

# Organización de Computadoras



---

## Clase 1



# Bibliografía y web de cátedra

---

- ***Organización y Arquitectura de Computadoras – Diseño para optimizar prestaciones***, Stallings W., Editorial Prentice Hall (5º edición).
- ***Organización de Computadoras***, Tanenbaum A., Editorial Prentice Hall (4º edición).
- ***Estructura de Computadores y Periféricos***, Martínez Durá R. et al., Editorial Alfaomega, 2001.
- ***Arquitectura de Computadores-Un enfoque cuantitativo*** Hennessy & Patterson, Editorial Mc Graw Hill (1º edición).
- **<http://weblidi.info.unlp.edu.ar/catedras/organizacion/>**

# Fechas importantes

## REGIMEN INGRESANTES

**Promoción:** deben Aprobarse con el 70% o mas en la primera fecha.

- 27 de ABRIL - 1<sup>er</sup> PARCIAL (Prácticas 1 y 2)
  - Para los que NO Aprobaron COC
  - Único Recuperatorio 1<sup>er</sup> Parcial: 11 de MAYO
- 29 de MAYO - 2<sup>do</sup> PARCIAL (Prácticas 3 y 4)
  - Para los que Aprobaron COC ó 1<sup>er</sup> parcial
  - Único Recuperatorio 2<sup>do</sup> Parcial: 08 de JUNIO
- 29 de JUNIO - 3<sup>er</sup> PARCIAL (Prácticas 5 y 6)
  - Para los que Aprobaron 1<sup>er</sup> y 2<sup>do</sup> parcial
  - Único Recuperatorio 3<sup>o</sup> Parcial: 06 de JULIO
- Se tomarán en aula y horario de práctica.
  - 13 de JULIO – Evaluación TOTAL
    - En aulas y horarios a establecer



# Promoción. Condiciones y fechas.

---

- Aprobar Parcial 1 y Parcial 2 en primera fecha con el 70% de la nota máxima habilita para rendir
  - Evaluación Corta de Teoría (ETC)
  - MIÉRCOLES 06 de JUNIO
- Aprobar Parcial 3 en primera fecha con el 70% de la nota máxima y la ETC habilita para rendir
  - Evaluación Teórica Promoción
  - MIÉRCOLES 11 de JULIO
- Se tomarán en aula y horario de teoría.



# Repaso Curso de Ingreso

---

- Representación de Datos.
- Números sin signo. BCD.
- Lógica digital. Álgebra de Boole.



# Representación de datos

---

- Las computadoras almacenan datos e instrucciones en memoria
- Para ello utilizan el sistema binario
- Razones :
  - el dispositivo se encuentra en uno de dos estados posibles (0 ó 1)
  - identificar el estado es más fácil si sólo hay dos



# Representación de datos

---

- Ejemplo :
  - lámpara encendida ó apagada
  - lámpara encendida con 10 intensidades distintas
  - Es más fácil conocer el “estado” de la lámpara en el primer caso (encendida ó apagada), que determinar alguna de las 10 intensidades distintas



# Tipos de datos

---

Las computadoras manejan 4 tipos básicos de datos binarios

- Números enteros sin/con signo
- Números reales con signo
- Números decimales codificados en binario (BCD)
- Caracteres





# Representación de números enteros

---

- Sin signo
- Módulo y signo
- Complemento a uno (  $Ca1$  )  
Complemento a la base reducida
- Complemento a dos (  $Ca2$  )  
Complemento a la base
- Exceso



# Números enteros sin signo

---

Si el número tiene  $n$  bits, puedo representar

➤  $2^n =$  números distintos

El rango va desde

➤ 0 a  $(2^n - 1)$



# Números enteros sin signo

---

Ejemplo:  $n = 3$  bits

Decimal	Representación sin signo
0	000
1	001
2	010
..	....
7	111



# Números enteros sin signo

---

Ejemplo:  $n = 8$  bits

0	00000000
..	.....
128	10000000
..	.....
254	11111110
255	11111111



# Números enteros sin signo

---

- RECORDAR: la cantidad de representaciones distintas depende del número de bits

$$\text{N}^{\circ}\text{s distintos} = 2^n$$





# Sistemas Posicionales

---

## Teorema Fundamental de la Numeración

$$N^{\circ} = \sum_{i=-m}^n (\textit{dígito})_i \times (\textit{base})^i$$

$$\dots + x_4 \times B^4 + x_3 \times B^3 + x_2 \times B^2 + x_1 \times B^1 + x_0 \times B^0 + x_{-1} \times B^{-1} + x_{-2} \times B^{-2} + \dots$$

