2.- Representación de la información en formato digital (parte 1)

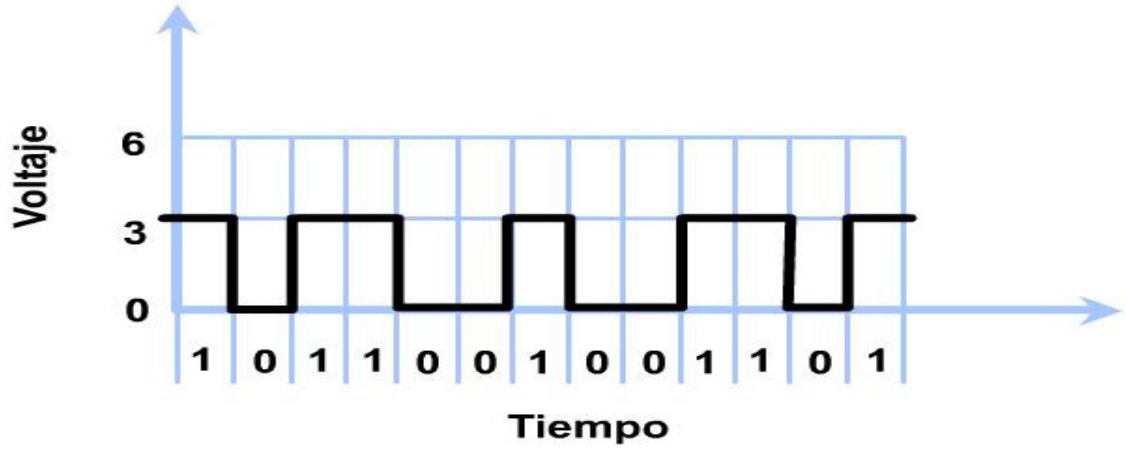
Diseño Lógico 2018

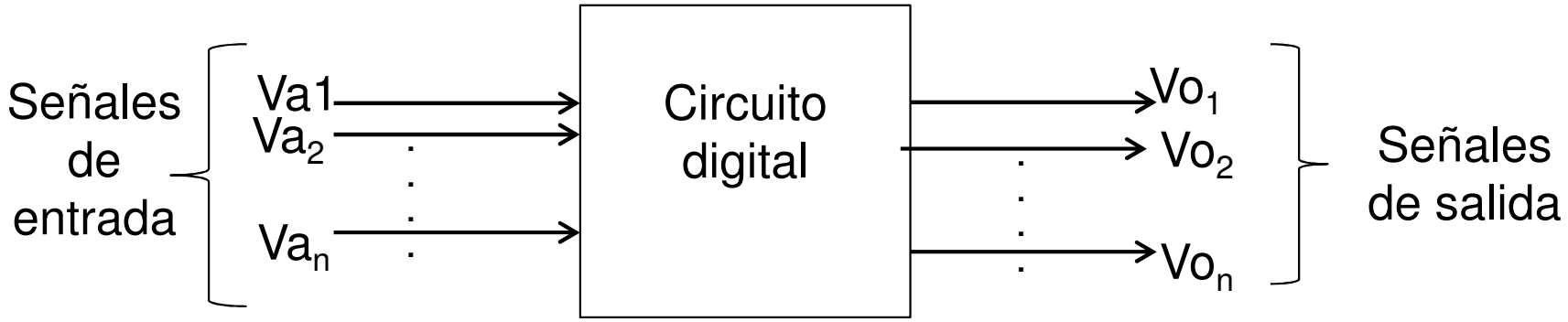
Representación de la información

- Sistema digital binario:
  - Representación de la información por cantidades físicas que solo pueden asumir 2 valores (0 o 1).
- Escalares:
  - Variable binaria ***x*** (minúscula) que puede tomar un de 2 valores (0 o 1)
  - Esta variable representa **1 bit** de información
  - **bit**: del inglés *"binary digit"*, dígito binario.
- Vector (n-upla)
  - Variable formada por un conjunto n de escalares binarios que se representa $\boldsymbol{X = [x_1, \ldots\ldots, xn]}$, (X mayúscula).
  - Si cada variable binaria puede tomar 2 valores posibles, el vector puede representar: 2 x 2 x 2x …..x 2 = 2^n valores.

