**Lógica digital**

Las computadoras necesitan almacenar datos e instrucciones en memoria.

Sistema binario: sólo dos estados posibles.

¿Por qué?

Es mucho más sencillo identificar entre solo dos estados.

Es menos propenso a errores

**Diseño de circuitos**

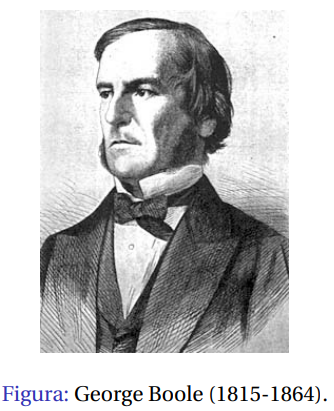
Circuitos que operan con valores lógicos:

Verdadero = 1

Falso = 0

**Idea**: realizar diferentes operaciones lógicas y matemáticas combinando circuitos.

**Álgebra de Boole**



George Boole, desarrolló un sistema algebraico para formular proposiciones con símbolos.

Su álgebra consiste en un método para resolver problemas de lógica que recurre solamente a los valores binarios:

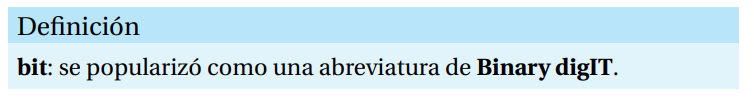
* verdadero / falso.
* on / off.
* 1 / 0.

Tres operadores:

* AND (y).
* OR (o).
* NOT (no).

Las variables Booleanas **sólo** pueden tomar los valores binarios: 1 ó 0.

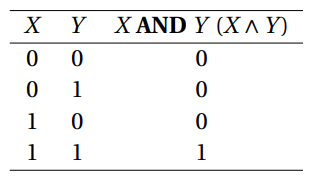
Una variable Booleana **representa un bit**.



**Operadores básicos: AND**

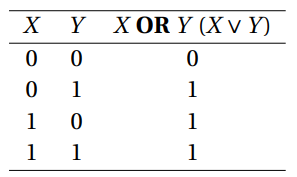
Un operador booleano puede ser completamente descrito usando tablas de verdad.

El operador **AND** es conocido como producto booleano (.). También se lo nota como ∧:



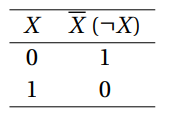
**Operadores básicos: OR**

El operador OR es conocido como suma booleana (+). También se lo nota como ∨:



**Operadores básicos: NOT**

El operador NOT se nota con una barra X. Otra forma de notarlo es ¬X.



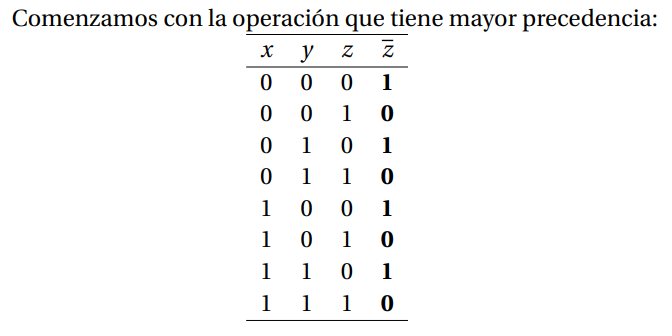
Funciones booleanas

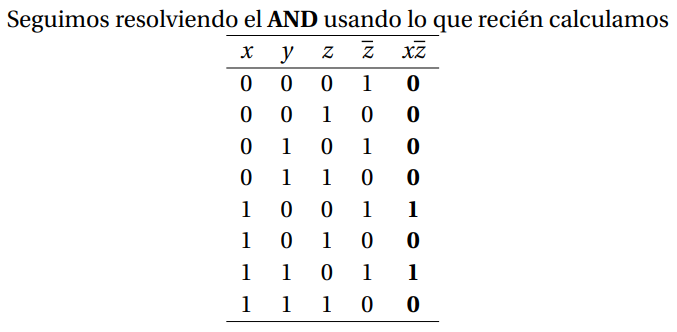
Queremos hacer la tabla de verdad de esta función

