ISPC
INSTITUTO SUPERIOR
POLITÉCNICO CÓRDOBA

Electrónica Microcontrolada

Circuito Digital
Numeración

Índice

- **Codificación.**
- **Sistemas Decimal, Binario y Hexadecimal:**
  - Codificación en cada sistema.
  - Cambios de base.
  - Operaciones en Binario (suma, multiplicación).
  - Desbordamiento (Overflow).
- **Números Enteros:**
  - Sistemas de representación en binario.
  - Complemento a 2.
  - Overflow en Ca2.

Codificación

- ***Codificación:*** *Conversión de la información a un sistema de representación distinto.*

- **Codificación de información en Binario:**
  - Un elemento concreto, de un conjunto de M elementos, se codifica como un vector (tira, secuencia) de n bits, con $n \geq \log_2 M$:

$$\mathbf{X = (X_{n-1}, X_{n-2}, \ldots, X_2, X_1, X_0)}$$

- ¿Cómo se realiza la asignación elemento ↔vector? Depende:
  - Caracteres Alfanuméricos: código ASCII de 8 bits (1 byte).
  - Números naturales: sistema convencional en base 2 (binario).
  - Números enteros: Complemento a 2.
  - Números reales: ANSI/IEEE Floating Point Standard.

Codificación

- **Ejemplo: Tabla ASCII:**
  - **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
    - 1963 (Telegrafía)
    - Ordenado Alfabéticamente (pero empieza en 00110000??)
    - 32 primeros códigos, caracteres de control (para una impresora, por ejemplo)

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

Índice

- Codificación.
- **Sistemas Decimal, Binario y Hexadecimal:**
  - Codificación en cada sistema.
  - Cambios de base.
  - Operaciones en Binario (suma, multiplicación).
  - Desbordamiento (Overflow).
- Números Enteros:
  - Sistemas de representación en binario.
  - Complemento a 2.
  - Overflow en Ca2.

Decimal, Binario y Hexadecimal

Probablemente…

- **¿Por qué decimal?**
- **Sistema convencional en Base 10:**
  - Codificación de un subconjunto de números naturales en base 10.
  - Números Naturales= {0, 1, 2, 3, …}.
  - Sistema de numeración (reglas de representación):
    - Sea el vector de digitos $X = (X_{n-1}, X_{n-2}, ..., X_2, X_1, X_0)$ con $X_i \in \{0, 1, 2, ... 8, 9\}$.
    - El valor que representa el vector X interpretándolo como un número codificado en el sistema convencional en base 10 es:

      $$X_{ud} = X_{n-1}10^{n-1} + X_{n-2}\,10^{n-2} + \dots + X_2\,10^2 + X_1\,10^{1} + X_0\,10^0 = \sum_{(i=0,\dots,n)} X_i\,10^i$$

  - Rango de representación (para n dígitos):

    $$0 \le X_{ud} \le 10^n - 1$$

Decimal, Binario y Hexadecimal

- **Sistema convencional en Base 2:**
  - Codificación de un subconjunto de números naturales en base 2.
  - Números Naturales= {0, 1, 2, 3, …}.
  - Sistema de numeración (reglas de representación):
    - Sea el vector de digitos $X = (X_{n-1}, X_{n-2}, ..., X_2, X_1, X_0)$ con $X_i \in \{0, 1\}$.
    - El valor que representa el vector X interpretándolo como un número codificado en el sistema convencional en base 2 es:

      $$X_u = X_{n-1}2^{n-1} + X_{n-2}2^{n-2} + \ldots + X_2 2^2 + X_1 2^{1+} X_0 2^0 = \sum_{(i=0,\ldots,n)} X_i 2^i$$

  - Rango de representación (para n dígitos):

    $$0 \leq X_u \leq 2^n - 1$$

  - Ejemplo: X= 1011, Valor??

    $$X_u = 1 \cdot 2^3 + 0 \cdot 2^2 \; X + 1 \cdot 2^1 + 1 \cdot 2^0 = 8+0+2+1 = 11$$

Decimal, Binario y Hexadecimal

- Cambio de base: **Binario → Decimal**:
  - Dado $X = (X_{n-1}, X_{n-2}, \ldots, X_2, X_1, X_0)$ con $X_i \in \{0, 1\}$, encontrar X_u.