$N^{\circ}$  es el valor decimal de una cantidad expresada en base B y con  $(n+1+m)$  dígitos en posiciones  $i$ .



# Sistema Decimal

---

- Base 10.
- Dígitos  $\{0,1,2,3,4,5,6,7,8,9\}$

$$3574 = 3000 + 500 + 70 + 4$$

$$= 3 \times 10^3 + 5 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

- 3 unidades de mil + 5 centenas + 7 decenas + 4 unidades

$$3.1416_{(10)} = 3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2} + 1 \times 10^{-3} + 6 \times 10^{-4}$$

- 3 unidades + 1 décima + 4 centésimas + 1 milésima + 4 diezmilésimas



# Sistema Binario

---

- Base 2.
- Dígitos {0,1}

$$\begin{aligned} 1001,1_2 &= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} \\ &= 8 + 0 + 0 + 1 + 0,5 \\ &= 9,5_{10} \end{aligned}$$





# Números en punto fijo (1)

---

- Se considera que todos los números a representar tienen exactamente la misma cantidad de dígitos y la coma fraccionaria está siempre ubicada en el mismo lugar.
- En sistema decimal: 0,23 ó 5,12 ó 9,11
  - En los ejemplos cada número tiene tres dígitos, y la coma está a la derecha del mas significativo



# Números en punto fijo (2)

---

- En sistema binario:

$11,10 (3,5)_{10}$  ó  $01,10 (1,5)_{10}$  ó  $00,11 (0,75)_{10}$

- Hay 4 dígitos y la coma está entre el 2<sup>do</sup> y 3<sup>er</sup> dígito.
- La diferencia principal entre la representación en el papel y su almacenamiento en computadora, es que no se guarda coma alguna, se supone que está en un lugar determinado.



# Punto Fijo: Rango y Resolución

---

**Rango:** diferencia entre el número mayor y el menor

**Resolución:** diferencia entre dos números consecutivos

- Para el ejemplo anterior en sistema decimal

Rango es de 0,00 a 9,99 ó  $[0,00...9,99]$

Resolución es 0,01

$$2,32 - 2,31 = 0,01 \quad \text{ó} \quad 9,99 - 9,98 = 0,01$$



## Rango y Resolución(2)

---

- ❖ Notar que hay un compromiso entre rango y resolución.
- ❖ Si mantenemos tres dígitos y desplazamos la coma dos lugares a la derecha, el rango pasa a ser  $[0, \dots, 999]$  y la resolución valdrá 1.

En cualquiera de los casos hay  $10^3$  números distintos



# Ejemplo en binario con 4 bits

---

4 parte ent. y 0 parte frac.

- - - -

Resolución

$$0001 - 0000 =$$

$$0001_2 = 1_{10}$$

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



# Ejemplo en ... (1)

---

3 parte ent. y 1 parte frac.

- - - , -

Resolución

$$000,1 - 000,0 =$$

$$000,1_2 = 0,5_{10}$$

Binario	Decimal
000,0	0
000,1	0,5
001,0	1
001,1	1,5
010,0	2
010,1	2,5
011,0	3
011,1	3,5
100,0	4
100,1	4,5
101,0	5
101,1	5,5
110,0	6
110,1	6,5
111,0	7
111,1	7,5



## Ejemplo en ... (2)

2 parte ent. y 2 parte frac.

- - , - -

Resolución

$$00,01 - 00,00 =$$

$$00,01_2 = 0,25_{10}$$

Binario	Decimal
00,00	0
00,01	0,25
00,10	0,5
00,11	0,75
01,00	1
01,01	1,25
01,10	1,5
01,11	1,75
10,00	2
10,01	2,25
10,10	2,5
10,11	2,75
11,00	3
11,01	3,25
11,10	3,5
11,11	3,75



## Ejemplo en ... (3)

1 parte ent. y 3 parte frac.

