

# Clase 02 - Lógica Digital

---

Profesor: **IIC2343 - Arquitectura de Computadores**

- Felipe Valenzuela González

Correo:

frvalenzuela@alumni.uc.cl

# **Resumen de la clase pasada**

# Sistema posicional de base genérica

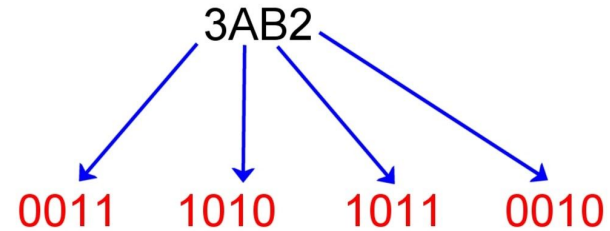
- $s$ : símbolo
- $k$ : = Posición del símbolo en la secuencia, siendo 0 la posición del extremo derecho.
- $b$ : **base**
- $n$ : cantidad de símbolos en la secuencia
- Notación típica:  $( \text{---} )_b$

$$\sum_{k=0}^{n-1} s_k \times b^k$$

# Método de conversión: binaria hacia hexa

- Ocuparemos un método aprovechando concatenación
- Agrupamos los términos numéricos para obtener el resultado

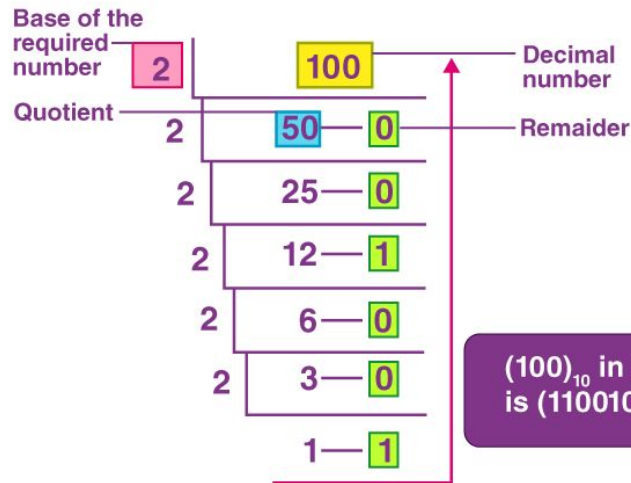
Converting Hex to Binary



$$3AB2_{16} = 11101010110010_2$$

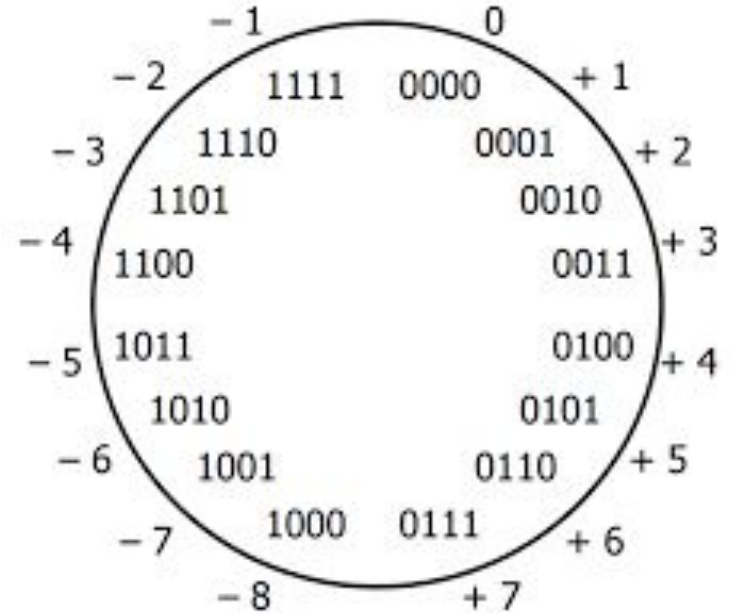
# Método de conversión: decimal hacia binario