$$X_u = X_{n-1}2^{n-1} + X_{n-2}2^{n-2} + \ldots + X_2 2^2 + X_1 2^1 + X_0 2^0 = \sum_{(i=0,\ldots,n)} X_i 2^i$$

- Cambio de base: **Decimal → Binario**:
  - Dado X_u encontrar $X = (X_{n-1}, X_{n-2}, \ldots, X_2, X_1, X_0)$ con $X_i \in \{0, 1\}$.

$$X_u = X_{n-1}2^{n-1} + X_{n-2}2^{n-2} + \ldots + X_2 2^2 + X_1 2 + X_0$$

Dividendo *divisor*

$$X_u / 2 = (X_{n-1}2^{n-1} + X_{n-2}2^{n-2} + \ldots + X_2 2^2 + X_1 2) / 2 + X_0 / 2 =$$

cociente *resto*

$$(X_{n-1}2^{n-2} + X_{n-2}2^{n-3} + \ldots + X_2 2 + X_1) + X_0 / 2$$

*El Bit de menor peso es el resto de dividir por 2 el número que se desea representar en binario.
*Repetir el proceso con el cociente para encontrar el resto de bits.

Remember…

D | d

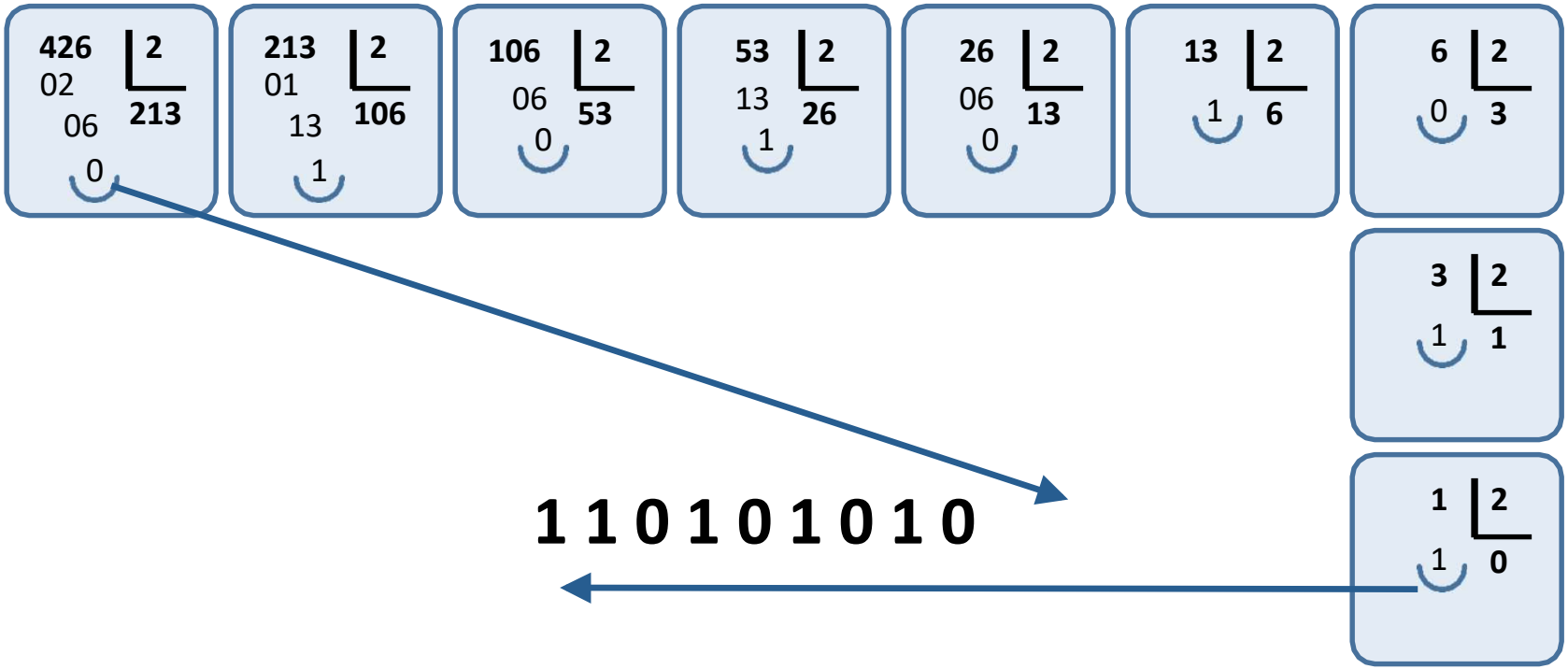
r c

$c * d + r = D$

$D/d = c + r/d$

Decimal, Binario y Hexadecimal

- **Ejemplo <u>Decimal → Binario</u>:**
  - Dado $X_u = 426$ encontrar $X = (x_{n-1}, x_{n-2}, ..., x_2, x_1, x_0)$ que lo representa en binario (con $x_i \in \{0, 1\}$.



