



# *Introducción al Diseño Lógico (E0301)*

*Ingeniería en Computación*

*Gerardo Sager*

*Clase 2 curso 2025*

- Conversión entre sistemas numéricos
  - Decimal, binario, hexadecimal.
- Ventajas del uso de hexadecimal
  - Conteo en hexadecimal.
- Representación de números decimales usando código BCD.
- Códigos Alfanuméricos: código ASCII.
- Paridad y su utilización para la detección de errores.

## ● Conversión Binaria a Decimal

- Convertir de binario a decimal sumando las posiciones que contienen 1 con su peso correspondiente:

$$11011_2$$

$$\begin{aligned} 11011_2 &= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = \\ &= 16 + 8 + 2 + 1 = 27 \end{aligned}$$

- Un ejemplo con más bits

$$10110101_2$$

$$\begin{aligned} 10110101_2 &= 1 \times 2^7 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^0 = \\ &= 128 + 32 + 16 + 4 + 1 = 181 \end{aligned}$$

## ● Conversión Binaria a Decimal

- Los números binarios pueden convertirse con el método “double-dabble”

Given:	1	1	0	1	$1_2$
Results:	$1 \times 2 = 2$				
	$+ 1$				
	$3 \times 2 = 6$				
	$+ 0$				
	$6 \times 2 = 12$				
	$+ 1$				
	$13 \times 2 = 26$				
	$+ 1$				
	$27_{10}$				

El método double-dabble permite la conversión sin hacer la suma de grandes números

# Conversión Binaria a Decimal

Los números binarios pueden convertirse a decimal con el método “double-dabble”

$$\begin{array}{rcll} \text{Dado:} & 1 & 1 & 0 & 1 & 1 \\ \text{Resultado} & 1 & \times 2 = & 2 & & \\ & & + & 1 & & \\ & & \hline & 3 & \times 2 = & 6 & & \\ & & & + & 0 & \\ & & & \hline & & 6 & \times 2 = & 12 & \\ & & & & + & 1 & \\ & & & & \hline & & & 13 & \times 2 = & 26 & \\ & & & & & + & 1 & \\ & & & & & \hline & & & & & 27_{10} \end{array}$$

El método double-dabble permite la conversión sin hacer la suma de grandes números

# Conversión Binaria a Decimal

- Proceso inverso a la suma de potencias pesadas.
- Debo tener una tabla de potencias de 2 expresadas en decimal.
- Me fijo cual es el mayor número de la tabla, contenido en el número a convertir.
- Resto del número a convertir el valor obtenido
- Repito el proceso hasta llegar a 0
- Establezco en 1 el bit correspondiente si usé el valor de la tabla y en 0 si no lo usé

$2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
3	1	8	4	2	1	5	2	1	6	3	1	8	4	2	1
2	6	1	0	0	0	1	5	2	4	2	6				
7	3	9	9	4	2	2	6	8							
6	8	2	6	8	4										
8	4														

Ejemplo:  $314_{10} = ?$   $\rightarrow 314 - 256 = 58$  ( $2^8$ )  $\rightarrow 58 - 32 = 26$  ( $2^5$ )  $\rightarrow 26 - 16 = 10$  ( $2^4$ )  
 $\rightarrow 10 - 8 = 2$  ( $2^3$ )  $\rightarrow 2 - 2 = 0$  ( $2^1$ )  $\rightarrow 314_{10} = 100111010_2$

## División Repetida

- Dividir el número decimal por 2.
- Escribir el resto de cada división y volver a dividir el cociente x 2 hasta que se obtenga un cociente = 0
- El primer resto es el LSB
- El último resto es el MSB

