

# S3 Operaciones Lógicas

Lenin G. Falconí

2024-10-26

# Outline

- 1 Operaciones Lógicas e Instrucciones del Computador
- 2 Lenguaje de Transferencia de Registros
- 3 Electricidad
- 4 Electrónica y Lógica Digital
- 5 Algebra de Boole
- 6 Circuitos Combinacionales
- 7 Expresión de Funciones Booleanas en Min-Terms(SOP) y Max-Terms(POS)
- 8 Circuitos Secuenciales
- 9 Ejercicios



# Programa:

- Secuencia de pasos para resolver un problema
- Cada paso ejecuta una operación aritmética o lógica
- Cada operación requiere de un diferente conjunto de señales de control
- Cada operación tiene un código único e.g. ADD, MOVE

- sistema integrado de dispositivos electrónicos que interactúa con el entorno a través de dispositivos periféricos o líneas de comunicación y que procesa información.
- La integración de distintos componentes o subsistemas de manera jerárquica. Este conjunto de sistemas busca realizar las siguientes funciones básicas:
  - Procesamiento de Datos
  - Almacenamiento de Datos
  - Transferencia de Datos
  - Control

La *estructura* estudia cómo están interrelacionados los diferentes componentes del computador. Mientras que el *funcionamiento* estudia la operación de cada componente individual como parte de la estructura.



# CPU

Se encarga del control del funcionamiento del computador y del procesamiento.

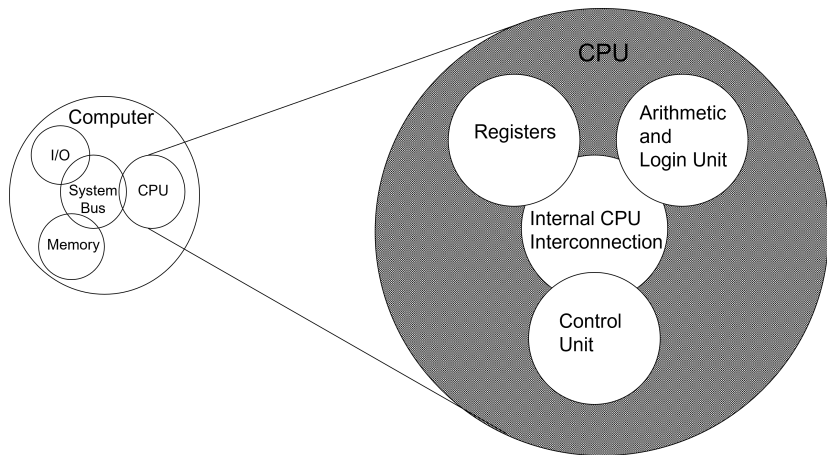


Figure: CPU

# Unidad de Control

Conformada por los distintos circuitos digitales, registros, decodificadores y memorias necesarios para el funcionamiento del computador.

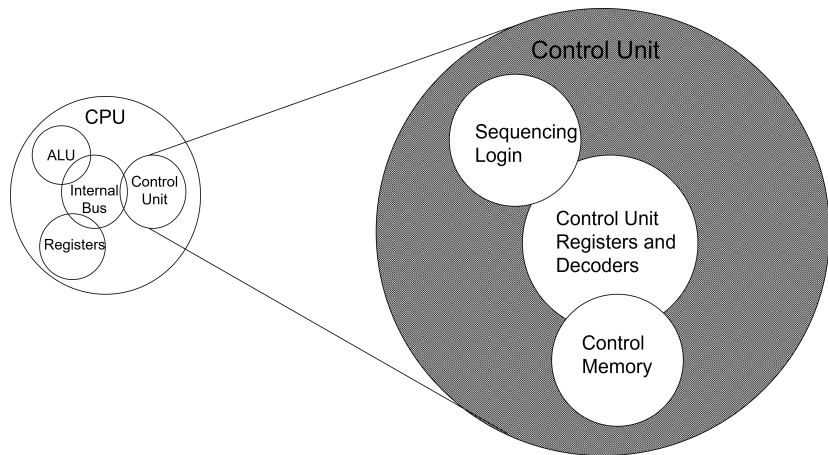
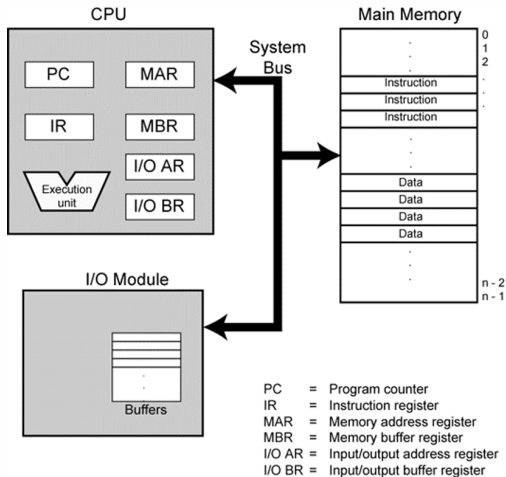