Decimal, Binario y Hexadecimal

- **Sistema convencional en Base 16 (Hexadecimal):**
  - Codificación de un subconjunto de números naturales en base 16.
  - Sistema de numeración (reglas de representación):
    - Sea el vector de digitos $X = (X_{n-1}, X_{n-2}, ..., X_2, X_1, X_0)$ con $X_i \in \{0, 1,2,.. 9,A,B,C,D,E,F \}$.
    - El valor que representa el vector X interpretándolo como un número codificado en el sistema convencional en hexadecimal es:

      $$X_u = X_{n-1}16^{n-1} + X_{n-2}\,16^{n-2} + \ldots + X_2\,16^2 + X_1\,16^{1} + X_0\,2^0 \quad = \sum_{(i=0,\ldots,n)} X_i\,16^i$$

  - Rango de representación (para n dígitos):

    $$0 \leq X_u \leq 16^n - 1$$

- **¿Por qué Hexadecimal?:**
  - La palabra (word) de los procesadores actuales es de 32 ó 64 bits.
  - Es engorroso escribir vectores tan largos en binario. Utilizaremos notación hexadecimal, que es más compacta.

Decimal, Binario y Hexadecimal

- Cambio de base: **Binario → Hexadecimal:**
  - Dado X = $(X_{n-1}, X_{n-2}, \ldots, X_2, X_1, X_0)$ con $X_i \in \{0, 1\}$, encontrar X_u.

$$X_u = X_{n-1}2^{n-1} + \cdots + X_1 2^1 + X_0 2^0 =$$

$$\underbrace{(x_{n-1}2^3 + x_{n-2}2^2 + x_{n-3}2^1 + x_{n-4})}_{h_k} 16^k + \ldots + \underbrace{(x_7 2^3 + x_6 2^2 + x_5 2^1 + x_4)}_{h_1} 16 + \underbrace{(x_3 2^3 + x_2 2^2 + x_1 2^1 + x_0)}_{h_0}$$

Ejemplo: X= 1 0 0 1 0 0 1 1 1 0 1 1 1 0 1 0

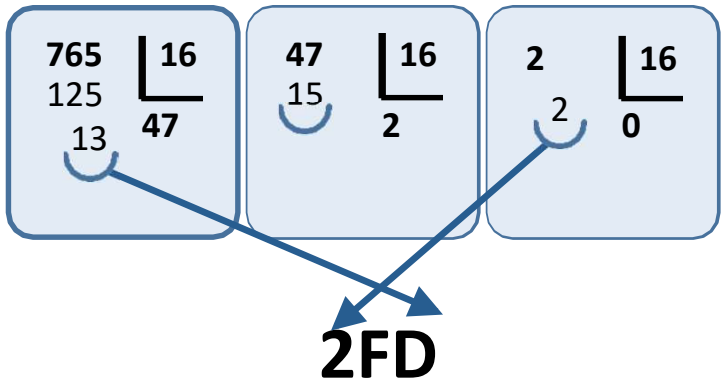
9 3 B A

X_u= 0x93BA

- Cambio de base: **Hexadecimal → Binario:**
  - Convertir cada dígito hexadecimal en su equivalente binario.

Decimal, Binario y Hexadecimal

- Cambio de base **<u>Decimal → Hexadecimal</u>:**
  - Mismo proceso de división que en el caso de decimal-->binario. El divisor cambia de 2 a 16.
  - Dado X_u = 426 encontrar X = (x_{n-1}, x_{n-2}, ..., x_2, x_1, x_0) que lo representa en binario (con x_i ϵ {0, 1}).