$$314/2 \quad Q=157 \quad R=0 \text{ (LSB)}$$

$$157/2 \quad Q=78 \quad R=1$$

$$78/2 \quad Q=39 \quad R=0$$

$$39/2 \quad Q=19 \quad R=1$$

$$19/2 \quad Q=9 \quad R=1$$

$$9/2 \quad Q=4 \quad R=1$$

$$4/2 \quad Q=2 \quad R=0$$

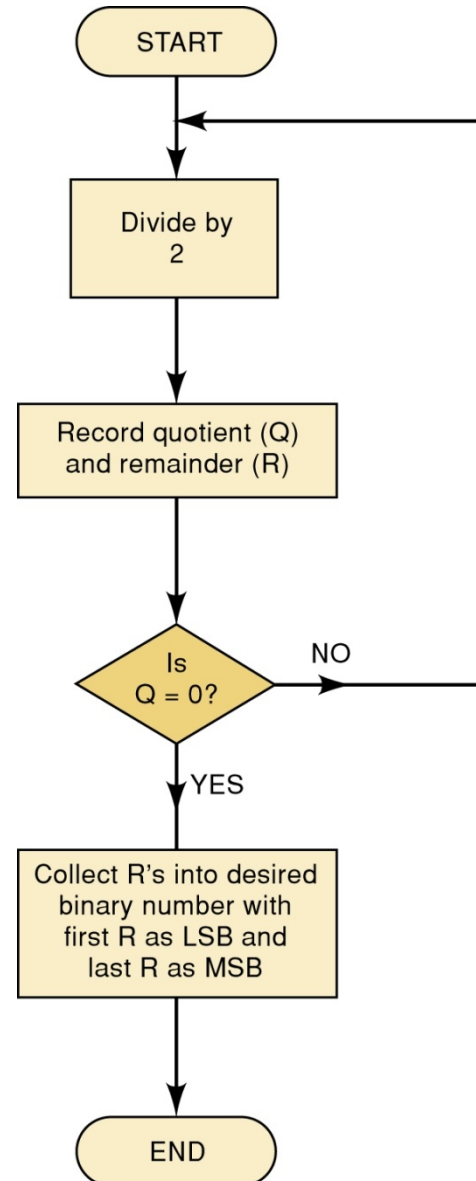
$$2/2 \quad Q=1 \quad R=0$$

$$1/2 \quad Q=0 \quad R=1 \text{ (MSB)}$$

$$314_{10} = 100111010_2$$

## División Repetida

- Este diagrama de flujo describe el proceso a realizar para convertir de Decimal a Binario.
- Si en vez de dividir por 2 se divide por N, me permite convertir de decimal a base N





# Sistema de Numeración Hexadecimal

- Hexadecimal permite un manejo conveniente de cadenas binarias largas, ya que 4 bits agrupados pueden representarse como un único dígito hexadecimal.(base 16)
- Los símbolos que representan los dígitos hexadecimales, son 0-9 y A-F
- La representación puede ser entera o fraccionaria.

$16^4$	$16^3$	$16^2$	$16^1$	$16^0$	$16^{-1}$	$16^{-2}$	$16^{-3}$	$16^{-4}$
--------	--------	--------	--------	--------	-----------	-----------	-----------	-----------

Hexadecimal point

**Relación entre números  
hexadecimales,  
decimales, y binarios.**

Hexadecimal	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

- Convertir de hexa a decimal multiplicando cada dígito por su peso posicional.
- En el siguiente ejemplo, E se sustituye por 14 y A por 10 para realizar la suma pesada.

$$E2A_{16} = E \times 256 + 2 \times 16 + 10 = \dots$$

**Para practicar, verificar que  $3BC2_{16}$  is igual a ....**

- Se pueden usar varios métodos: convertir a binario y luego agrupar en dígitos hexadecimales.
- Usar el método de división repetida
  - Dividir el número decimal por 16
  - El primer resto es el LSB
  - El último es el MSB.
- Ejemplo:
- $100000_{10}$

## ● Sistema de Numeración Hexadecimal – Hexa a Binario

- Se convierte cada dígito Hexa directamente a binario y se reescribe.
- Si es necesario se pueden agregar ceros a la izquierda para completar una palabra
- $19F2_{16} = 0001\ 1001\ 1111\ 0010$

**Para practicar verificar que  $BA6_{16} = 101110100110_2$**

- Convertir de binario a Hexa, agrupando bits de a cuatro comenzando por el LSB.
  - Luego cada grupo se escribe como un único dígito hexa equivalente