Figure: Unidad de Control

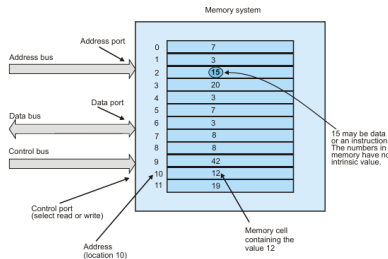


# Componentes del Computador



# Concepto de Memoria

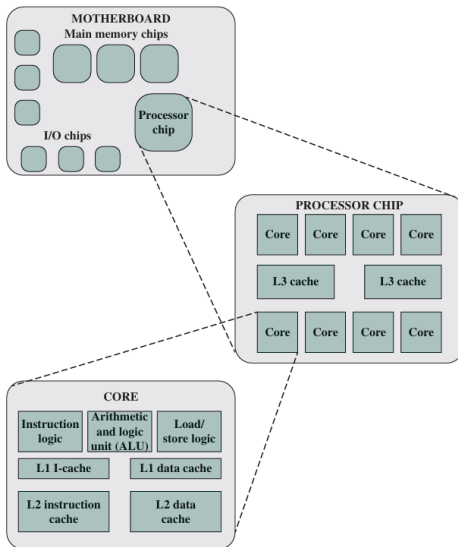
- Puede ser de escritura o lectura dependiendo de una señal de control
- Las distintas operaciones y datos con los que trabaja el computador son mapeados con direcciones de memoria en donde sus valores se encuentran almacenados.
- Se puede pensar como una lista o tabla de elementos almacenados.



# Computadores Multi Núcleo I

- **Computador Multi núcleo:** aquel que tiene múltiples procesadores en un sólo chip
- **Núcleo:** cada unidad de procesamiento que incluye su unidad de control, ALU, registros, y cache.
- **Procesador:** dispositivo de silicón que contiene uno o más núcleos.
- **Memoria Cache:** Memoria más rápida y pequeña que la memoria principal cuyo objetivo principal es **acelerar el acceso a la memoria**. Se conforma en varias capas:  $L_1$ ,  $L_2$ , etc según su cercanía al núcleo.
- **Motherboard:** placa principal de circuito impreso en una computadora. Las placas más pequeñas que se conectan a las ranuras de la placa principal se llaman **tarjetas de expansión**.
- **Printed Circuit (PCB):** placa rígida y plana que sostiene e interconecta chips y otros componentes electrónicos. Típicamente de 2 a 10 capas

# Computadores Mult Núcleo



# Componentes del Núcleo

- **Instruction Logic (IL)**: ejecuta tareas relacionadas con la captación y decodificación de instrucciones.
- **ALU**: realiza las operaciones indicadas por la instrucción.
- **Load/store logic**: administra la transferencia de datos hacia y desde la memoria principal a través de la **cache**.