- Se obtiene el resto entre el número en base decimal y el divisor 2.
- Se obtiene el resto entre el número en base decimal y el divisor 2.
- Para obtener el siguiente símbolo de la secuencia, realizar la misma operación con el resultado de la división entera del número



## Complemento 2 (C2)

- Sumar una unidad al complemento al C1
- Ahora el cero es intuitivo
- **Contra:** Tenemos una representación desbalanceada
- **Overflow:** Si una operación aritmética resulta en un valor no representable, nos dará un valor erróneo




<b>string</b>	<b>unsigned</b>	<b>sign &amp; magnitude</b>	<b>1's complement</b>	<b>2's complement</b>
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	8	-0	-7	-8
1001	9	-1	-6	-7
1010	10	-2	-5	-6
1011	11	-3	-4	-5
1100	12	-4	-3	-4
1101	13	-5	-2	-3
1110	14	-6	-1	-2
1111	15	-7	-0	-1

**¿Dudas?**



# Introducción:

- Un computador lo definimos como una **máquina programable que ejecuta programas.**
- Para programar necesitamos:
  - Datos: números (enteros, reales) , texto, imágenes, etc 
  - Operaciones: suma, resta, multiplicación, división, etc
  - Variables: simples, arreglos
  - Control de flujo: comparaciones, manejo de ciclos
- La clase de hoy veremos con lo básico que sería los datos, específicamente cómo **representar datos en un computador!**

# Introducción:

- Un computador lo definimos como una **máquina programable que ejecuta programas.**
- Para programar necesitamos:
  - Datos: números (enteros, reales) , texto, imágenes, etc
  - Operaciones: suma, resta, multiplicación, división, etc
  - Variables: simples, arreglos
  - Control de flujo: comparaciones, manejo de ciclos
- La clase de hoy veremos con lo básico que sería los datos, específicamente cómo **representar datos en un computador!**



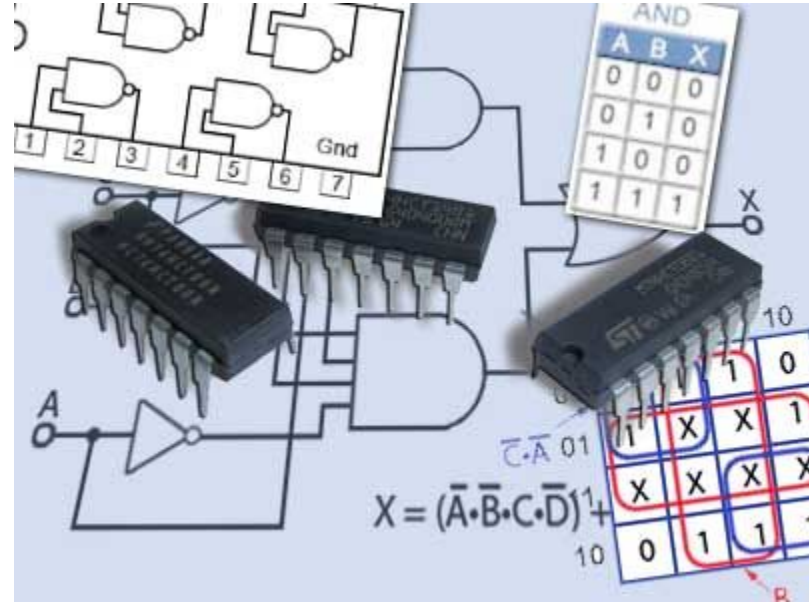
# Objetivos de la clase

- Conocer lógica booleana y circuitos digitales
- Conocer operaciones aritméticas y lógicas
- Entender el manejo de un sumador