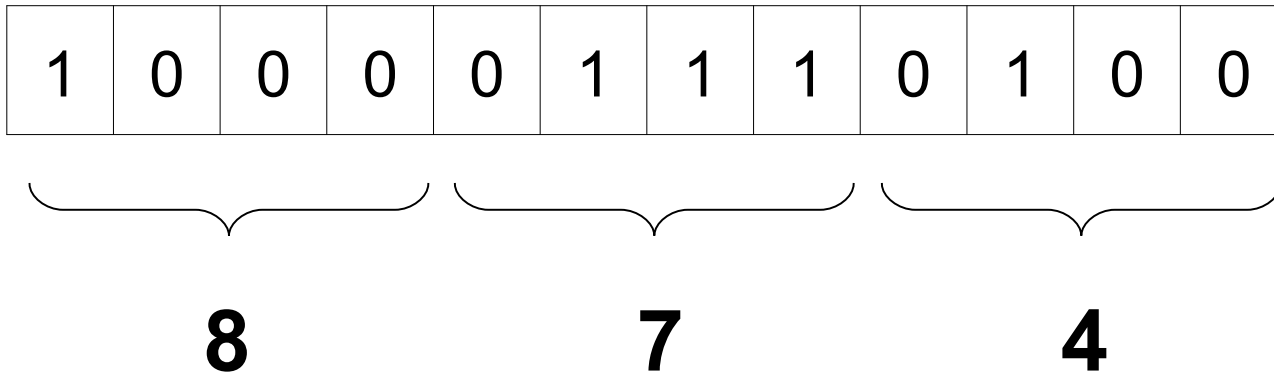
$$1110100110_2 = \underbrace{0011}_3 \underbrace{1010}_A \underbrace{0110}_6 = 3A6_{16}$$

**Verificar que  $101011111_2 = \dots$**

- La codificación BCD se usa ampliamente, sobre todo en equipos de bajo costo, para representar números decimales en forma binaria
  - Combina características de los sistemas decimal y binario.
    - Cada dígito decimal se convierte a su equivalente binario
- **BCD NO ES** un sistema numérico.
  - Es un número decimal, con cada dígito codificado como su equivalente en binario.
- Un número BCD no es lo mismo que un número binario.
  - La ventaja principal de BCD es que resulta fácil convertir de BCD a decimal y viceversa.

# Codificación BCD

- Convertir el número  $874_{10}$  a BCD:
  - Cada dígito se representa mediante 4 bits.
  - El número representado mediante esos 4 bits no puede ser superior a 9



- Para convertir de BCD a decimal se realiza el proceso inverso:
  - Se agrupan los bits de a 4.
  - Se determina el valor decimal de cada grupo de 4 bits, que deben estar comprendidos entre 0 y 9



- Comparación entre BCD y Binario.
  - Cual es el rango de valores que pueden representarse en BCD con 16 bits?
  - Cual es el rango de valores que pueden representarse en binario con 16 bits?
- Convertir 0110100000111001 a su equivalente decimal considerando primero que es BCD y luego que es binario.
  - Cuál de las dos conversiones es mas sencilla?

## Byte, Nibble, Word

- La mayoría de las Microcomputadoras maneja y almacena datos binarios en grupos de ocho bits.
  - 8 bits = 1 **byte**.
    - Un byte puede representar distintos tipos de datos o información.
- Frecuentemente los números binarios se agrupan en grupos de cuatro bits.
  - Un grupo de cuatro bits suele llamarse **nibble** (del inglés nibble = mordisquito).
- **Word** o **palabra** es un grupo de bits que representa una cierta unidad de información.
  - El **tamaño de palabra (Word size)** puede ser definido de distintas maneras en distintos sistemas. Lo más común es definirlo como la cantidad de bits en la palabra que utiliza el sistema digital para operar.
  - La longitud de palabra en una PC actual es 64 bits.
  - La longitud de palabra en un 80x86 es 16 bits.
  - La longitud de palabra en un sistema basado en ARM es 32 bits.

## ● Códigos alfanuméricos

- Representan caracteres, dígitos, símbolos y valores que pueden interpretarse como comandos o funciones.
- Normalmente se encuentran en el teclado de una computadora.
- Hace falta definir al menos 26 minúsculas, 26 Mayúsculas, 10 dígitos, 7 signos de puntuación y entre 20 y 40 caracteres adicionales.
- En distintos idiomas, puede ser necesario agregar más letras o símbolos
- ASCII : Está definido con 128 valores que pueden representarse con 7 bits.
- ASCII Extendido: Posee características que se adaptan a distintos idiomas y consta de 256 valores que pueden representarse con 8 bits.