# Lenguaje de Transferencia de Registros (RTL) I

- Permite definir de manera sencilla las operaciones en el computador
  - No es un lenguaje ensamblador
  - No es un lenguaje de Programación
  - Es una notación
  - Distingue entre las *localidades* de memoria y su *contenido*
  - Se usa [ ] para indicar el contenido de una ubicación de memoria
  - El símbolo  $\leftarrow$  se usa para indicar *transferencia de datos*
- 1 Suponga una pequeña memoria que tenga 4 bits para el bus de dirección ¿cuántas localidades puede almacenar?
  - 2 Estructure la tabla de memoria suponiendo que el contenido de la memoria será de máximo 8 bits.

Si las direcciones son de 4 bits, se puede almacenar hasta  $2^{n=4} = 16$  localidades.

direcc				dato							
0	0	0	0								
0	0	0	1								
0	0	1	0								
.	.	.	.								
.	.	.	.								
1	1	1	1								

En Hexadecimal tendríamos localidades desde la 0x0 hasta la 0xF

# Lenguaje de Transferencia de Registros (RTL) I

- $[0x0F] \leftarrow [0x0F] + 1$ : el contenido de la localidad de memoria  $0x0F$  se incrementa en 1 y se almacena en la misma localidad
- El símbolo  $=$  se usa alternativamente para expresar transferencia

Considere las siguientes operaciones:

- 1  $[0x14] = 5$ : el contenido de la dirección de memoria  $0x14$  es 5
- 2  $[0x14] \leftarrow 6$ : el valor o literal 6 se carga en  $0x14$
- 3  $[0x14] \leftarrow [6]$ : el contenido de la dirección  $0x06$  se carga en  $0x14$
- 4  $[0x0C] \leftarrow [0x03] + 3$ : el contenido de la dirección  $0x03$  se suma con el valor 3 y el resultado se carga en  $0x0C$
- 5  $[0x13] \leftarrow [0x07] + [0x08]$ : la suma de los contenidos de las localidades de memoria 7 y 8 se colocan en la dirección 19 ( $19_{10}=13_{16}$ )
- 6  $[0x04] \leftarrow [[0x02]]$ : **puntero** o **direccionamiento indirecto**. El valor a copiar en la localidad 4 es el contenido en la dirección definida por el contenido de la localidad 2.



# Ejercicio I

Considere la siguiente memoria abstracta.

Obtenga:  $X =$

$$3 + [0x04] + [1 + [0x03]] + [[0x0A]] + [[0x09] * 3]$$

Dirección	Dato
0x00	6
0x01	2
0x02	3
0x03	4
0x04	5
0x05	2
0x06	8
0x07	1
0x08	5
0x09	2
0x0A	1
0x0B	5

# Ejercicio - Solución I

Considere la siguiente memoria abstracta.