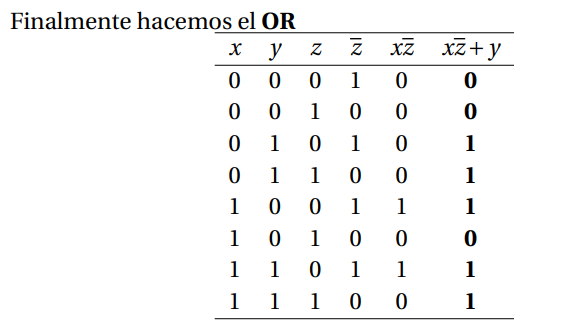


El **NOT** tiene mayor precedencia que el resto de los operadores

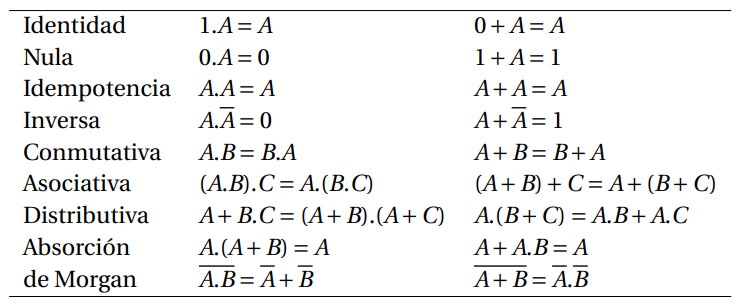
El **AND** mayor precedencia que el **OR**

****

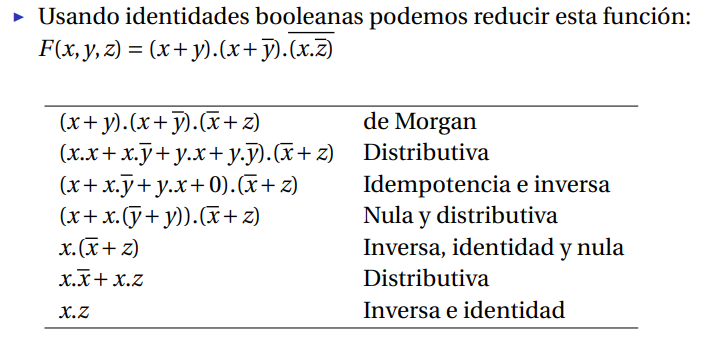
****

****

**Propiedades**

****

**Identidades: ejemplo de aplicación**

****

**Fórmulas equivalentes**

Varias fórmulas pueden tener la misma tabla de verdad:

Son lógicamente equivalentes.

En general se suelen elegir las formas **canónicas**

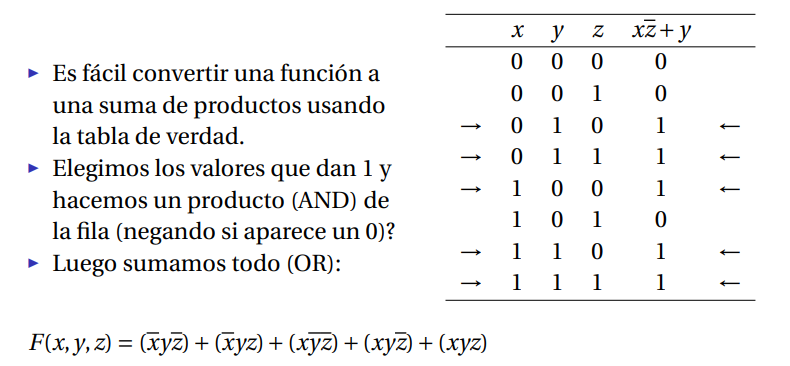
Suma de productos:

F1(x,y,z) = x.y +z.x +y.z

Producto de sumas:

F2(x,y,z) = (x +y).(z +x).(y +z)