# Lógica Digital

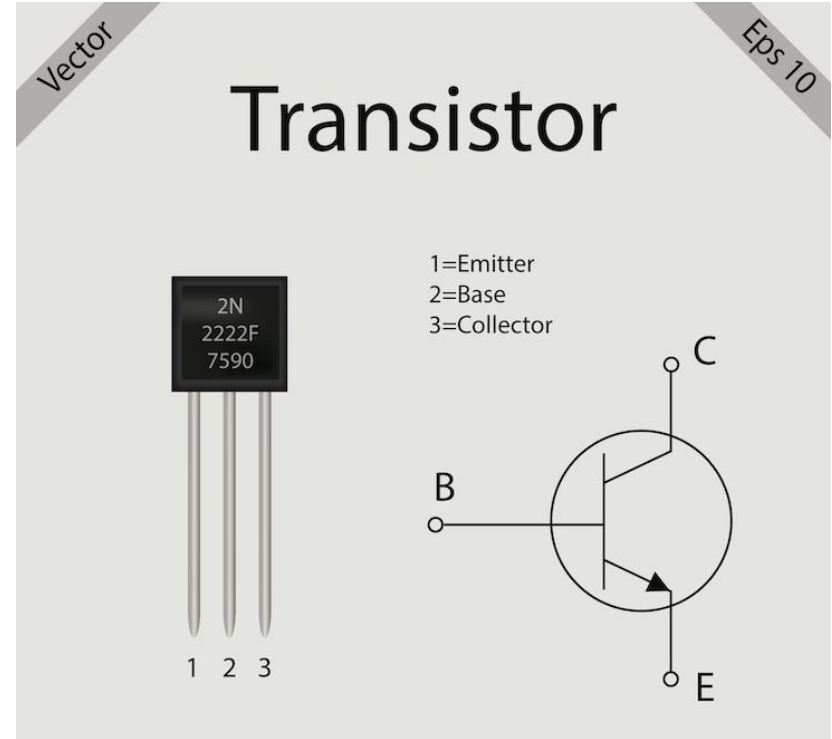
# Lógica Digital

- Un computador por dentro son múltiples **circuitos digitales**
- Cada circuito puede ser descompuesto en una unidad básica llamada **compuerta lógica**



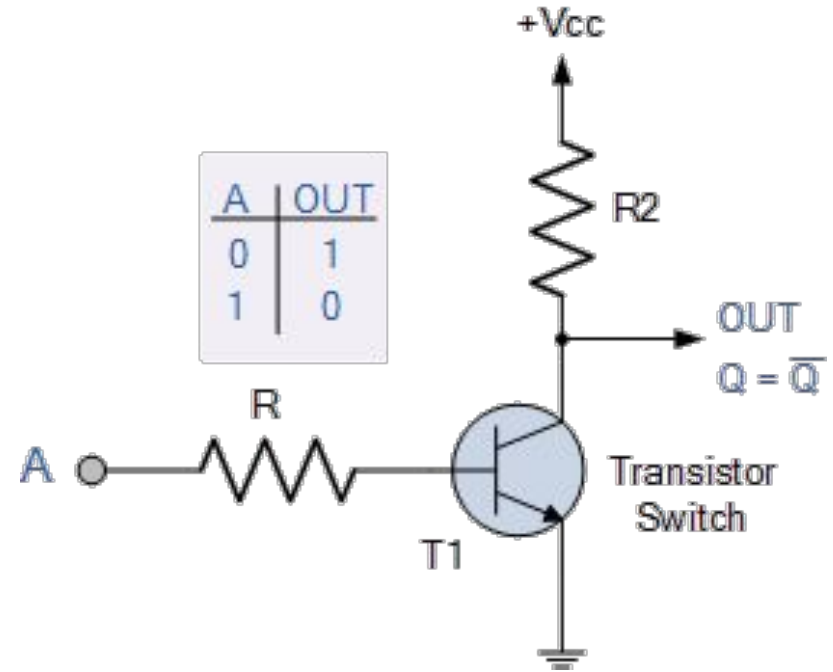
# Compuerta lógica

- Los elementos físicos que utilizamos en la modernidad son los **transistores**
- Es un semiconductor que permite amplificar o bloquear señales eléctricas