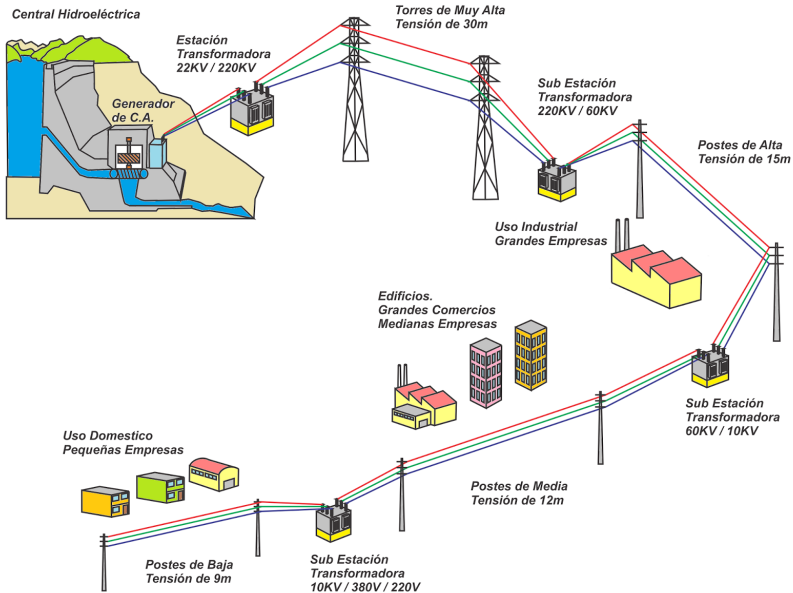
Obtenga:  $X =$

$$3 + [0x04] + [1 + [0x03]] + [[0x0A]] + [[0x09] * 3]$$

$$X = 3 + 5 + 2 + 2 + 8$$

Dirección	Dato
0x00	6
0x01	2
0x02	3
0x03	4
0x04	5
0x05	2
0x06	8
0x07	1
0x08	5
0x09	2
0x0A	1
0x0B	5

# Electricidad - Generación



# Electricidad - Tensión Trifásica I

```
import numpy as np
import matplotlib.pyplot as plt

frecuencia = 60
t = np.linspace(0,2/frecuencia, 1000)
omega = 2*np.pi*frecuencia
Vmax = 230*np.sqrt(2) # Voltaje Pico
R = Vmax*np.sin(omega*t)
S = Vmax*np.sin(omega*t-2*np.pi/3)
T = Vmax*np.sin(omega*t+2*np.pi/3)
# Plot
plt.figure(figsize=(10,6))
plt.plot(t,R,label="Fase R", color="red")
plt.plot(t,S,label="Fase S", color="green")
plt.plot(t,T,label="Fase T", color="blue")
```

```
plt.title("Tensión Trifásica")
plt.ylabel("Voltaje (V)")
plt.xlabel("Tiempo (s)")
plt.grid(True, linestyle="--", alpha=0.7)
plt.legend()
plt.tight_layout()
plt.savefig("./images/tensionTrifasica.png")
# plt.show()
```

# Electricidad - Tensión Trifásica III

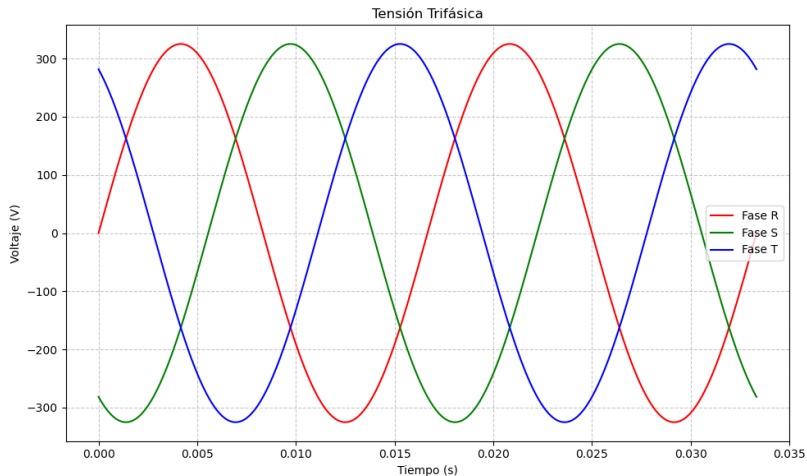


Figure: Tensión Trifásica

# Electricidad - Conceptos Básicos I




Magnitud	Descripción	Unidad
Voltaje $V$	Diferencia de potencial eléctrico entre dos puntos	Voltios (V)
Corriente $I$	Flujo de carga eléctrica por unidad de tiempo	Amperios (A)
Resistencia $R$	Oposición al flujo de corriente	Ohmios ( $\Omega$ )
Capacitancia $C$	Capacidad para almacenar carga eléctrica	Faradios (F)
Inductancia $L$	Oposición a cambios en la corriente	Henrios (H)

## Valor Eficaz

El Valor Eficaz o Root Mean Square (RMS) de una señal sinusoidal de

tensión se define como  $V_{\text{RMS}} = \sqrt{\frac{1}{T} \int_0^T V_{\text{max}} \sin^2(\omega t) dt}$