**Suma de productos**



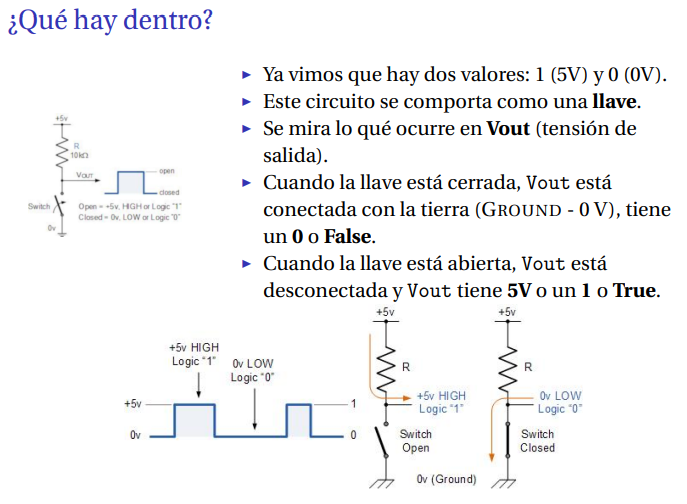
**Circuitos booleanos**

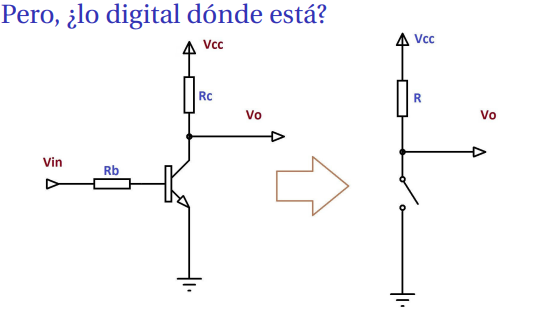
Las computadoras digitales contienen circuitos que implementan funciones booleanas.

Cuando más simple la función más chico el circuito.

Son más baratos, consumen menos, y ¡son más rápidos!

Podemos usar las identidades del álgebra de Boole para reducir estas funciones y hacer circuitos más simples.

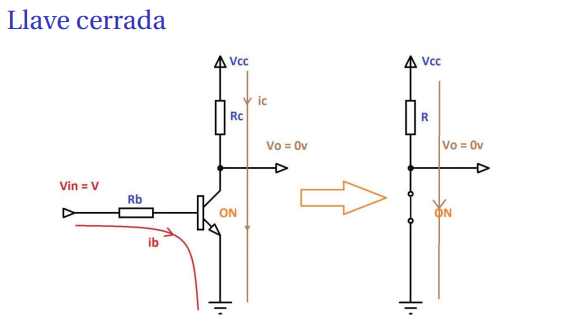




El dispositivo del centro del circuito es un **transistor**, dispositivo que revolucionó nuestra historia en los años 70.

Se lo puede usar como **amplificador** de una señal (lo que entra en Vin).

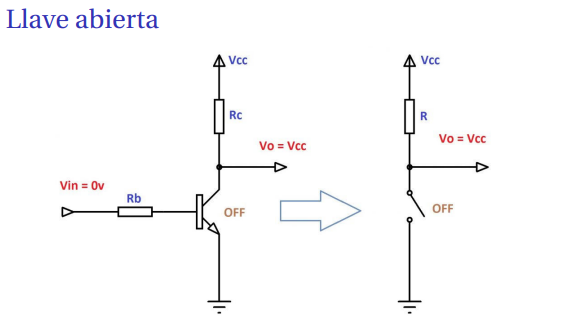
También puede ser usado como una llave controlada por una señal.



El transistor se satura (se lo conoce como saturated state) cuando se aplica un potencial suficientemente alto en Vin.

Durante esta condición el transistor actúa (casi) como si fuera un cortocircuito.

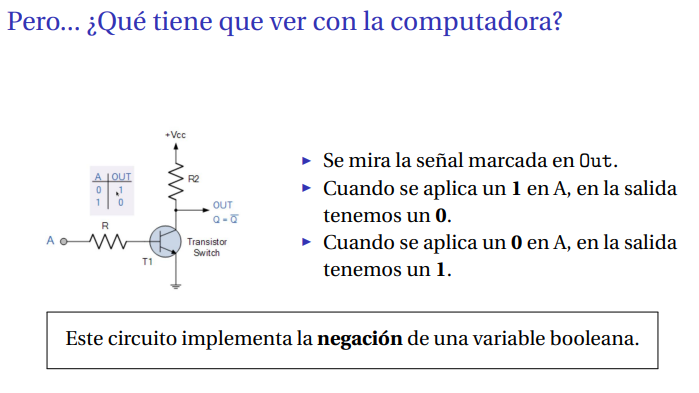
Circula una corriente que depende de la resistencia.



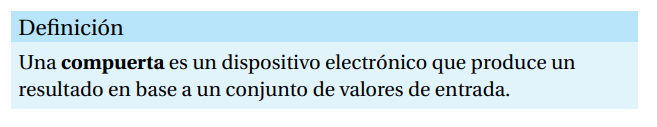
El transistor está apagado (se lo conoce también como cut-off state cuando el potencial aplicado a Vin es 0V.

En este estado, el transistor actúa como si fuera un circuito abierto.

No hay circulación de corriente entre VCC y GROUND.