## ASCII

Character	HEX	Decimal	Character	HEX	Decimal	Character	HEX	Decimal	Character	HEX	Decimal
NUL (null)	0	0	Space	20	32	@	40	64	.	60	96
Start Heading	1	1	!	21	33	A	41	65	a	61	97
Start Text	2	2	"	22	34	B	42	66	b	62	98
End Text	3	3	#	23	35	C	43	67	c	63	99
End Transmit.	4	4	\$	24	36	D	44	68	d	64	100
Enquiry	5	5	%	25	37	E	45	69	e	65	101
Acknowledge	6	6	&	26	38	F	46	70	f	66	102
Bell	7	7	`	27	39	G	47	71	g	67	103
Backspace	8	8	(	28	40	H	48	72	h	68	104
Horiz. Tab	9	9	)	29	41	I	49	73	i	69	105
Line Feed	A	10	*	2A	42	J	4A	74	j	6A	106
Vert. Tab	B	11	+	2B	43	K	4B	75	k	6B	107
Form Feed	C	12	,	2C	44	L	4C	76	l	6C	108
Carriage Return	D	13	-	2D	45	M	4D	77	m	6D	109
Shift Out	E	14	.	2E	46	N	4E	78	n	6E	110

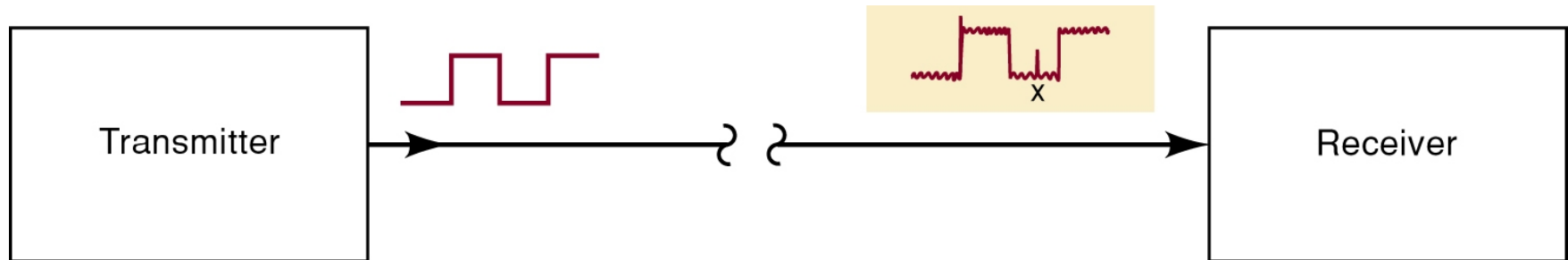
## Códigos alfanuméricos

- Convertir la siguiente sentencia de C a código ASCII. Escribir en decimal, hexadecimal y binario  
`if (x>3)`

Ascii	<b>i</b>	<b>f</b>		<b>(</b>	<b>x</b>	<b>&gt;</b>	<b>3</b>	<b>)</b>
Hexadecimal	<b>69</b>	<b>66</b>	<b>20</b>	<b>28</b>	<b>78</b>	<b>3E</b>	<b>33</b>	<b>29</b>
Decimal	<b>109</b>	<b>102</b>	<b>32</b>	<b>40</b>	<b>120</b>	<b>62</b>	<b>51</b>	<b>41</b>
Binario	<b>01101001</b>	<b>01100110</b>	<b>00100000</b>	<b>00101000</b>	<b>01111000</b>	<b>00111110</b>	<b>00110011</b>	<b>00101001</b>

## 2-9 Método de la Paridad para detectar errores

- El ruido eléctrico puede causar errores durante la transmisión.
  - Ruido? Fluctuaciones espurias y aleatorias en el voltaje o corriente. Aparecen en TODOS los sistemas electrónicos
  - NO puede eliminarse



- Muchos sistemas digitales emplean métodos para la detección e errores y algunas veces también su corrección.
- Uno de los sistemas de control de errores más simples y más utilizados es el método de PARIDAD.

- El método de paridad para la detección de errores requiere la adición de un bit extra al grupo de bits a transmitir.
  - Es llamado el bit de paridad.
  - Puede ser 0 o 1, dependiendo del número de 1s en el grupo de bits a transmitir..
- Hay dos métodos de paridad, PARIDAD PAR y PARIDAD IMPAR.
  - El transmisor y el receptor deben utilizar el mismo tipo de paridad.

- MÉTODO DE PARIDAD PAR:
  - Se basa en completar el grupo de bits a transmitir con el bit de paridad de manera tal que la cantidad total de “1” que tenga el conjunto sea PAR.
    - La cantidad de bits en 1 del (grupo de bits a transmitir + bit de paridad) debe ser par.
  - Se cuenta la cantidad de “1” que tiene la palabra
  - Si la cantidad es par, el bit de paridad se establece en 0
  - Si la cantidad es impar el bit de paridad se establece en 1
  - En ambos casos la cantidad TOTAL de bits en “1” debe ser par
  - Ejemplo:

Grupo de bits a transmitir: 1000011

Bit de Paridad Par?