- , - - -

Resolución

$$0,001 - 0,000 =$$

$$0,001_2 = 0,125_{10}$$

Binario	Decimal
0,000	0
0,001	0,125
0,010	0,25
0,011	0,375
0,100	0,5
0,101	0,625
0,110	0,75
0,111	0,875
1,000	1
1,001	1,125
1,010	1,25
1,011	1,375
1,100	1,5
1,101	1,625
1,110	1,75
1,111	1,875





## Ejemplo en ... (4)

parte ent. y 4 parte frac.

, - - - -

Resolución

$$,0001 - ,0000 =$$

$$,0001_2 = 0,0625_{10}$$

Binario	Decimal
,0000	0
,0001	0,0625
,0010	0,125
,0011	0,1875
,0100	0,25
,0101	0,3125
,0110	0,375
,0111	0,4375
,1000	0,5
,1001	0,5625
,1010	0,625
,1011	0,6875
,1100	0,75
,1101	0,8125
,1110	0,875
,1111	0,9375



# Representación y error

---

- Al convertir un número decimal a sistema binario tendremos 2 casos:
  - Sin restricción en la cantidad de bits a usar
    - $3,125_{10} = 11,001_2$
  - Con restricción, por ejemplo 3 bits para parte entera y 4 bits para parte fraccionaria
    - $3,125_{10} = 011,0010_2$

No cometemos error



# Representación y error (2)

- Convertir  $3,2_{10}$  con distintas restricciones
  - 3 bits para parte fraccionaria:  $011,001_2 = 3,125_{10}$ 
    - Error =  $3,2 - 3,125 = 0,075$
  - 4 bits para parte fraccionaria:  $011,0011_2 = 3,1875_{10}$ 
    - Error =  $3,2 - 3,1875 = 0,0125$
  - 5 bits para parte fraccionaria:  $011,00111_2 = 3,21875_{10}$ 
    - Error =  $3,2 - 3,21875 = -0,01875$
- El error más pequeño es 0,0125 entonces 3,1875 es la representación más cercana a 3,2 y podría utilizar sólo 4 bits para la parte fraccionaria.



# Operaciones aritméticas

---

- Suma en binario
  - Al ser un sistema posicional la suma es como en decimal con acarreos entre posiciones al superar el máximo valor representable con un dígito
    - Ej:  $1+1=10$  (ó '1 y me llevo 1')
  - Valores mas grandes requieren mas bits
- Hasta ahora sólo representamos valores en binario sin signo (que llamamos BSS).
  - Las restas se podrán realizar si acomodamos los operandos de modo tal que resultado sea mayor que cero, sino deberemos 'pedir prestado'.



# Bits de condición (banderas)

---

- ✓ Son bits que el procesador establece de modo automático acorde al resultado de cada operación realizada.
- ✓ Sus valores permitirán tomar decisiones como:
  - ✓ Realizar o no una transferencia de control.
  - ✓ Determinar relaciones entre números (mayor, menor, igual).



# Banderas aritméticas

---

- ❖ Z (cero): vale 1 si el resultado de la operación son todos bits 0.
- ❖ C (carry): en la suma vale 1 si hay acarreo del bit más significativo; en la resta vale 1 si hay 'borrow' hacia el bit más significativo.
  - ❖ Cuando la operación involucra números sin signo,  $C=1$  indica una condición fuera de rango.