# Electricidad - Relaciones Básicas I

Elemento	Voltaje	Corriente	Potencia
	$v(t) = R \cdot i(t)$	$i(t) = \frac{v(t)}{R}$	$p = vi = i^2 R$
	$v(t) = L \frac{di(t)}{dt}$	$i(t) = \frac{1}{L} \int v(t) dt$	$p = vi = Li \frac{di}{dt}$
	$v(t) = \frac{1}{C} \int i(t) dt$	$i(t) = C \frac{dv(t)}{dt}$	$p = vi = Cv \frac{dv}{dt}$

## Ley de Ohm

El voltaje  $V$ , la corriente  $I$  y la resistencia  $R$  se relacionan con la Ley de Ohm  
 $V = IR$



# Leyes de Kirchhoff y Tipos de Conexiones

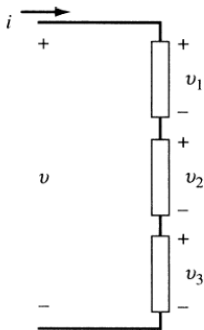


Figure: Circuito Serie

LVK:  $v = v_1 + v_2 + v_3$

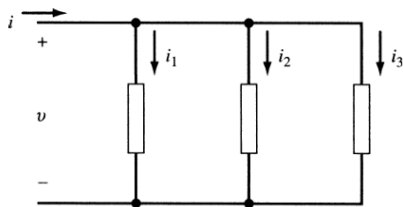
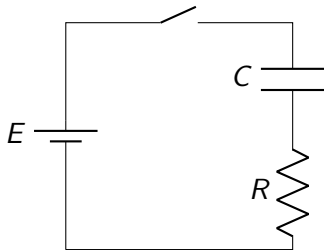


Figure: Circuito Paralelo

LCK:  $i = i_1 + i_2 + i_3$

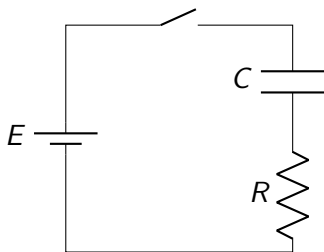
# Aplicación Leyes de Kirchhoff



## Problema:

Encuentre expresiones para el voltaje del capacitor y corriente del circuito, si el switch se cierra en  $t = 0$