- MÉTODO DE PARIDAD IMPAR:
  - Se basa en completar el grupo de bits a transmitir con el bit de paridad de manera tal que la cantidad total de “1” que tenga la el conjunto sea IMPAR.
    - La cantidad de bits en 1 del (grupo de bits a transmitir + bit de paridad) debe ser impar.
  - Se cuenta la cantidad de “1” que tiene la palabra
  - Si la cantidad es par, el bit de paridad se establece en 1
  - Si la cantidad es impar el bit de paridad se establece en 0
  - En ambos casos la cantidad TOTAL de bits en “1” debe ser IMPAR
  - Ejemplo:

Grupo de bits a transmitir: 1000011

Bit de Paridad Impar?

## ● Metodos de Paridad para detección de errores

- En ambos casos el bit de paridad se AGREGA al grupo original de bits a transmitir.
- El grupo de bits junto con la paridad, constituye la palabra de CÓDIGO.
- Por ejemplo si quiero transmitir palabras codificadas en ASCII con paridad PAR, la palabra de código tendrá 8 bits, 7 de los datos en ASCII y uno de paridad.

## ● Metodos de Paridad para detección de errores

- Codificar HOLA MUNDO en ASCII
- Agregar los bits de paridad PAR (como MSB) correspondientes a cada símbolo ASCII.(PP)
- Repetir para paridad impar (PI)

H	O	L	A		M	U	N	D	O	
48	4F	4C	41	20	4D	55	4E	44	4F	ASCII
48	CF	CC	41	A0	4D	55	4E	44	CF	ASCII + PP
C8	4F	4C	C1	20	CD	D5	CE	C4	4F	ASCII + PI

- Repasamos los métodos de conversión entre bases numéricas
- Vimos un método de representación de dígitos decimales llamado BCD.
- Vimos un sistema de detección de errores llamado paridad
  - Podemos utilizar dos tipos de control de paridad.
    - Paridad Par
    - Paridad Impar

- Representación binaria con signo
  - Signo y Magnitud / Signo y Módulo (SM)
  - Complemento a 1 (CA1)
  - Complemento a 2 (CA2)
- Rangos de representación.
- Ventajas y desventajas entre los tres sistemas de representación.
- Adición, substracción multiplicación y división binarias en CA2.

# Representación binaria con signo

- Para poder representar los números con signo utilizando solamente dos símbolos (0 y 1) es necesario codificar el signo del número:
  - Normalmente se codifica el signo con un 0 para números positivos y un 1 para números negativos
  - Se utilizan 3 diferentes sistemas de representación, que varían según las diferentes formas de codificar la magnitud:
    - Signo y magnitud (SM)
    - Complemento a 1 (CA1)
    - Complemento a 2 (CA2)

## SIGNO Y MAGNITUD

- Dado que sólo es posible representar la **magnitud** con un número binario, el signo (+) o (-) puede mostrarse agregando un **bit de signo**.
  - El bit de signo 0 indica un número positivo
  - El bit de signo 1 indica un número negativo
  - Ejemplo:

$$25_{10} = 11001_{\text{BIN}} \text{ (Binario Natural)}$$

$$+25_{10} = 011001_{\text{SM}} \text{ (Módulo y signo)}$$

$$-25_{10} = 111001_{\text{SM}} \text{ (Módulo y signo)}$$

# Representación binaria con signo

## COMPLEMENTO a 1

- Si el número es positivo:
  - El MSB es un 0 (signo)
  - El resto de los bits son la magnitud en binario natural
- Si el número es negativo:
  - El MSB es un 1 (signo)
  - El resto de los bits son el complemento (a 1) de la magnitud

$$\text{Ca1}(A) = \neg A$$

**La operación  $\neg A$  Se lee como “complemento a 1 de A” o más comunmente “A negado”. Significa que se invierten todos los bits de A, incluyendo el de signo.**

*Equivalentemente, se puede definir como (siendo n el número de bits):*

$$\text{Ca1}(A) = 2^n - 1 - A = (2^n - 1) - A = 11\dots 11 - A = \neg A$$

- Ejemplos:
  - $+25_{10} = 011001_{\text{Ca1}}$  (Complemento a 1)
  - $-25_{10} = 100110_{\text{Ca1}}$  (Complemento a 1)