# Sistema Hexadecimal

---

- Base 16.
- Dígitos {0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F}  
10,11,12,13,14,15

$$\begin{aligned} 2CA,8_{16} &= 2 \times 16^2 + C \times 16^1 + A \times 16^0 + 8 \times 16^{-1} \\ &= 512 + 192 + 10 + 0,5 \\ &= 714,5_{10} \end{aligned}$$



# Sistema hexadecimal codificado en binario (BCH)

---

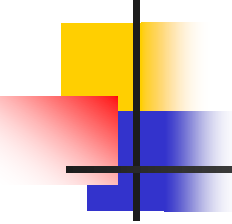
- Los dígitos hexadecimales se convierten uno a uno en binario
- Para representar un dígito hexadecimal se utilizará siempre 4 bits
- Se asocia cada dígito con su valor en binario puro





# BCH

Dígito hexadecimal	Código BCH
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111



# Sistema decimal codificado en binario (BCD)

---

- Los dígitos decimales se convierten uno a uno en binario
- Para representar un dígito decimal se requerirán 4 bits
- Se asocia cada dígito con su valor en binario puro



# BCD

---

<b>Dígito decimal</b>	<b>Código BCD</b>
<b>0</b>	<b>0000</b>
<b>1</b>	<b>0001</b>
<b>2</b>	<b>0010</b>
<b>3</b>	<b>0011</b>
<b>4</b>	<b>0100</b>
<b>5</b>	<b>0101</b>
<b>6</b>	<b>0110</b>
<b>7</b>	<b>0111</b>
<b>8</b>	<b>1000</b>
<b>9</b>	<b>1001</b>



# BCD

---

- BCD tiene dos ámbitos de aplicación:
  - E/S y periféricos, los números se codifican usando un byte por dígito. Se dice que el número está *desempaquetado*.
  - En cálculo, se reservan 4 bits por dígito. Se dice que el número está *empaquetado*.



# BCD

---

- Ejemplo: desempaquetado sin signo

$$\begin{aligned} 834 &= 11111000 \ 11110011 \ 11110100 \\ &= F8 \ F3 \ F4 \end{aligned}$$

- Por cada dígito se usan 8 bits, 4 para el binario puro y 4 se completan con "1"



# BCD

---

- Desempaquetado con signo
- Con 4 bits hay  $2^4=16$  combinaciones posibles de unos y ceros :
  - Diez usamos para los dígitos 0 al 9
  - Nos quedan seis sin usar
  - $C_{16} = 1100$  representa al signo +
  - $D_{16} = 1101$  representa al signo -



# BCD

---

- Ejemplo: desempaquetado con signo

$$\begin{aligned} + 834 &= 11111000 \ 11110011 \ 11000100 \\ &= F8 \ F3 \ C4 \end{aligned}$$

- Los 4 bits que acompañan al último dígito son reemplazados por el signo.



# BCD

---

Ejemplo:

$$\begin{aligned} \blacksquare \quad - 834 &= 1111\textcolor{red}{1}000 \quad 1111\textcolor{green}{0}011 \quad 1101\textcolor{blue}{0}100 \\ &= \textcolor{red}{F}8 \textcolor{green}{F}3 \textcolor{blue}{D}\textcolor{yellow}{4} \end{aligned}$$





# BCD

---

Ejemplo: empaquetado con signo

$$\blacksquare + 834 = 10000011 \ 01001100$$

$$= 83 \ 4C$$

$$\blacksquare - 34 = 00000011 \ 01001101$$

$$= 03 \ 4D$$



# Suma en BCD

---

- De las 16 representaciones posibles con 4 bits, usamos 10 para los dígitos 0 al 9
- Nos sobran 6 combinaciones de 4 bits
- Al sumar dos dígitos BCD, se nos presentan dos casos :
  - ❖ la suma es  $\leq 9$
  - ❖ la suma es  $> 9$



# Suma en BCD

---

- En el primer caso no hay problema

$$\begin{array}{r} + \quad 41 \\ \quad 22 \\ \hline 63 \end{array}$$

$$\begin{array}{r} + \quad 0100 \quad 0001 \\ \quad 0010 \quad 0010 \\ \hline 0110 \quad 0011 \end{array}$$



# Suma en BCD

- En el segundo caso ¿Qué sucede ?

$$\begin{array}{r} 1 \\ + 15 \\ + 27 \\ \hline 42 \end{array}$$

$$\begin{array}{r} 111 \\ + 0001 \ 0101 \\ + 0010 \ 0111 \\ \hline 0011 \ 1100 \\ \underbrace{\hspace{1cm}}_3 \quad \underbrace{\hspace{1cm}}_{\text{no válido}} \end{array}$$



