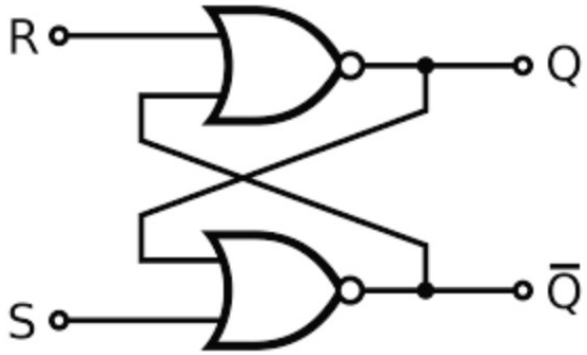
Implementación de puertas XOR. Tabla de la verdad

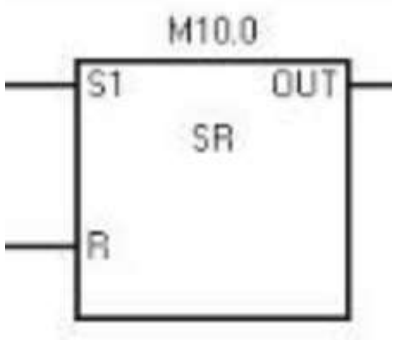


A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

OPERACIONES LÓGICAS

Puerta biestable o circuito multivibrador: flip-flop RS. Mantiene un bit hasta que se *resetea* o *setea*.





R	S	Q
0	0	Q
1	0	0
0	1	1
1	1	NO POSIBLE