# Compuerta lógica: NOT

- Tiene una **entrada** que llamaremos **A**
- Su salida será **OUT** que se comportará como su inverso



**¿Dudas?**

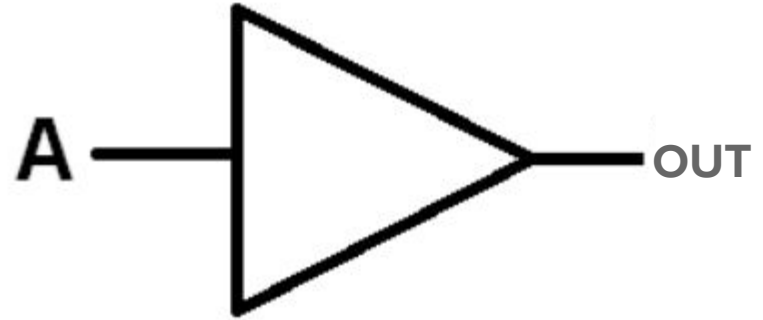


# Lógica Boleana

# Compuerta Booleana: NOT

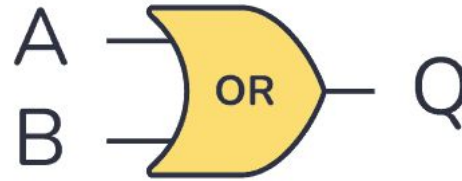
- Tiene una **entrada** que llamaremos **A**
- Su salida será **OUT** que se comportará como su inverso

Input	Output
A	Y
0	1
1	0



# Compuerta Booleana: OR

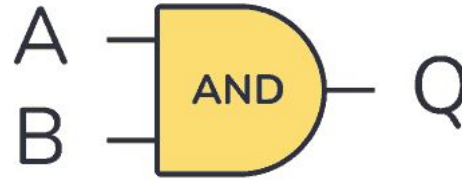
- Tiene dos **entradas** que llamaremos **A y B**
- Su salida será **Q** que será 0 si ambas entradas son cero



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

# Compuerta Booleana: AND

- Tiene dos **entradas** que llamaremos **A** y **B**
- Su salida será **Q** que será 1 si ambas entradas son uno



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

# Compuerta Booleana: XOR

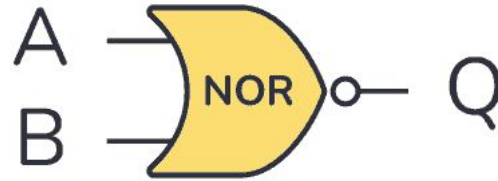
- Tiene dos **entradas** que llamaremos **A y B**
- Su salida será **Q** que será 1 si ambas entradas son distintas



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

# Compuerta Booleana: NOR

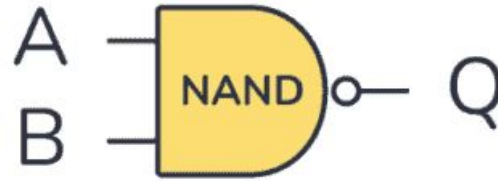
- Tiene dos **entradas** que llamaremos **A y B**
- Su salida será **Q** que será 1 solo si ambas entradas son cero



A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

# Compuerta Booleana: NAND

- Tiene dos **entradas** que llamaremos **A y B**
- Su salida será **Q** que será 0 solo si ambas entradas son uno



A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

# Algebra booleana

Equivalence	Name of Identity
$p \wedge T \equiv p$ $p \vee F \equiv p$	Identity Laws <input type="checkbox"/>
$p \wedge F \equiv F$ $p \vee T \equiv T$	Domination Laws <input type="checkbox"/>
$p \wedge p \equiv p$ $p \vee p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \wedge q \equiv q \wedge p$ $p \vee q \equiv q \vee p$	Commutative Laws
$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ $(p \vee q) \vee r \equiv p \vee (q \vee r)$	Associative Laws
$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	Distributive Laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's Laws <input type="checkbox"/>
$p \wedge (p \vee q) \equiv p$ $p \vee (p \wedge q) \equiv p$	Absorption Laws
$p \wedge \neg p \equiv F$ $p \vee \neg p \equiv T$	Negation Laws