# Suma en BCD

---

- Cuando la suma de los dos dígitos da  $>9$  hay que generar el “acarreo” porque hay seis combinaciones no usadas
- Entonces: cuando la suma de los dígitos es  $> 9$  hay que sumar 6 en ese dígito



# Suma en BCD

$$\begin{array}{r} 1 \\ + 15 \\ + 27 \\ \hline 42 \end{array}$$

$$\begin{array}{r} 111 \\ + 0001 \ 0101 \\ + 0010 \ 0111 \\ \hline 111 \ 1 \\ 0011 \ 1100 \\ + \phantom{00} \ 0110 \quad \leftarrow 6 \\ \hline 0100 \ 0010 \end{array}$$

Resultado correcto



# Suma en BCD

- Ejemplo

$$\begin{array}{r} 1 \\ 476 \\ + 55 \\ \hline 531 \end{array}$$

$$\begin{array}{r} 111 \quad 1 \\ 0100 \quad 0111 \quad 0110 \\ + \quad 0101 \quad 0101 \\ \hline 0100 \quad 1100 \quad 1011 \\ + \quad 0110 \quad 0110 \\ \hline 0101 \quad 0011 \quad 0001 \end{array}$$



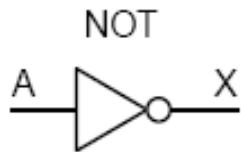
# El nivel de lógica digital

---

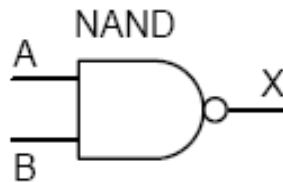
- Un circuito digital es en el que están presentes dos valores lógicos
- Compuertas son dispositivos electrónicos que pueden realizar distintas funciones con estos dos valores lógicos
- Como vimos en el Ingreso las compuertas básicas son: AND, OR, NOT, NAND, NOR y XOR



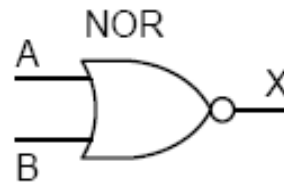
# Compuertas: símbolo y descripción funcional



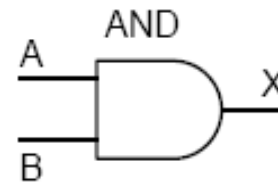
A	X
0	1
1	0



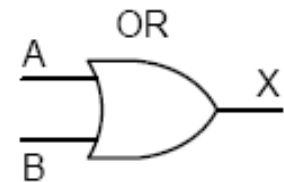
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0



A	B	X
0	0	1
0	1	0
1	0	0
1	1	0



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1



# Algebra Booleana

---

- Para describir los circuitos que pueden construirse combinando compuertas, se requiere un nuevo tipo de álgebra, donde las variables y funciones sólo puedan adoptar valores 0 ó 1: álgebra booleana.



# Algebra Booleana

---

- Puesto que una función booleana de  $n$  variables tiene  $2^n$  combinaciones de los valores de entrada, la función puede describirse totalmente con una tabla de  $2^n$  renglones, donde c/u indica un valor de la función (0 ó 1) para cada combinación distinta de las entradas:

**$\Rightarrow$  *tabla de verdad***



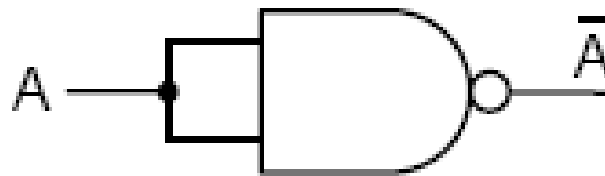
## Recordemos algunas identidades del álgebra booleana

Identidad	$1.A=A$	$0+A=A$
Nula	$0.A=0$	$1+A=1$
Idempotencia	$A.A=A$	$A+A=A$
Inversa	$A.\overline{A}=0$	$A+\overline{A}=1$
Conmutativa	$A.B=B.A$	$A+B=B+A$
Asociativa	$(AB).C=A(BC)$	$(A+B)+C=A+(B+C)$
Distributiva	$A+B.C=(A+B).(A+C)$	$A.(B+C)=AB+AC$
Absorción	$A.(A+B)=A$	$A+A.B=A$
De Morgan	$\overline{A.B}=\overline{A}+\overline{B}$	$\overline{A+B}=\overline{A}.\overline{B}$