# Aplicación Leyes de Kirchhoff I



**Solución:**

$$\text{LVK: } E = v_C + v_R$$

$$\text{Pero } i = C \frac{dv_C}{dt}$$

$$\text{Ley de Ohm: } v_R = iR$$

Dado que el circuito es serie  $i$  es la misma, entonces

$$E = v_C + RC \frac{dv_C}{dt}$$

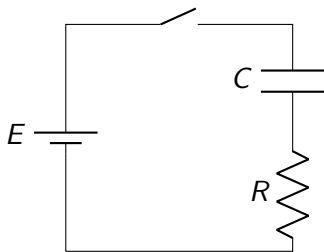
Para  $t = 0$ ,  $v_C = 0$ , entonces

$$v_C = E(1 - e^{-\frac{t}{RC}})$$

Aplicando las propiedades de la capacitancia, tenemos

$$i = \frac{E}{R} e^{-\frac{t}{RC}}$$

# Aplicación Leyes de Kirchhoff



## Preguntas:

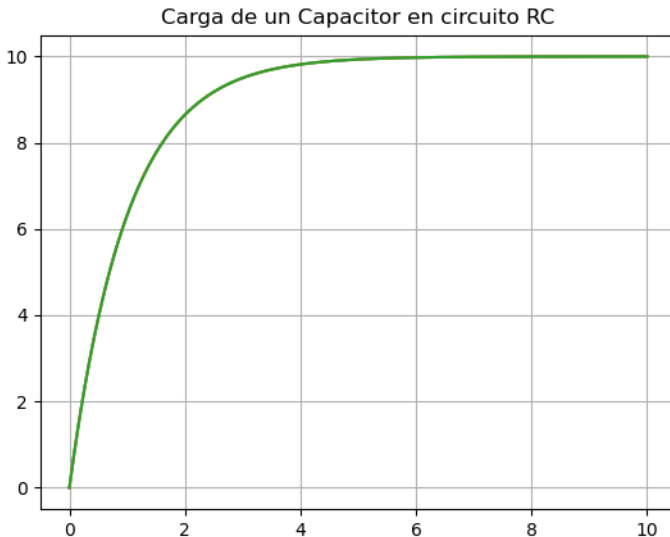
- 1 ¿Qué sucede con el voltaje en el capacitor y la corriente del circuito cuando  $t \rightarrow +\infty$ ?
- 2 ¿Qué sucede con las expresiones obtenidas para el voltaje del capacitor y la corriente si la fuente de voltaje no es constante y es de la forma  $A\sin(\omega t)$ ?
- 3 Para  $t = RC$ , en qué porcentaje se ha cargado el capacitor?
- 4 Considere un capacitor de  $1\mu\text{F}$ , una resistencia de  $1\text{M}\Omega$ , obtenga la curva de la corriente en el tiempo usando python.

# Aplicaciones Leyes de Kirchhoff I

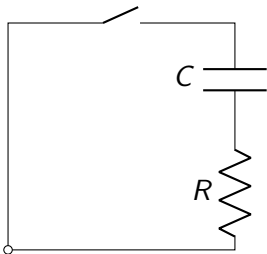
Para el circuito de carga del capacitor se obtuvo que  $v_c = E(1 - e^{-\frac{t}{RC}})$ , considerando  $C = 1\mu\text{F}$ ,  $R = 1\text{M}\Omega$ , y  $E = 10\text{V}$  el siguiente código permite obtener  $v_c$  en función del tiempo hasta  $t = 10\text{s}$

```
import numpy as np
import matplotlib.pyplot as plt
R = 1e6 # 1 MegaOhm
C = 1e-6 # 1 microFarad
V = 10 # 10 Vdc
t = np.linspace(0,10,1000)
vc = V*(1-np.exp(-t/(R*C)))
plt.plot(t,vc)
plt.title("Carga de un Capacitor en circuito RC")
plt.grid()
# plt.show() ## descomentar para ver fig
# plt.savefig('./images/cargaCapacitor.png') # guardar plot
```

# Aplicaciones Leyes de Kirchhoff



# Aplicación Leyes de Kirchhoff



## Problema:

Encuentre expresiones para el voltaje del capacitor y corriente del circuito, si el switch se cierra en  $t = 0$  y el capacitor inicia cargado con un voltaje  $V_0$

# Materiales Semiconductores I

- Los materiales conductores tienen la característica de producir una corriente eléctrica en presencia de un campo eléctrico.
- Un semiconductor es un material que exhibe las características tanto de un buen conductor como de un buen aislante. Esta característica se controla por una entrada de control.
- La conductividad de un semiconductor se controla mediante: temperatura, impurezas (dopaje), y campo eléctrico.
- **Ejemplos comunes:** Silicio (Si), Germanio (Ge), Arseniuro de Galio (GaAs)
- Un transistor es un semiconductor que opera como un switch digital. Cambia de alta a baja resistencia dependiendo del estado de una señal de entrada.
- Introducción a Diodos y Transistores(Vídeo)
- ¿Qué es un semiconductor?(Vídeo)

