1. Sistema binario

2 dígitos posibles → 0, 1 — A esta unidad mínima se le llama **bit**

Al expresar un número binario, el bit + significativo (de + peso) se sitúa a la izquierda y el − significativo a la derecha.

ej: $(1\ 0\ 1\ 1\ 0)_2 = 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 =$
$= 16 + 4 + 2 = (22)_{10}$

bit + significativo ↑ — ↑ bit − significativo

1.1. De binario a decimal

$(10011010)_2 = 2^7 + 2^4 + 2^3 + 2^1 = 128 + 16 + 8 + 2 = (154)_{10}$

1.2 De decimal a binario

$(25)_{10}$

25 | 2
↓ 12 | 2
0 6 | 2
0 3 | 2
↓ 1

$(25)_{10} = (11001)_2$

También:

	1	1	0	0	1	
64	32	16	8	4	2	1

1.3 Operaciones en el sistema binario

→ SUMA BINARIA

ej: $(543)_{10} + (226)_{10} = (769)_{10}$

1	0	0	0	0	0	1	1	1	1
512	256	128	64	32	16	8	4	2	1

1	1	1	0	0	0	1	0	
128	64	32	16	8	4	2	1	

← Acarreo

1 0 0 0 0 1 1 1 1 ← Sumando 1 $(543)_{10}$

\+ 1 1 1 0 0 0 1 0 ← Sumando 2 $(226)_{10}$

1 1 0 0 0 0 0 0 1 $= 2^9 + 2^8 + 2^0 = 512 + 256 + 1$
$= (769)_{10}$

Para sumar y restar necesitamos circuitería física distinta. Para solucionar el problema aplicamos el convenio de complementos que nos permitirá usar la misma circuitería: SUMADORES.

→ RESTA BINARIA

Por lo tanto, para realizar una resta binaria sumaremos números negativos.

Para obtener los números negativos podremos aplicar el complemento a uno o el complemento a dos:

* COMPLEMENTO A UNO DE UN NÚMERO BINARIO:

El complemento a uno de un número se obtiene cambiando '1's por '0's y '0's por '1's

El bit de signo es el bit más significativo de modo que si es '1' es negativo y si es '0' es positivo.

Ejemplo: Hallar el complemento a uno de $(15)_{10}$

0	0	1	1	1	1
32	16	8	4	2	1

$(15)_{10} = (01111)_2$ (bit de signo ⊕)

$(-15)_{10} = (10000)_{2\,C1}$ (bit de signo)

* COMPLEMENTO A DOS DE UN NÚMERO BINARIO:

El complemento a dos de un número se obtiene sumando 1 al complemento a uno de ese número

Ejemplo: Hallar el complemento a dos de $(26)_{10}$

0	0	1	1	0	1	0
64	32	16	8	4	2	1

$(26)_{10} = (011010)_2$ (bit de signo ⊕)

$(-26)_{10} = (100101)_{2\,C1}$

$+\ 1$

$(100110)_{2\,C2}$

Ejemplos

Utilizaremos números de 8 bits.

Si realizamos $26 - 15 = (11)_{10}$

0	0	0	1	1	0	1	0
128	64	32	16	8	4	2	1

$(26)_{10} = (00011010)_2$

0	0	0	0	1	1	1	1
128	64	32	16	8	4	2	1

$(15)_{10} = (00001111)_2$

Complemento a dos $(-15)_{10} = (11110001)_2 C_2$

Complemento a uno $(-15)_{10} = (11110000)_2 C_1$

bit de signo

Resta en complemento a UNO

$$\begin{array}{r} 00011010 \leftarrow (26)_{10} \\ + \; 11110000 \leftarrow (-15)_2 C1 \\ \hline 100001010 \\ \downarrow \\ \text{Acarreo} \rightarrow 1 \\ \hline 1011 = (11)_2 \end{array}$$

Resta en Complemento a DOS

$$\begin{array}{r} 00011010 \\ + \; 11110001 \\ \hline \downarrow \; \underbrace{00001011}_{\text{8 bits}} \\ \text{+ representativo} \end{array} = (11)_2$$

Realizamos ahora $15 - 26 = (-11)_{10}$

C1

$$\begin{array}{r} 00001111 \leftarrow (15)_{10} \\ + \; 11100101 \leftarrow (26)_{10} \\ \hline 0 \; \downarrow 11110100 \leftarrow (\text{Resultado})_2 C1 \end{array}$$

Acarreo → 0; ↓ bit de signo

Resultado = 10001011

signo $= (-11)_{10}$

C2

$$\begin{array}{r} 00001111 \; (15)_{10} \\ + \; 11100110 \leftarrow (26)_{10} \\ \hline \downarrow 11110101 \leftarrow (X)_2 C_2 \\ \text{signo} \end{array}$$