# Leyes de De Morgan

➤ Ejemplo: construir un NOT con NAND

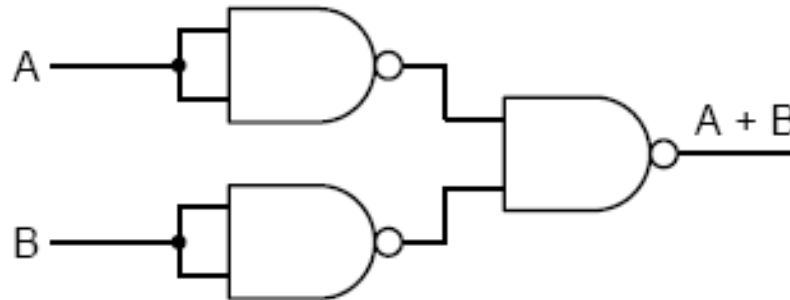
$$F = \overline{A \cdot B} = \overline{A \cdot A} = \overline{A}$$



# Leyes de De Morgan

- Ejemplo: construir un OR con NAND

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





# Implementación de funciones booleanas

---

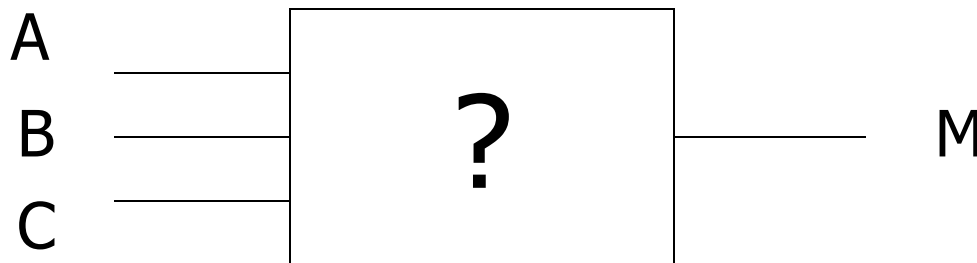
- Escribir la tabla de verdad para la función
- Dibujar una AND para cada término que tiene un 1 en la columna de resultado (con sus entradas apropiadas )
- Invertir las entradas necesarias
- Unir todas las AND a una OR



# Implementación

---

Ejemplo: construir la tabla de verdad e implementar el circuito de una función booleana  $M$ , de tres entradas  $A$ ,  $B$  y  $C$ , tal que  $M=1$  cuando la cantidad de '1' en  $A$ ,  $B$  y  $C$  es  $\geq 2$  y  $M=0$  en otro caso.