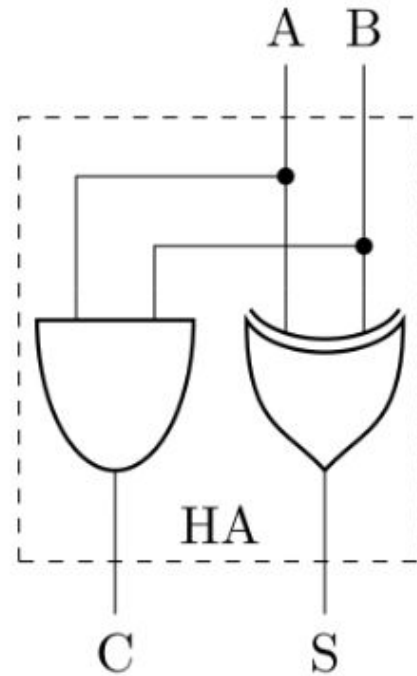


**¿Dudas?**

# Sumador

# Half Adder

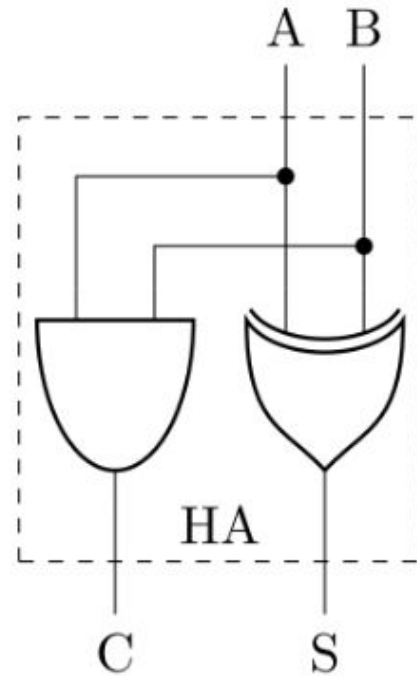
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