Decimal, Binario y Hexadecimal

- **Ejercicios:**
  - Obtener el valor del número natural Zu representado en binario por los siguientes vectores de bits Z:

    Z=11000100 Z=00101111

  - Obtener el vector X de 8 bits que representa en binario cada uno de los siguientes números naturales. Expresar X también en hexadecimal. Indicar en qué casos el número no se puede representar con 8 bits:

    X_u=35 X_u=79 X_u=145 X_u=284

  - Obtener el valor de los siguientes vectores de 16 bits (alguno representado en hexadecimal):

    X_u=65342 X_u=23 X_u=98767

Decimal, Binario y Hexadecimal

- **Operaciones en base b: SUMA**
  - Dados 2 vectores de n dígitos, $X = x_{n-1}x_{n-2}\cdots x_1x_0$, $Y = y_{n-1}y_{n-2}\cdots y_1y_0$ con $x_i, y_i \in \{0, 1, \ldots, b-1\}$ que representan dos números naturales X_u e Y_u en un sistema convencional en base b, encontrar el vector $W = w_nw_{n-1}\cdots w_1w_0$ con $w_i \in \{0,1,\ldots,b-1\}$ que representa en el sistema convencional en base b al número natural $W_u = X_u + Y_u$

  - Expresado de otra forma; encontrar los dígitos $w_nw_{n-1}\cdots w_1w_0$ tales que cumplan lo siguiente:

$$\sum_{i=0}^{n} w_i \times b^i = \sum_{i=0}^{n-1} x_i \times b^i + \sum_{i=0}^{n-1} y_i \times b^i \qquad \text{EQ1}$$

$$w_i \in \{0,1,\ldots,b-1\} \ \forall i \qquad \text{EQ2}$$

Decimal, Binario y Hexadecimal

- Un primer intento de solución:
  - Manipulando la EQ1 encontramos la siguiente expresión equivalente:

$$w_n \times b^n + \sum_{i=0}^{n-1} w_i \times b^i = \sum_{i=0}^{n-1} (x_i + y_i) \times b^i$$

  - Una solución trivial (de las infinitas que hay):

$$W_i = x_i + y_i \text{ para } 0 \le i \le n-1 \text{ y } w_n = 0$$

- Ejemplo: para b = 10 y n = 4, sumar X = 6493 e Y = 8199:

		Dígito 4	Dígito 3	Dígito 2	Dígito 1	Dígito 0
	X		6	4	9	3
	Y		8	1	9	9
W		0	14	5	18	12

Atención, W no cumple EQ2, ya que algunos valores no están representados con un solo dígito de la base.

Decimal, Binario y Hexadecimal

- ¿Cómo solucionamos el problema con EQ2? Restando b en w_k y sumando 1 (acarreo ó **carry**) a w_{k+1}
  - Se sigue cumpliendo EQ1, ya que:

$$w_{k+1} \times b^{k+1} + w_k \times b^k = w_{k+1} \times b^{k+1} + (w_k + b - b) \times b^k = w_{k+1} \times b^{k+1} + b^{k+1} + (\hat{w}_k - b) \times b^k =$$
$$(w_{k+1} + 1) \times b^{k+1} + (w_k - b) \times b^k$$