$10001011 = (-11)_{10}$

1.4 Códigos decimales codificados en binario

→ CÓDIGO BCD natural (pesos 8, 4, 2 y 1) 8421

$(25)_{10}$ → $(2)_{10}$ — 0010 → $(0010\ 0101)_{BCD}$
$(25)_{10}$ → $(5)_{10}$ — 0101 → $(0010\ 0101)_{BCD}$

→ CÓDIGO BCD Aiken (pesos 2, 4, 2 y 1)

$(25)_{10}$ → $(2)_{10}$ — 0010 → $(0010\ 1011)_{BCD\ Aiken}$
$(25)_{10}$ → $(5)_{10}$ — 1011 → $(0010\ 1011)_{BCD\ Aiken}$

→ CÓDIGO BCD exceso a 3:
A cada dígito decimal se le asigna el binario y se suma 3

$(25)_{10}$ → $(2)_{10}$ — 0010 → 0010 0101 + 11 = $(0010\ 1000)_{BCD+3}$
$(25)_{10}$ → $(5)_{10}$ — 0101 → 0010 0101 + 11 = $(0010\ 1000)_{BCD+3}$

2. Sistema hexadecimal

Se utiliza para representar en binario números de forma simplificada (se usa mucho en µ)
Utiliza 10 dígitos decimales + letras de la A a la F

$(2EF)_{16} = 2.16^2 + 14.16^1 + 15$
$= 256 + 14.16 + 15 = 512 + 224 + 15 = (751)_{10}$

$(300)_{10}$

300 | 16
140 | 18 | 16
12 | 2 | 1

$(300)_{10} = (12C)_{16}$

→ DE BINARIO A HEXADECIMAL

0111	1010	1101	(binario)
7	10	13	(decimal)
7	A	D	(hexadecimal)

→ De hexadecimal a binario

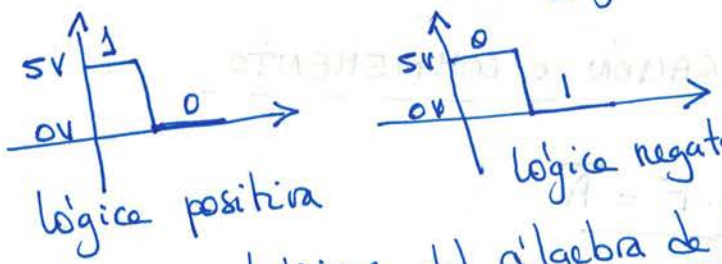
4 D F hexadecimal
4 13 15 decimal
0100 1101 1111 binario

$(4DF)_{16} = (0100\ 1101\ 1111)_2$

3. Algebra de Boole

Este álgebra tiene por objeto representar las formas de razonamiento lógico. Maneja variables que pueden ser únicamente verdadero y falso.

Existen 2 tipos de lógica → Lógica positiva: nivel alto de tensión es 1, nivel bajo de tensión es 0
→ Lógica negativa: nivel alto de tensión es 0, nivel bajo de tensión es 1



3.1 Operaciones básicas del álgebra de Boole

Una función lógica es aquella cuyos valores son binarios y dependen de una expresión algebraica formada por una serie de variables binarias relacionadas entre sí por determinadas operaciones.

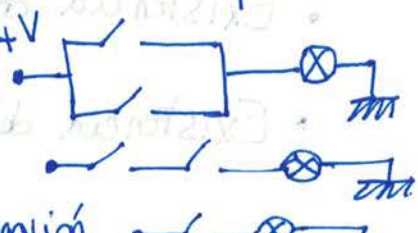
Ej: $f(A,B,C) = A \cdot B + C$

$f(A,B,C) = 1$ ssi A (y) B valen 1 (o) C vale 1
$f(A,B,C) = 0$ ssi A o B valen 0 y C vale 0
o si los 3 valen 0

Al producto lógico lo llamaremos AND
A la suma lógica la llamaremos OR
El número de combinaciones posibles en una tabla de verdad de n entradas es de 2^n combinaciones.

Con las funciones lógicas se pueden realizar numerosas operaciones

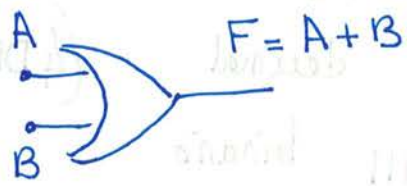
Las + importantes son:
- Función suma: OR
- Función producto: AND
- Función complemento o negación



3.2 PUERTA OR o SUMA LÓGICA

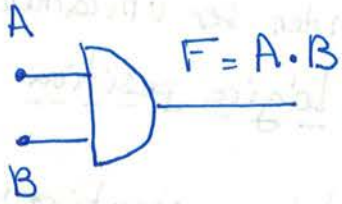
Tabla de Verdad

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1



3.3 PUERTA AND o PRODUCTO LÓGICO

	A	B	F
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1



3.4 PUERTA NOT o NEGACIÓN o COMPLEMENTO

A	F
0	1
1	0

3.5 Propiedades del álgebra de Boole

- Idempotencia: $A \cdot A = A$
  $A + A = A$
- Ley de involución $\overline{\overline{A}} = A$
- Propiedad conmutativa $A + B = B + A$
  $A \cdot B = B \cdot A$
- Propiedad asociativa $A + B + C = (A+B) + C = A + (B+C)$
  $A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$
- Propiedad distributiva $A \cdot (B+C) = A \cdot B + A \cdot C$
  $A + (B \cdot C) = (A+B) \cdot (A+C)$
- Existencia del elemento opuesto $A + \overline{A} = 1$
  $A \cdot \overline{A} = 0$
- Existencia del elemento neutro $A + 0 = A$
  $A \cdot 1 = A$