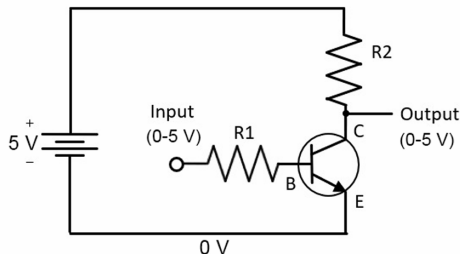


# Materiales tipo N y tipo P

- **Material Tipo N:** Semiconductor dopado con impurezas donadoras
- **Impurezas comunes:** Fósforo (P), Arsénico (As), Antimonio (Sb) en Silicio
- **Características:**
  - Exceso de electrones libres
  - Portadores mayoritarios: electrones (n)
  - Portadores minoritarios: huecos (p)
- **Material Tipo P:** Semiconductor dopado con impurezas aceptoras
- **Impurezas comunes:** Boro (B), Aluminio (Al), Galio (Ga) en Silicio
- **Características:**
  - Exceso de huecos libres
  - Portadores mayoritarios: huecos (p)
  - Portadores minoritarios: electrones (n)

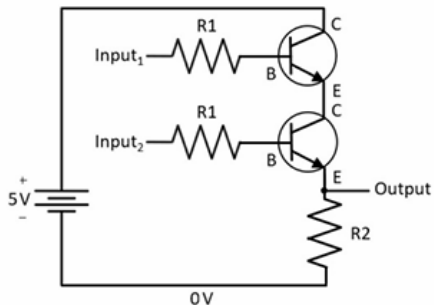
# Compuertas Lógicas

- Son arreglos de circuitos con transistores que permiten realizar operaciones lógicas
- Un transistor tiene un voltaje de switching de 0.7V.
- Con un  $V \geq 0.7$ , el transistor se activa y la resistencia entre colector y emisor se reduce, colocando la salida a un bajo voltaje.
- El comportamiento del circuito se puede expresar en una **tabla de verdad**



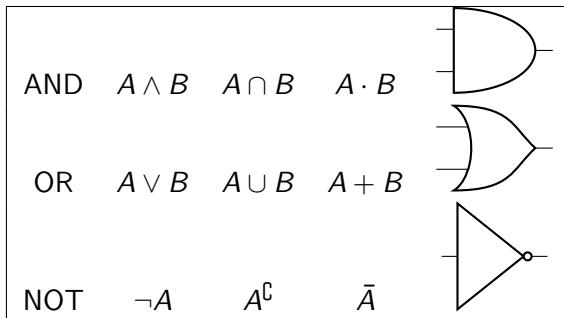
# Compuertas Lógicas

$input_1$	$input_2$	salida
0	0	0
0	1	0
1	0	0
1	1	1



# Álgebra de Boole y Compuertas Lógicas I

- Utilizada para resolver problemas de diseño de circuitos de conmutación
- Las variables y las operaciones son lógicas
- 1 equivale a Verdadero
- 0 equivale a Falso
- Las operaciones lógicas AND, OR y NOT se denotan como:



# Álgebra de Boole y Compuertas Lógicas II

- Es importante notar que las compuertas NAND y NOR son las respectivas negaciones de las compuertas AND y OR i.e.

$$A \text{ NAND } B = \neg(A \wedge B) = \overline{A \wedge B}$$

$$A \text{ NOR } B = \neg(A \vee B) = \overline{A \vee B}$$

- AND, OR y NOT son un conjunto funcionalmente completo.
- NAND y NOR pueden implementar cualquier circuito digital ya que las AND, OR y NOT se pueden implementar directamente sólo con compuertas NAND o NOR. Condición favorable para procesos de fabricación.

# Leyes del Algebra de Boole

+	·	Ley
$A + 0 = A$	$A \cdot 1 = A$	Identidad
$A + 1 = 1$	$A \cdot 0 = 0$	Dominio
$A + A = A$	$A \cdot A = A$	Idempotencia
$A + \bar{A} = 1$	$A \cdot \bar{A} = 0$	Complementariedad
$A + A \cdot B = A$	$A \cdot (A + B) = A$	Absorción
$A + \bar{A} \cdot B = A + B$	$A \cdot (\bar{A} + B) = A \cdot B$	Reducción
$(A + B) + C = A + (B + C)$	$(AB)C = A(BC)$	Asociatividad
$A + B \cdot C = (A + B) \cdot (A + C)$	$A \cdot (B + C) = A \cdot B + A \cdot C$	Distribución
$\overline{(A + B)} = \bar{A} \cdot \bar{B}$	$\overline{A \cdot B} = \bar{A} + \bar{B}$	De Morgan