(a) Half Adder

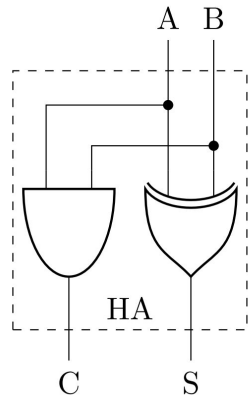
# Half Adder

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

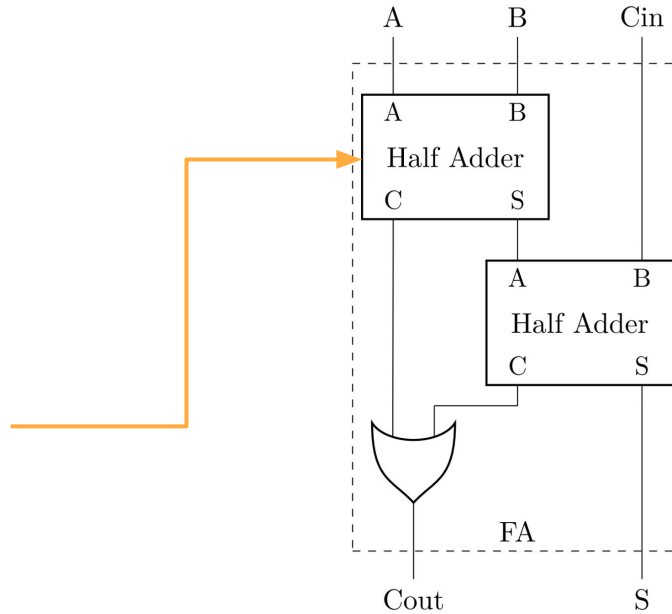


(a) Half Adder

# Full Adder

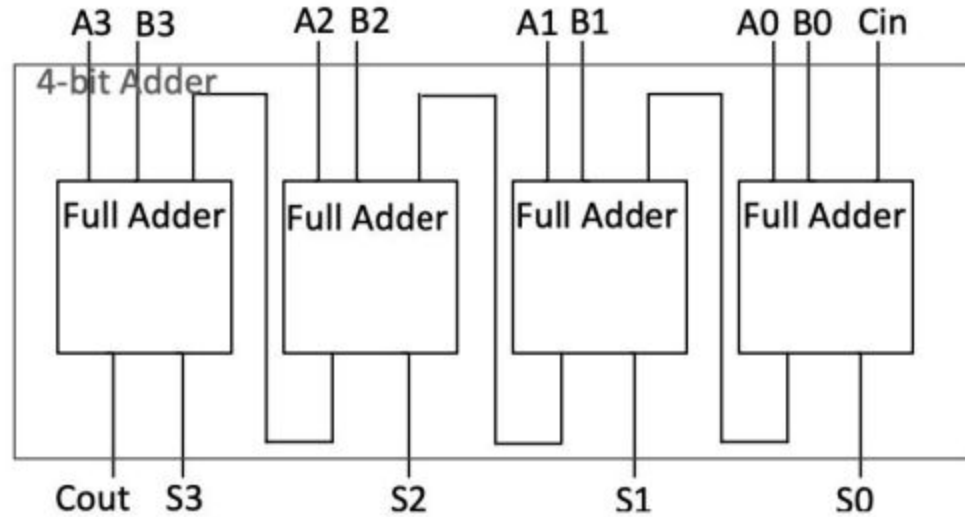


(a) Half Adder

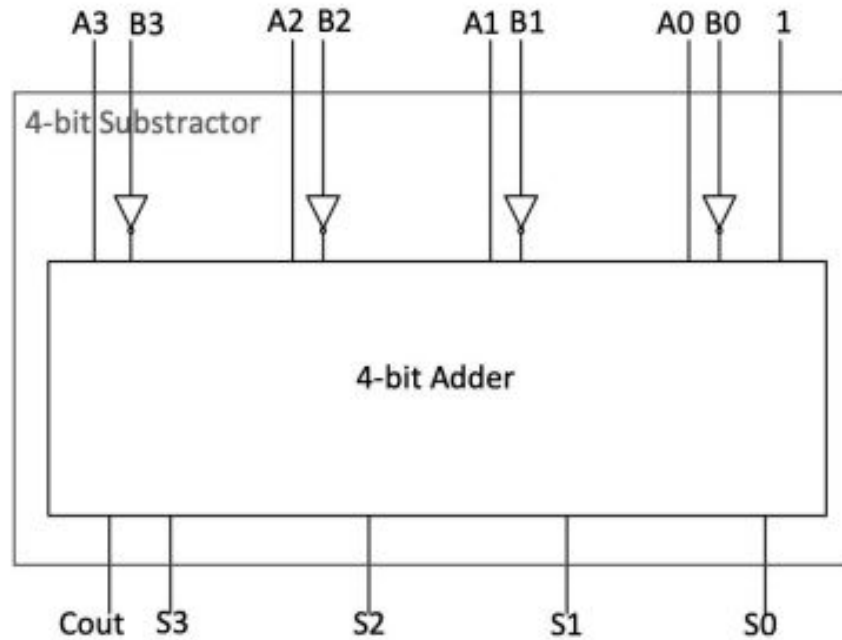


(b) Full Adder

# Sumador



# Restador



**¿Dudas?**



# Clase 02 - Lógica Digital

---

Profesor: **IIC2343 - Arquitectura de Computadores**

- Felipe Valenzuela González

Correo:

frvalenzuela@alumni.uc.cl