# Representación binaria con signo

## COMPLEMENTO a 2

- Si el número es positivo:
  - El MSB es un 0 (signo)
  - El resto de los bits son la magnitud en binario natural
- Si el número es negativo:
  - El MSB es un 1 (signo)
  - El resto de los bits son el complemento a 2 de la magnitud.
  - El complemento a dos de un número es su complemento a 1, + 1

$$\mathbf{Ca2(A) = Ca1(A)+1 = \neg A+1}$$

*Equivalentemente, se puede definir como (siendo  $n$  el número de bits):*

$$\mathbf{Ca2(A) = 2^n - A = 2^n - A + 1 - 1 = (2^n - 1) + 1 - A = 11\dots 11 - A + 1 = \neg A + 1}$$

Ejemplos:

$$+25_{10} = 011001_{Ca2}$$

$$-25_{10} = 100111_{Ca2}$$

# Representación binaria con signo

## COMPLEMENTO a 2

- No deben confundirse los siguientes conceptos
- “Operación de complementar a 2”:
  - $Ca2(A) = Ca1(A)+1 = \neg A+1$
- “Representación en complemento a 2”
  - Es el valor que representa un numero dado, que se obtiene siguiendo las siguientes reglas que permiten representar dos conjuntos de números:
    - Si el número es positivo, se utiliza la representación binaria natural, si el número se va a representar con n bits, el MSB debe ser “0”.
    - Si el numero a representar es negativo, se escribe su valor positivo y se aplica la operación de complementar a 2.
- La unión de ambos conjuntos es la representación en complemento a 2 utilizando n bits.
- La representación Complemento a 2 es la más utilizada en sistemas digitales.

# Representación binaria con signo

## COMPLEMENTO a 2

- Otra forma de realizar la operación de complementar a 2 es la siguiente:
- Comenzando por el LSB, copiar los bits hasta encontrar el primer 1 inclusive, e invertir el resto:
- EJEMPLO:

$$\text{Ca2}(110001\textcolor{red}{10}) = 001110\textcolor{red}{10}$$

Comprobación:

$$\text{Ca2}(11000110) = 00111001 + 1 = 00111010$$

# Representación binaria con signo

## Extensión del número de bits

- Un mismo número puede representarse con diferente número de bits:

$$+25_{10} = 011001_{SM} = 0\textcolor{red}{00}11001_{SM} = 0\textcolor{red}{000000000000}11001_{SM}$$

$$+25_{10} = 011001_{CA1} = 0\textcolor{red}{00}11001_{CA1} = 0\textcolor{red}{000000000000}11001_{CA1}$$

$$+25_{10} = 011001_{CA2} = 0\textcolor{red}{00}11001_{CA2} = 0\textcolor{red}{000000000000}11001_{CA2}$$

6 bits

8 bits

16 bits

- Si los números son positivos como en el ejemplo, en los tres sistemas se añaden ceros a la izquierda (en rojo) hasta completar el número de bits necesarios
- Nótese que el MSB se mantiene en 0 indicando el signo (positivo)

# Representación binaria con signo

## Extensión del número de bits

- En el caso de Números Negativos:
- Para representación SM, mantener el MSB que indica el signo y añadir ceros para completar el número de bits.
- Para representación Ca1 y Ca2 añadir unos a la izquierda del número hasta completar la cantidad de bits necesarios

$$-25_{10} = 111001_{SM} = 1\textcolor{red}{00}11001_{SM} = 1\textcolor{red}{0000000000}11001_{SM}$$

$$-25_{10} = 100110_{CA1} = \textcolor{red}{11}100110_{CA1} = \textcolor{red}{1111111111}100110_{CA1}$$

$$-25_{10} = 100111_{CA2} = \textcolor{red}{11}100111_{CA2} = \textcolor{red}{1111111111}100111_{CA2}$$

6 bits

8 bits

16 bits

# Representación binaria con signo

## Comparación:

- Vemos cómo la misma codificación en 3 bits representa distintos números en los tres sistemas
- Puede notarse que SM y Ca1 tienen dos representaciones del cero mientras que en Ca2 la representación del cero es única

Codificación	SM	Ca1	Ca2
000	0	0	0
001	1	1	1
010	2	2	2
011	3	3	3
100	-0	-3	-4
101	-1	-2	-3
110	-2	-1	-2
111	-3	-0	-1