- Ley de absorción:

$$A + A \cdot B = A$$

$$A \cdot (A+B) = A$$

- Leyes de Morgan:

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

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

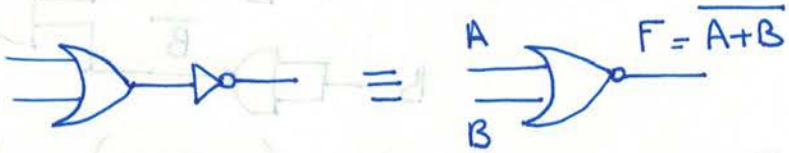
3.6 Puertas lógicas universales

Las funciones OR, AND y NOT son funciones elementales. Existen otras puertas llamadas universales con las que se pueden reproducir todas las operaciones del álgebra de Boole.

Estas puertas son:
- Puertas NOR
- Puertas NAND

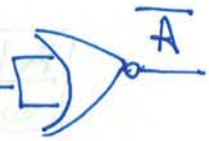
PUERTA NOR: puerta OR seguida de una negación

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

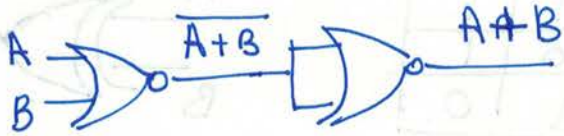


Aplicando la ley de Morgan $F = \overline{A+B} = \overline{A} \cdot \overline{B}$

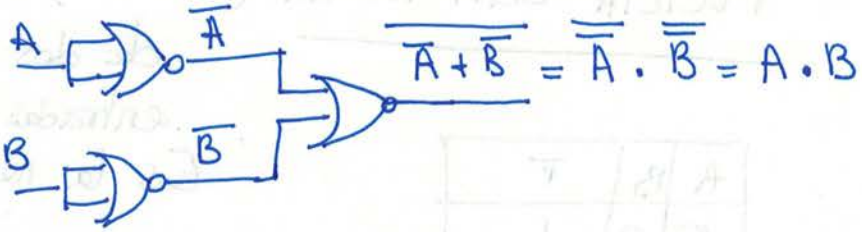
Uniendo las dos entradas obtenemos la puerta NOT



Obtenemos la función OR:



Obtenemos la función AND:



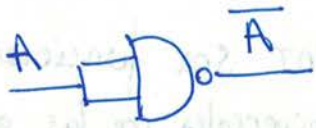
PUERTA NAND

Es una puerta AND seguida de una negación

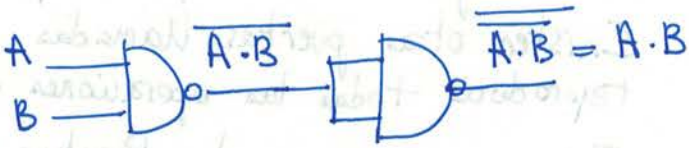
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Aplicando la ley de Morgan $F = \overline{A \cdot B} = \overline{A} + \overline{B}$

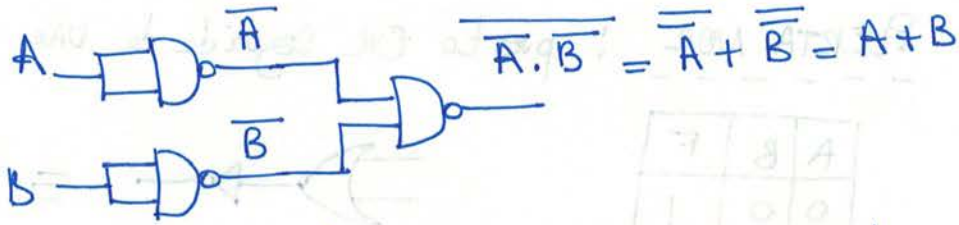
Si unimos las dos entradas en una sola se obtiene una NOT



La función AND se obtiene:

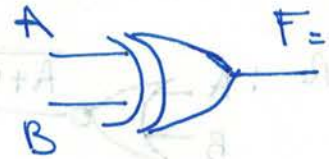


La función OR se obtiene:



PUERTA OR-EXCLUSIVA (EXOR): La salida de una puerta EXOR de 2 entradas es 1 si una y solo una de las entradas está en 1

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



PUERTA EQUIVALENCIA (EXNOR): La salida de una puerta de dos entradas es uno si las entradas son iguales
Es la negada de la EXOR

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1