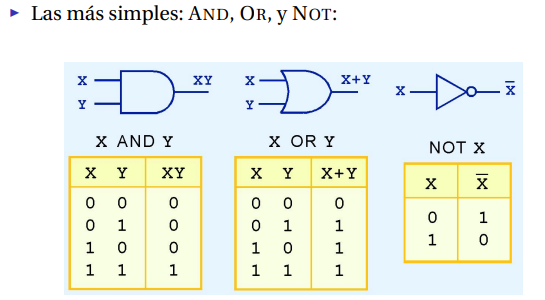


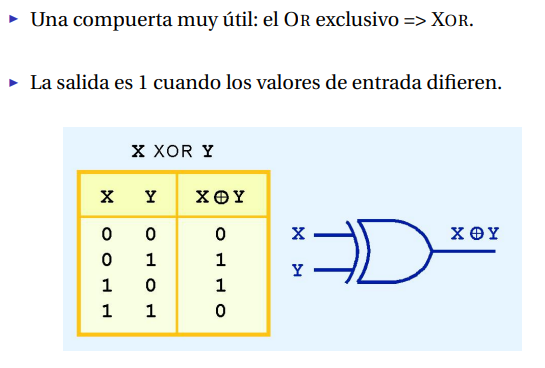
**Compuertas lógicas**



En realidad, están formadas por uno o varios transistores, pero lo podemos ver como una **unidad**.

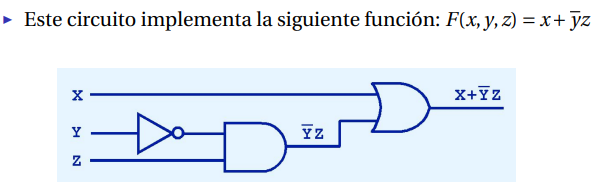
Los circuitos integrados contienen colecciones de compuertas conectadas con algún propósito.



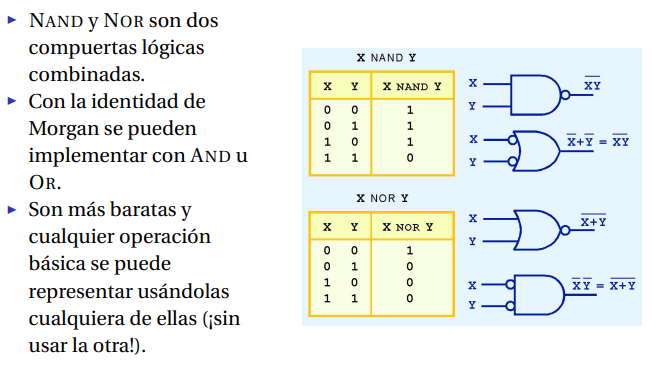


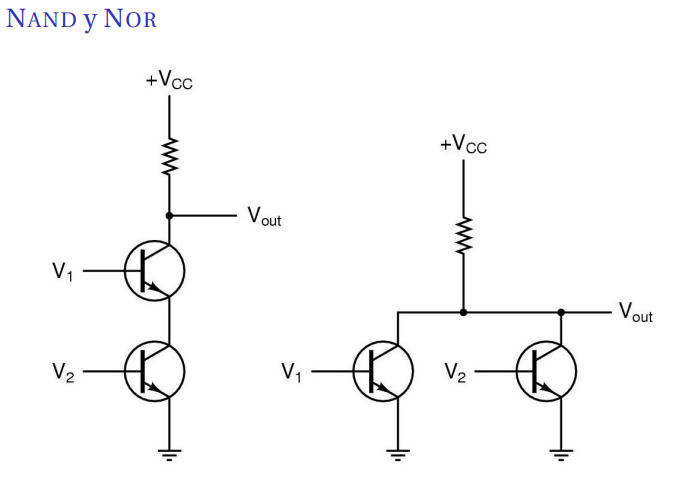
**Implementación de funciones booleanas**

Combinando compuertas se pueden implementar funciones booleanas.



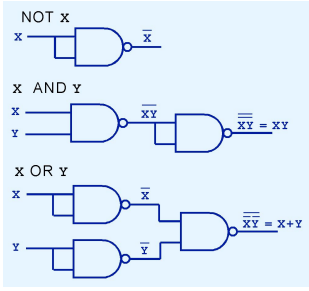
**Compuertas lógicas combinadas**

****

****

**Ejemplos**

* NOT usando NAND: simplemente unir las dos entradas.
* Utilizando solo NAND o NOR se pueden realizar circuitos con la misma funcionalidad que el AND y OR.