Negación de la Negación:  $\bar{\bar{A}} = A$

- Conjunto de compuertas lógicas interconectadas cuya salida, en un momento dado, es función únicamente de las entradas en ese instante.
- La relación puede ser expresada por *funciones booleanas* o por *tablas de verdad*.
- La ecuación booleana se puede simplificar con aplicación de las identidades o postulados básicos del álgebra booleana o por Mapas de Karnaugh
- Se pueden expresar como Suma de Productos (SOP) o productos de sumas (POS)
- El Teorema de Morgan permite hacer la conmutación de las dos representaciones.

# Representación de Min-Terms o Sumas de Productos (SOP)

Sea  $F(X_1, X_2, \dots, X_n)$  la salida de un circuito lógico combinacional booleano que recibe como entradas  $X_1, X_2, \dots, X_n$ , entonces:

- 1 Localizar los casos de la *Tabla de Verdad* donde la Función  $F = 1$
- 2 Para cada uno de los casos identificados escribir **el producto** de las entradas considerando que si la entrada en la tabla vale 1, se mantiene el símbolo. Si vale 0, se escribe el complemento.
- 3 Suma los productos obtenidos



# Representación de Max-Terms o Productos de Sumas (POS)

Sea  $F(X_1, X_2, \dots, X_n)$  la salida de un circuito lógico combinacional booleano que recibe como entradas  $X_1, X_2, \dots, X_n$ , entonces:

- 1 Localizar los casos de la *Tabla de Verdad* donde la Función  $F = 0$
- 2 Para cada uno de los casos identificados escribir **la suma** de las entradas considerando que si la entrada en la tabla vale 0, se mantiene el símbolo. Si vale 1, se escribe el complemento.
- 3 Sume los productos obtenidos

# Ejemplo de Representación como SOP I

Considera la Siguiete Tabla de Verdad:

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

- Los casos que interesan son: 000, 010, 011, y 110, porque  $F = 1$ .
- En consecuencia, existen 4 Sumas de Productos. En cada producto, si la variable está con 0 se complementa. Si está con 1 se deja:

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}$$

- Una vez obtenida se debe reducir por medio de Mapa K. o postulados del álgebra booleana.

# Ejemplo de Representación como POS I

Considera la Siguiete Tabla de Verdad:

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

- Los casos que interesan son: 001, 100, 101, y 111, porque  $F = 0$
- En consecuencia, existen 4 Productos de Sumas. En cada producto, si la variable está con 1 se complementa. Si está con 0 se deja:

$$F = (A + B + \bar{C})(\bar{A} + B + C)(\bar{A} + B + \bar{C})(\bar{A} + \bar{B} + \bar{C})$$

- Una vez obtenida se debe reducir por medio de Mapa K. o postulados del álgebra booleana.

La salida actual de estos circuitos depende de la entrada actual y de la historia pasada de las entradas. Estos circuitos usan una señal de reloj, generalmente. Ejemplos son:

- Biestables o latch SR
- Biestable D
- Registros
- Contadores

- ❶ A partir de la tabla de verdad de la compuerta OR exclusiva de dos entradas obtenga la función booleana como SOP (min-términos).
- ❷ Para el ejercicio anterior obtenga la representación en POS (max-términos).
- ❸ ¿Puede representar el circuito sólo con compuertas NAND?
- ❹ Simplificar  $F = ACD + \bar{A}BCD$ . Resp:  $CD(A + B)$
- ❺ Simplificar  $F = ABC + A\bar{B}\bar{A}\bar{C}$  R:  $A(\bar{B} + C)$
- ❻ A partir de la Tabla 1 de verdad obtener la representación en SOP.
- ❼ Usando Mapas de Karnaugh obtenga la simplificación del circuito de la Tabla 1

Table: Ejercicio de tres variables

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