  - Cumplimos con EQ2. Ejemplo anterior, **dígito 1:**

		D 4	D 3	D 2	D 1	D 0
	X		6	4	9	3
	Y		8	1	9	9
W		0	14	**5**	**18**	12

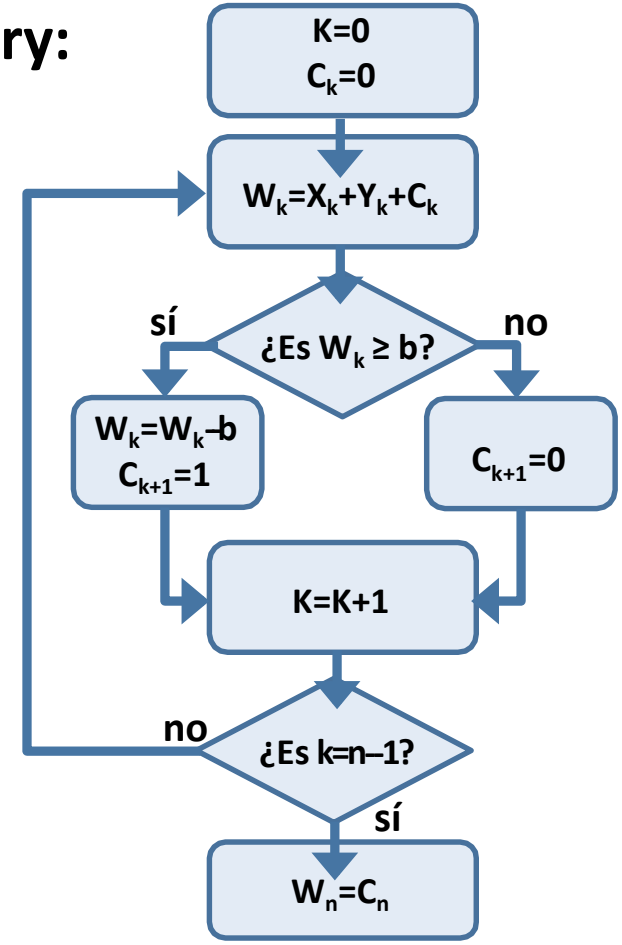
		D 4	D 3	D 2	D 1	D 0
	X		6	4	9	3
	Y		8	1	9	9
W		0	14	**6**	**8**	12

*** Es necesario repetir el proceso para cada dígito que no cumpla EQ2.**

Decimal, Binario y Hexadecimal

- **Algoritmo de suma con propagación de carry:**

		D 4	D 3	D 2	D 1	D 0
			6	4	9	3
			8	1	9	9
k =0	w_0					
	2					
	c_1				1	
k =1	w_1				9	
	c_2			1		
k = 2	w_2			6		
	c_3		0			
k = 3	w_3		4			
	c_4	1				
	W	**1**	**4**	**6**	**9**	**2**



Decimal, Binario y Hexadecimal

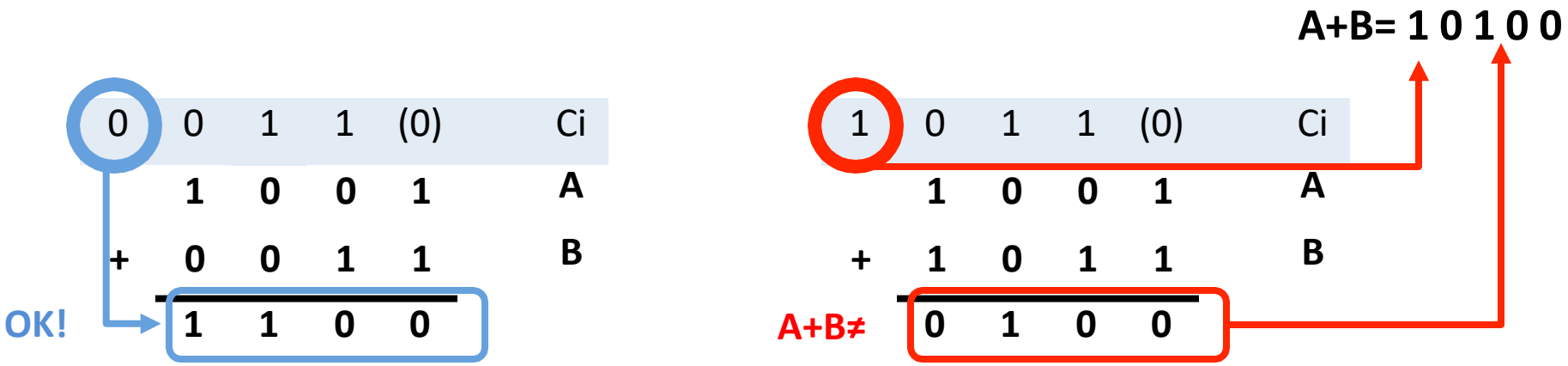
- **SUMA de números BINARIOS:**

			512	256	128	64	32	16	8	4	2	1
x	(987)		1	1	1	1	0	1	1	0	1	1
y	(123)					1	1	1	1	0	1	1
Carries		1	1	1	1	1	1	1	0	1	1	
x+y	(1110)	1	0	0	0	1	0	1	0	1	1	0
		s_{10}	s_9	s_8	s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0

x_i	y_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Decimal, Binario y Hexadecimal

- **El problema del Desbordamiento (overflow):**
  - Los sistemas digitales operan normalmente sobre un número fijo de dígitos. Suele ser el mismo valor para operandos y resultado.
  - Con n bits el rango representable es **$[0,2^n-1]$**.
  - Si **$A+B>2^n-1$** el resultado no es representable, hay **overflow**.
  - El bit de carry señala la existencia de desbordamiento.