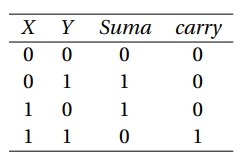
**Circuitos combinatorios**

Producen una salida específica al (casi) instante que se le aplican valores de entrada.

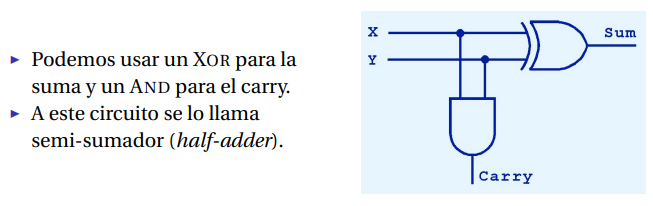
Implementan funciones booleanas.

La aritmética y la lógica de la CPU se implementan con estos circuitos.

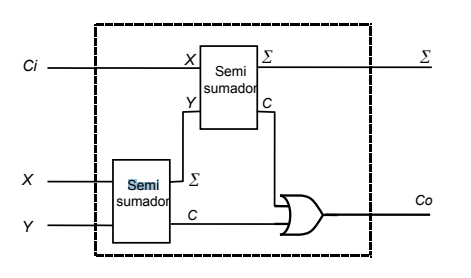
**Sumador**



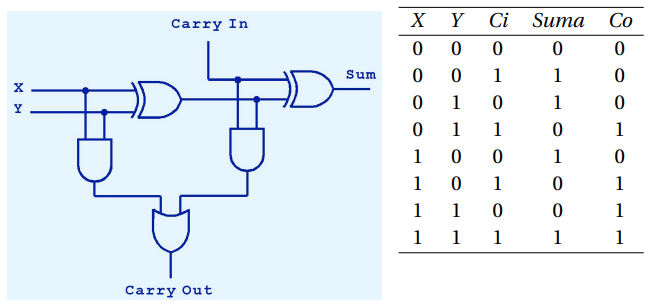
**Semi-Sumador**



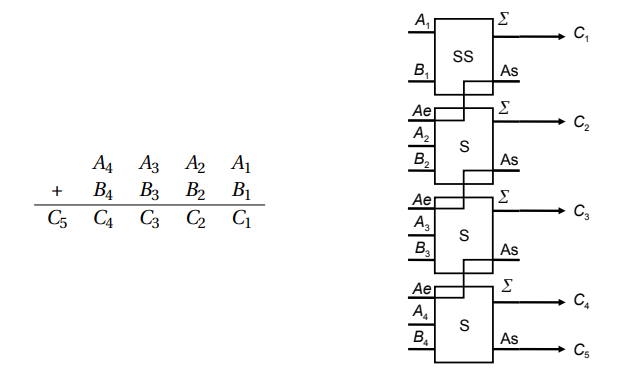
**Sumador: estructura interna**



**Sumador: estructura interna y tabla de verdad**



**Sumador de 4 bits**

****

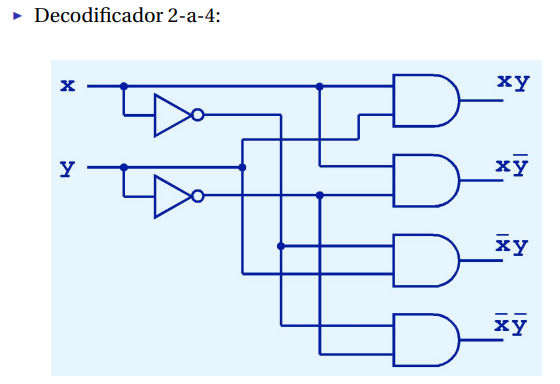
**Decodificadores**

Los decodificadores de n entradas pueden seleccionar una de 2^n salidas.

Son ampliamente utilizados.

Por ejemplo: Seleccionar una locación en una memoria a partir de una dirección colocada en el bus memoria.

Decodificadores: ejemplo



**Multiplexores**

Selecciona una salida a de varias entradas.

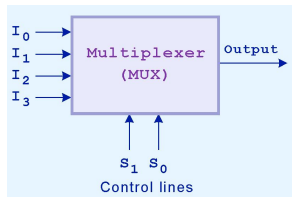
La entrada que es seleccionada como salida es determinada por las líneas de control.

Para seleccionar entre n entradas, se necesitan log2n líneas de control.