# Tabla de verdad

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1





## Función M

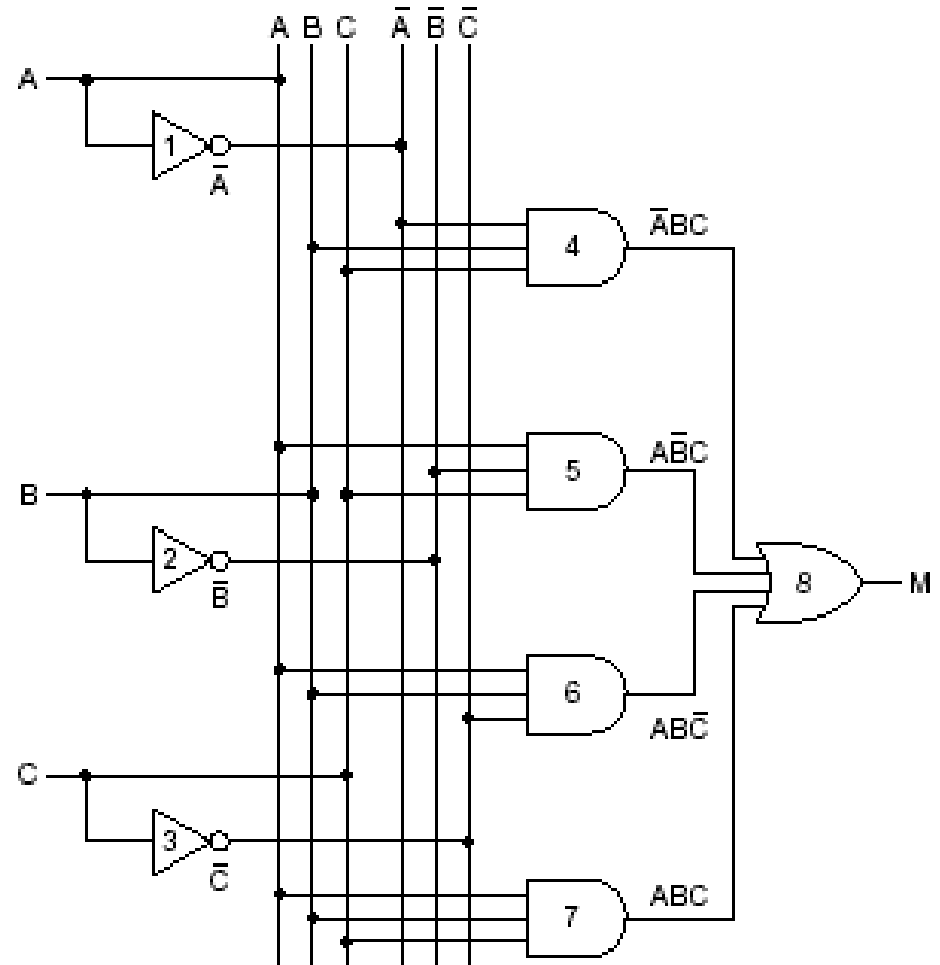
---

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

- ❖ Hay tantos términos como 1s en la tabla
- ❖ Cada término vale 1 para una única combinación de A, B y C
- ❖ Las variables que valen 0 en la tabla aparecen aquí negadas

# Función M (2)

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





# Otro ejemplo

**Supongamos la siguiente Tabla de Verdad**

A	B	M
0	0	0
0	1	1
1	0	1
1	1	0

**Función**      **$M = \bar{A}B + A\bar{B} \Rightarrow M = A \text{ XOR } B$**



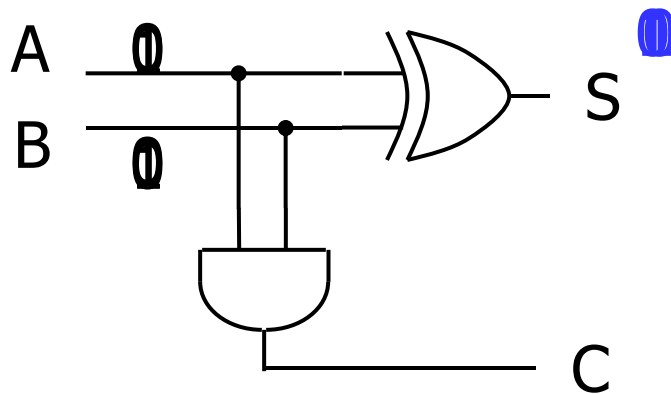
# Recordemos

---

- ✓ En un AND, basta que una de sus entradas sea 0 para que la función valga 0.
- ✓ En un OR, basta que una de sus entradas sea 1 para que la función valga 1.
- ✓ Hacer el XOR con 1 invierte el valor de la variable.
- ✓ Hacer el XOR con 0 deja el valor de la variable como estaba.

# Circuitos combinatorios

## ■ Ejemplo



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

**S** representa la suma aritmética de 2 bits y **C** es el acarreo

**Semi-sumador ó Half adder**



# mayor información ...

---

- Sistemas enteros y Punto fijo
  - Apunte 1 de Cátedra
- Operaciones lógicas
  - Apunte 3 de Cátedra
- Apéndice 8A: Sistemas de Numeración
  - Stallings, 5º Ed.
- Apéndice A: Lógica digital (A.1., A.2.)
  - Stallings, 5º Ed.
- Capítulo 3: Lógica digital y representación numérica
  - Apuntes COC – Ingreso