Decimal, Binario y Hexadecimal

- **Multiplicación y división por potencias de 2:**
  - Multiplicación: desplazamiento hacia la izquierda:

$$X_u = X_{n-1}2^{n-1} + X_{n-2}2^{n-2} + \ldots + X_2 2^2 + X_1 2^1 + X_0 2^0$$

$$X_u * 2 = X_{n-1}2^n + X_{n-2}2^{n-1} + \ldots + X_2 2^3 + X_1 2^2 + X_0 2^1$$

Ejemplo: 1010*2 = 10100.

  - División: desplazamiento hacia la derecha:

$$X_u = X_{n-1}2^{n-1} + X_{n-2}2^{n-2} + \ldots + X_2 2^2 + X_1 2^1 + X_0 2^0$$

$$X_u / 2 = X_{n-1}2^{n-2} + X_{n-2}2^{n-3} + \ldots + X_2 2^1 + X_1 2^0$$

Ejemplo: 1010/2 = 101.

Índice

- **Codificación.**
- **Sistemas Decimal, Binario y Hexadecimal:**
  - Codificación en cada sistema.
  - Cambios de base.
  - Operaciones en Binario (suma, multiplicación).
  - Desbordamiento (Overflow).
- **Números Enteros:**
  - Sistemas de representación en binario.
  - Complemento a 2.
  - Overflow en Ca2.

Números Enteros

- **Buscando una representación:**
  - Un entero { …, –2,–1,0, 1, 2, … } se representa internamente en el computador, como cualquier otra información, mediante un vector de n bits:

    $$X = x_{n-1}x_{n-2}\ldots x_2x_1x_0 \text{ con } x_i \in \{0,1\}$$

  - Definir una representación consiste en encontrar una tabla o una expresión aritmética que, para cada posible vector de bits, nos indique el número que representa.
  - No puede ser la misma para enteros que para naturales (la que conocemos), porque ésta es solo para positivos.
  - Hay muchas formas de representación. Se busca una con la que sea sencillo y rápido hacer operaciones con los números.

Números Enteros

- **Una posible solución: <u>Signo--Magnitud</u>:**
  - Dado un vector X de n bits que representa al número entero X_{sm} en signo--magnitud, el bit de más a la izquierda del vector de bits codifica el signo (si $x_{n-1}= 0 \rightarrow$positivo; si $x_{n-1}= 1 \rightarrow$negativo).
  - Los n-1 bits restantes representan en binario el valor absoluto de X_{sm} (su magnitud), que es un número natural.
  - Inconveniente: ¡¡Dos representaciones para el cero!!

$$X_{sm}= \begin{cases} \sum_{i=0}^{n-2} x_i 2^i & si \quad x_{n-1} = 0 \\ -\left(\sum_{i=0}^{n-2} x_i 2^i \right) & si \quad x_{n-1} = 0 \end{cases}$$

X			
x_2	x_1	x_0	X_{sm}
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	-0
1	0	1	-1
1	1	0	-2
1	1	1	-3

$$-(2^{n-1}-1) \leq X_{sm} \leq 2^{n-1}-1$$

Números Enteros

- **Suma en Signo--Magnitud:**
  - Algoritmo habitual:
    - Operandos con el mismo signo: la magnitud del resultado es la suma de las magnitudes de los operandos y el signo del resultado es el signo de los operandos.
    - Operandos de distinto signo: la magnitud del resultado se obtiene restando a la magnitud mayor la menor. El signo del resultado es el signo del operando de mayor magnitud.
  - Este algoritmo requiere: un sumador de números naturales codificados en binario con n–1bits (operandos del mismo signo), un comparador y un restador de números naturales codificados en binario con n–1bits (números con distinto signo).
  - Conclusión: la suma de números enteros en signo--magnitud es mucho más costosa en hardware y/o en tiempo de propagación que la suma de números naturales en binario.
  - Los computadores actuales no usan esta representación para números enteros.

Números Enteros

- **Buscando una representación más efectiva:**
  - Se busca una representación tal que la suma de números enteros se pueda realizar de la mism forma (con el mismo sumador) que se usa para los números naturales codificados en binario.
  - La representación en signo--magnitud no sirve.