# Rangos de representación.

**Cuál es el rango de valores que puede representarse con 8 bits?**

- **Binario sin signo:**  $2^n = 2^8 = 256$  valores distintos (0 a 255)

$$00000000 = 0_{10} \text{ al } 11111111 = 255_{10}$$

- **Magnitud y signo:**  $2^n - 1 = 2^8 - 1 = 255$  valores distintos  
(-127  $\rightarrow$  -0 y 0  $\rightarrow$  127)

$$11111111 = -127_{10} \rightarrow 10000000 = -0_{10}$$

$$00000000 = 0_{10} \rightarrow 01111111 = 127_{10}$$

- **Ca1 :**  $2^n - 1 = 2^8 - 1 = 255$  valores distintos (-127  $\rightarrow$  -0 y 0  $\rightarrow$  127)

$$10000000_{Ca1} = -127_{10} \rightarrow 11111111_{Ca1} = -0_{10}$$

$$00000000_{Ca1} = 0_{10} \rightarrow 01111111_{Ca1} = 127_{10}$$

- **Ca2 :**  $2^n = 2^8 = 256$  valores distintos (-128  $\rightarrow$  0  $\rightarrow$  127)

$$10000000_{Ca2} = -128_{10} \rightarrow 00000000_{Ca2} = 0_{10} \rightarrow 01111111_{Ca2} = 127_{10}$$

# Ventajas y desventajas de los distintos sistemas de representación.

- Módulo y Signo y Ca1 tienen dos representaciones del cero.
  - Esto ocasiona que tengo dos condiciones distintas para verificar un mismo número → aumenta la complejidad
- Las operaciones de suma y resta en módulo y signo son complicadas, en particular no se verifica que  $x-z = x+(-z)$
- Ca2 tiene una única representación del cero, esto me permite representar un valor más.
- En Ca1 y Ca2 se cumple que  $x-z = x+(-z)$ . Esto permite simplificar las unidades aritméticas



## SUMA

- Realizar la suma binaria normal de magnitudes.
  - Los bits de signo son sumados con los bits de magnitud.
- Si de la adición resulta un acarreo más allá del MSB, se descarta
  - Si el MSB es cero, entonces el resultado es positivo.
  - Si el MSB es uno el resultado es negativo y se encuentra representado en  $Ca_2$ .
- **Overflow o desborde:**
  - Puede ocurrir sólo cuando se suman dos números positivos o negativos, y el resultado excede el rango de representación.
  - Cuando ocurre overflow, el signo del resultado es opuesto al signo de los operandos.

# Suma en Ca2 de 5 bits. Ejemplos

Dos números positivos.  
(+9) + (+4)

```

  0 1001 (+9)
+ 0 0100 (+4)
-----
  0 1101 (+13)
    
```

Número positivo mayor en  
módulo que número negativo.  
(+9) + (−4)

```

    0 1001 (+9)
+   1 1100 (−4)
-----
  1|0 0101 (+5)
    
```

Dos números iguales y  
de signo contrario.  
(+9) + (−9)

```

    0 1001 (+9)
+   1 0111 (−9)
-----
  1|0 0000 (0)
    
```

Dos números  
negativos  
(−9) + (−4)

```

  1 0111 (−9)
+ 1 1100 (−4)
-----
 1|1 0011 (−13)
    
```

Número positivo menor en  
módulo que número negativo.  
(+4) + (−9)

```

    0 0100 (+4)
+   1 0111 (−9)
-----
  1 1011 (−5)
    
```

OVERFLOW  
(+9) + (+9)

```

    0 1001 (+9)
+   0 1001 (+9)
-----
  1 0010 (−14)
    
```

# Operaciones en complemento a 2

## RESTA.

- Se utiliza la propiedad  $x - z = x + (-z)$
- Si se utiliza representación Ca2 de n bits, entonces:  
 $-z_{Ca2} = Ca2(z_{Ca2})$  dónde  $Ca2(z)$  es la operación Ca2.

por lo tanto  $x_{Ca2} - z_{Ca2} = x_{Ca2} + Ca2(z_{Ca2})$

o bien  $x_{Ca2} - z_{Ca2} = x_{Ca2} + \neg z_{Ca2} + 1$

( $\neg z_{Ca2}$  es la operación complemento a 1 o negación)

- En los sistemas digitales modernos se representan los números con signo en Ca2 y se implementa la resta a través de sumadores y negadores.