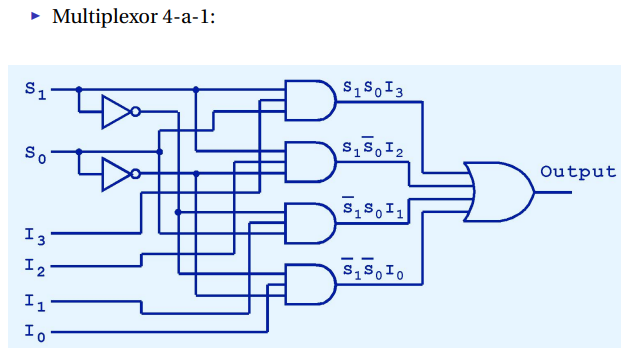
**Demultiplexor**

Exactamente lo contrario al multiplexor.

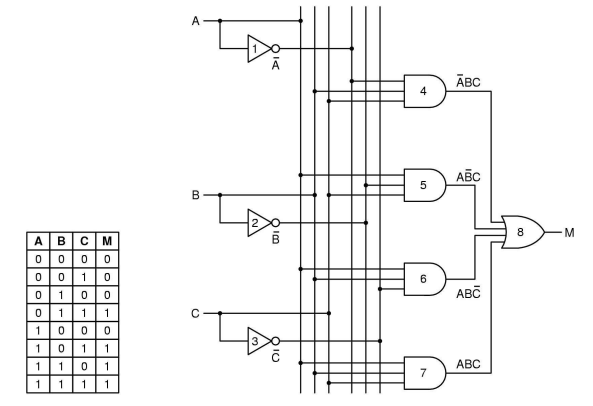
Dada una entrada la direcciona entre n salidas, usando log2n líneas de control.



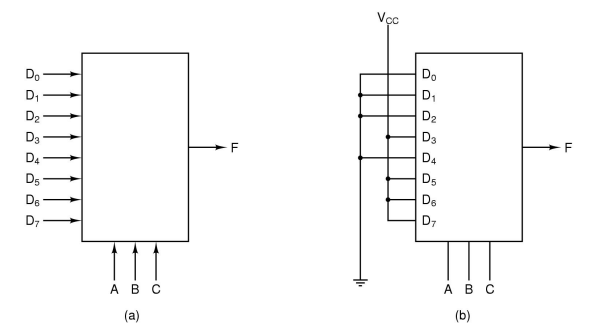
Multiplexor: ejemplo



**Función mayoría**



**Función mayoría con multiplexor**



Ejemplo

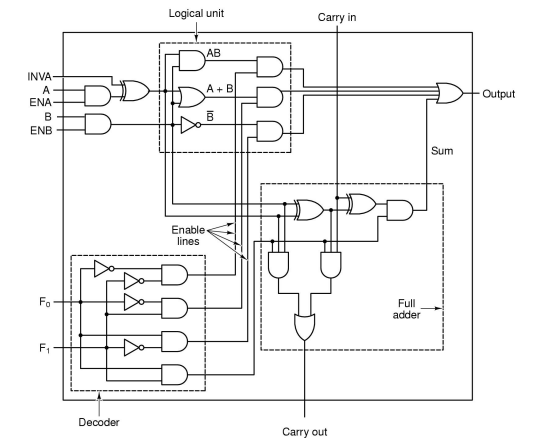
Construir una ALU de 1 bit.

3 entradas: A, B, carry.

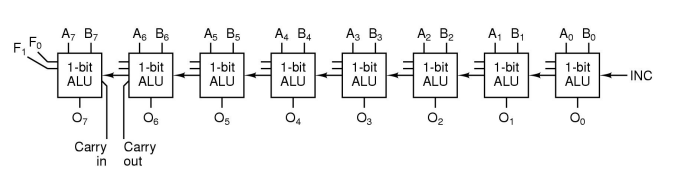
4 operaciones: A.B, A+B, NOT B, Suma(A,B,Carry).

Salidas: Resultado, Carry out.

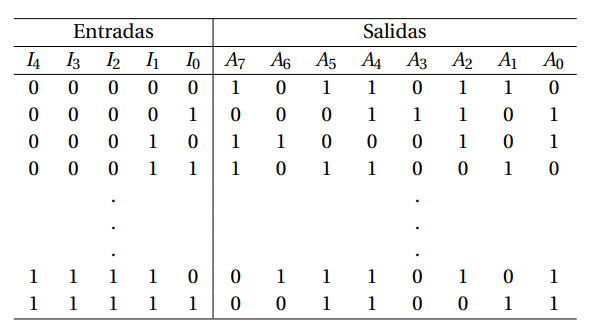
**ALU de 1 bit**



**ALU de 8 bits**



**Memoria ROM**



**ROM con decoder**