- **Una posible solución:**
  - Codificamos los números positivos como en binario (y como en signo–magnitud). Así la suma de positivos será correcta con el sumador binario (siendo el resultado representable.
  - Recordando cómo opera el sumador binario, el valor –1tiene que ser codificado como 111 (para n=3). Este es el único vector que al sumarle el vector 001 (que representa al 1) obtiene el vector 000 (el cero), haciendo que –1+1 sea correcto.

Números Enteros

- **Representación en Complemento a 2:**
  - Tomamos como valor de partida nuestro –1 (111 para n=3).
  - Si seguimos sumando –1, iremos encontrando el resto de valores negativos. Así, –2 = –1 + –1, se representa como:

$$\begin{array}{cccc} & 1 & 1 & 1 \\ + & \underline{1} & & \underline{1} \quad \underline{1} \\ & 1 & 1 & 0 \end{array}$$

  - Resumiendo:

X			
x_2	x_1	x_0	X_s	
0	0	0	0	Cero
0	0	1	1	Positivos
0	1	0	2	
0	1	1	…	
1	0	0		
1	0	1	…	
1	1	0	–2	
1	1	1	–1	Negativos

¿Dónde pongo la frontera entre positivos y negativos?

Números Enteros

- **Posibles representaciones:**

x_2	x_1	x_0	X_s
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	-3
1	1	0	-2
1	1	1	-1

$-3 \leq Xs \leq 4$

x_2	x_1	x_0	X_s
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	-5
1	0	0	-4
1	0	1	-3
1	1	0	-2
1	1	1	-1

$-5 \leq Xs \leq 2$

Complemento a 2

x_2	x_1	x_0	X_s
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	-4
1	0	1	-3
1	1	0	-2
1	1	1	-1

$-4 \leq Xs \leq 3$

- Rango más simétrico (sin ser completamente simétrico).
- El dígito más a la izquierda indica el signo (0→positivo, 1 → negativo).
- La detección de resultado no representable (overflow) es más sencilla que en otros casos (lo veremos a continuación).

Números Enteros

- **Ca2, generalizando para n bits (Ca2-->decimal):**
  - Número positivo:

$$X_s = \sum_{i=0}^{n-2} x_i 2^i \ para \ x_{n-1} = 0$$

  - Número negativo: al dígito de más a la izquierda le damos el mismo peso que le corresponde en binario (2^{n-1}) pero con signo negativo y al resto de dígitos el peso y el signo positivo correspondiente a binario.

$$X_s = -2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \ para \ x_{n-1} = 1$$

  - **Positivos y negativos:**

$$X_s = -x_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

Rango de representación:

$$-2^{n-1} \leq X_s \leq 2^{n-1}-1$$

Números Enteros

- **Cambio de Signo (usar para decimal -> Ca2):**
  - Algoritmo para la operación aritmética de cambio de signo de un entero representado en Ca2.
  - Dados los n bits de un vector X, obtener el vector W tal que $W_s = -X_s$:
    - Paso 1: se complementan los bits (Ca1).
    - Paso 2: se suma 1 al resultado (no se tiene en cuenta el acarreo).
  - Ejemplos:

    010 → 101 → 101+1=110

    000 → 111 → 111+1=000

	X		
x_2	x_1	x_0	X_s
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	-4
1	0	1	-3
1	1	0	-2
1	1	1	-1

Números Enteros

- **Suma en Ca2, con detección de overflow:**
  - El objetivo de la representación en Ca2 es que la suma de dos números enteros se pueda efectuar de la misma forma (con el mismo sumador) que para los naturales representados en binario.
  - ***La única diferencia es la detección del resultado no representable*.**
  - Un ejemplo:

Binario

	1	**1**	**1**	(+7)	
+	**0**	**0**	**1**	(+1)	
1	**0**	**0**	**0**	(0)	Incorrecto

Ca2

	1	**1**	**1**	(–1)	
+	**0**	**0**	**1**	(+1)	
1	**0**	**0**	**0**	(0)	Correcto

  - En Ca2 la detección de resultado no representable se efectúa con los bits de signo de los operandos (x_{n-1} e y_{n-1}) y del resultado (w_{n-1}):
    - **Si $SIG(x_{n-1}) \neq SIG(y_{n-1})$: Resultado siempre representable.**
    - **Si $SIG(x_{n-1}) = SIG(y_{n-1})$ y $SIG(w_{n-1}) = SIG(y_{n-1})$: representable.**
    - **Si $SIG(x_{n-1}) = SIG(y_{n-1})$ y $SIG(w_{n-1}) \neq SIG(y_{n-1})$: overflow.**