# Operaciones en complemento a 2

## MULTIPLICACIÓN

- Se realiza de manera similar a la multiplicación en base 10.
- En el caso de Números con signo representados en complemento a 2:
  - Si los dos números son positivos, se realiza la operación y se añade un bit de signo 0 (el resultado es positivo)
  - Si los dos números son negativos, se realiza la operación Ca2 sobre ambos números, se multiplican y al resultado se le añade un bit de signo 0 ( el resultado es positivo).
  - Si uno de los números es negativo se hace la operación de Ca2 en ese número, se multiplican y al resultado se le vuelve a hacer la operación de Ca2 (el resultado es negativo). Si es necesario se extiende el signo.
  - Si los operandos están representados en  $n$  bits, el producto podrá llegar a necesitar  $2n$  bits para su representación.

# Operaciones en complemento a 2

MULTIPLICACIÓN ( multiplicando y multiplicador, números positivos representados en 4 bits. Producto representado en 8 bits)

Ejemplo; 9 x 10

$$\begin{array}{r} 1001 \text{ (9)} \\ \times 1010 \text{ (10)} \\ \hline 0000 \\ 1001 \\ 0000 \\ 1001 \\ \hline 01011010 \text{ (90)} \end{array}$$

Ejemplo; 1 x 1

$$\begin{array}{r} 0001 \text{ (1)} \\ \times 0001 \text{ (1)} \\ \hline 0001 \\ 0000 \\ 0000 \\ 0000 \\ \hline 00000001 \text{ (1)} \end{array}$$

Ejemplo; 15 x 15

$$\begin{array}{r} 1111 \text{ (15)} \\ \times 1111 \text{ (15)} \\ \hline 1111 \\ 1111 \\ 1111 \\ 1111 \\ \hline 11100001 \text{ (225)} \end{array}$$

# Multiplicación en Ca2 de 4 bits

Ejemplo: un factor positivo y el otro negativo:  $-3 \times 6$

$$-3 = 1101_{\text{Ca2}} \rightarrow +3 = 0011_{\text{Ca2}}$$

$$\begin{array}{r} 0011 \text{ (+3)} \\ \times 0110 \text{ (+6)} \\ \hline 0000 \\ 0011 \\ 0011 \\ 0000 \\ \hline 00010010 \text{ (+18)} \end{array}$$

Aplico:

$$\begin{aligned} &\text{Ca2}(00010010) = \\ &= 11101110_{\text{Ca2}} = -18 \end{aligned}$$

Ejemplo: dos factores negativos  
 $(-1) \times (-2)$

$$\begin{aligned} -1 &= 1111_{\text{Ca2}} \rightarrow +1 = 0001_{\text{Ca2}} \\ -2 &= 1110_{\text{Ca2}} \rightarrow +2 = 0010_{\text{Ca2}} \end{aligned}$$

$$\begin{array}{r} 0001 \text{ (+1)} \\ \times 0010 \text{ (+2)} \\ \hline 0000 \\ 0001 \\ 0000 \\ 0000 \\ \hline 00000010 \text{ (+2)} \end{array}$$

# DIVISIÓN o COCIENTE

- También puede realizarse de la forma que se realiza para el sistema decimal
- Por ejemplo hagamos  $25 / 4$  y utilicemos 6 bits para representar los operandos.

- $25_{10} = 011001$     $4_{10} = 000100$

- Dividendo  $\rightarrow 11001$     $/100$   $\rightarrow$  Divisor

$$\begin{array}{r} -100 \\ \hline \end{array} \quad 110 \rightarrow \text{Cociente}$$

0100

$$\begin{array}{r} -100 \\ \hline \end{array}$$

00001

$$\begin{array}{r} -000 \\ \hline \end{array}$$

00001  $\rightarrow$  Resto

Se obtiene un cociente 6  
y un resto 1

### DIVISIÓN o COCIENTE

- Para las operaciones de división también valen las consideraciones que se tomaron en cuenta para la multiplicación respecto a los signos de los operandos y resultados
- Si dividendo y divisor son del mismo signo, entonces el cociente resultará positivo
- Si son de distinto signo, el cociente resultará negativo.
- Si el dividendo tiene  $m$  bits y el divisor tiene  $n$  bits, entonces el cociente tendrá  $(m-n)$  bits y el resto tendrá a lo sumo  $n$  bits.



- Se vieron tres formas de representar números binarios con signo:
  - Signo y Módulo (SM)
  - Complemento a 1 (Ca1)
  - Complemento a 2 (Ca2)

Se enumeraron ventajas y desventajas de las distintas representaciones.

- Se vieron como se efectúan las operaciones básicas en el sistema más utilizado que es Ca2.