Números Enteros

- DETECCIÓN DE RESULTADO NO REPRESENTABLE (OVERFLOW):

Binario

	1	1	1	(+7)
+	0	0	1	(+1)
1	0	0	0	(0)

C_n (carry de la última columna) dicta si el resultado es correcto o no:

C_n=0: Correcto

C_n=1: Overflow

Ca2

	1	1	1	(-1)
+	0	0	1	(+1)
1	0	0	0	(0)

x_{n-1} , y_{n-1} , w_{n-1} , dictan si el resultado es correcto o no:

$x_{n-1} \neq y_{n-1}$: Correcto

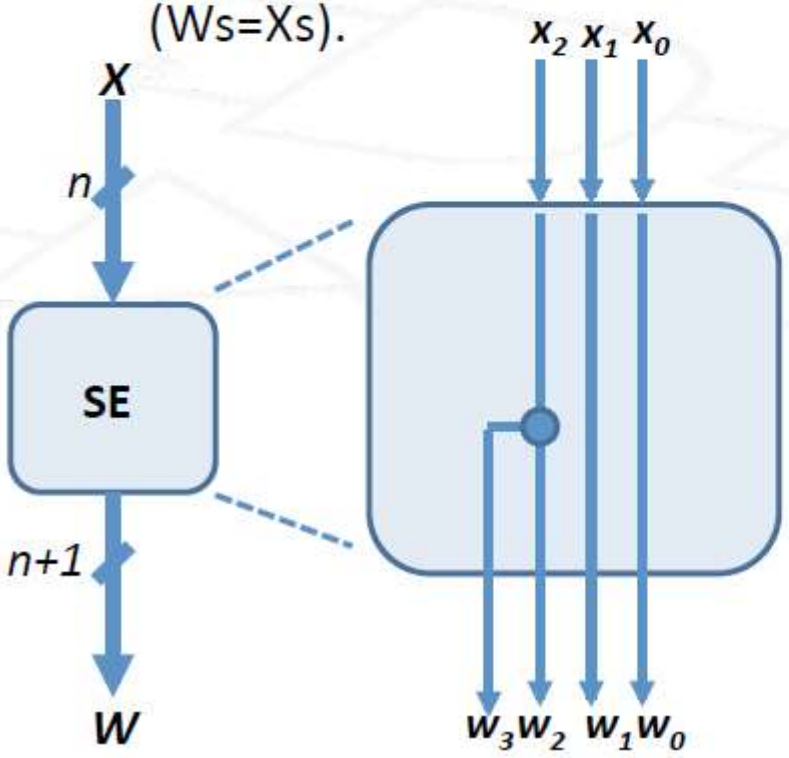
$x_{n-1} = y_{n-1} = w_{n-1}$: Correcto

$x_{n-1} = y_{n-1} \neq w_{n-1}$: Overflow

$$V_n = \overline{x}_{n-1} \cdot \overline{y}_{n-1} \cdot w_{n-1} + x_{n-1} \cdot y_{n-1} \cdot \overline{w}_{n-1}$$

Números Enteros

- **Extensión de Rango:**
  - Dado el vector X de n bits que representa en Ca2 al número entero Xs, ¿Cómo se obtiene un vector W de n+1 bits que represente a ese mismo número? (Ws=Xs).



X			
x_2	x_1	x_0	X_s
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	-4
1	0	1	-3
1	1	0	-2
1	1	1	-1

W				
w_3	w_2	w_1	w_0	W_s
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	-8
1	0	0	1	-7
1	0	1	0	-6
1	0	1	1	-5
1	1	0	0	-4
1	1	0	1	-3
1	1	1	0	-2
1	1	1	1	-1

Números Enteros

- **Ejercicios:**
  - Expresar en Ca2 los siguientes valores (en decimal):

    –5, 176, –176, 204, –204.

  - Resultado en Ca2

    de: Xs= 176+176.

    Xs= –176–204.

    Xs= 176–204.

    ¿Hay desbordamiento? ¿Por qué?