Representación de la información

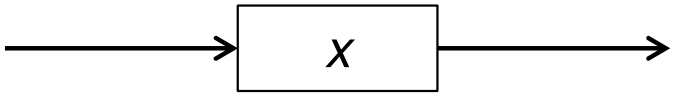
- **Señal**:
  - Información siendo transmitida dentro del sistema



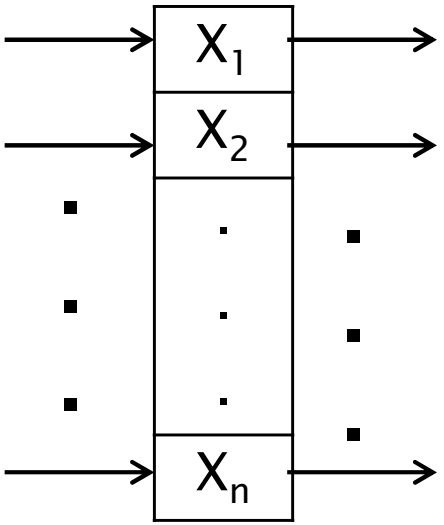


Representación de la información

- **Celda**:
  - Unidad básica de almacenamiento de información.
  - Consideraciones:
    - La salida tiene el valor almacenado en la celda
    - La entrada permite cambiar su valor.
  - Def: Estado de la celda = Valor de la variable ***x***

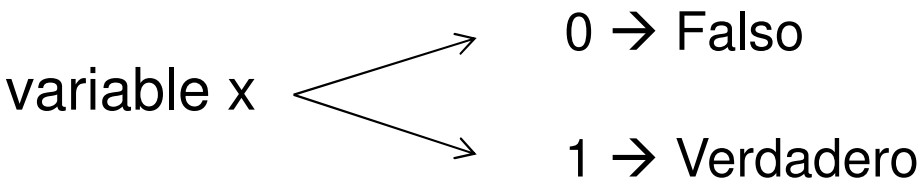


- **Registro**
  - Conjunto ordenado de celdas
  - Un registro de n bits tiene 2^n estados posibles
  - 1 byte = 8 bits ➔ 256 estados posibles
  - 1 nibble = 4 bits ➔ 16 estados posible



Información NO numérica

- **Codificación Lógica**



Ejemplo: Lenguaje programación

Var x:boolean	Define la variable *x,* utiliza 1 byte
x ← False	*x* ← (0,0,0,0,0,0,0,0)
x ← True	*x* ← (1,1,1,1,1,1,1,1)

¿Qué representa?

True: "La puerta está abierta"

False: "La puerta NO esta abierta" ≠ "La puerta está cerrada"

Información NO numérica

- **Codificación simbólica**
  - Estados de un determinado proceso
    - Ej Portón: Cerrado, Cerrando, Abierto, Abriendo, Fuera de servicio.
  - Comandos de un sistema
    - Ej Control remoto: Subir el volumen, siguiente canal, mute, etc

Control remoto TV Philips ➔

Comando	Codificación
Volumen +	010000
Programa +	100001
Mute	001101

  - Caracteres
    - Código ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchangeange)
      - 7 bits: inicialmente (128 caracteres)
      - 8 bits ASCII extendido (256 caracteres)
    - ISO 8859-1: Incluye todos los caracteres de todos los alfabetos latinos.

Lo importante es que todos usemos el mismo código (en la realidad no es así).

Tabla ASCII

Binario	Dec	Hex	Representación
0010 0000	32	20	espacio ( )
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

Binario	Dec	Hex	Representación
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_

Binario	Dec	Hex	Representación
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~

- Ref. https://sites.goog[illegible]com/site/naomihuit303/primer-parcial/codigo-ascii

Información numérica

SISTEMAS DE NUMERACIÓN:

El sistema numérico que utilizamos es el "*posicional base 10*"

Un numero se representa por una cadena de dígitos donde cada posición tiene un "**peso**" asociado (10^i).

Ej: $1543{,}28 = \mathbf{1} \times 10^3 + \mathbf{5} \times 10^2 + \mathbf{4} \times 10^1 + \mathbf{3} \times 10^0 + \mathbf{2} \times 10^{-1} + \mathbf{8} \times 10^{-2}$

El 5 tiene peso 10^2

Potencias negativas indican a la derecha de la coma

Notar que en base 10 hay 10 símbolos.

En general, para base r se necesitan r símbolos:

$$N = a_{n-1} . r^{n-1} + \ldots\ldots + a_1 . r^1 + a_0 . r^0 + a_{-1} . r^{-1} + \ldots\ldots + a_{-m} . r^{-m}$$

con $a_i / 0 \leq a_i \leq r-1$

La representación sería: $N = a_{n-1}\ldots\ldots a_1\ a_0, a_{-1}\ \ldots\ldots a_{-m}$

Información numérica

- **Las bases más utilizadas**

Nombre	Base	Símbolos
Binario	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Notación:

- 100_r indica el número 100 en base r
- Para base 10 no se suele utilizar subíndice
- Otras formas de notación: Hexadecimal: **0xAF** = AFH = AF_{16}
  Binario: 1010b = 1010_2

Información numérica

DECIMAL	BINARIO	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

DECIMAL	BINARIO	HEXADECIMAL
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Información numérica

- **Sistema binario – base 2:** (r=2, símbolos 0 y 1)

  Son de vital importancia pues pueden ser fácilmente procesados por sistemas digitales.

  Ejemplos:

$$10011_2 = \mathbf{1} \times 2^4 + \mathbf{0} \times 2^3 + \mathbf{0} \times 2^2 + \mathbf{1} \times 2^1 + \mathbf{1} \times 2^0$$
$$= 16 + 2 + 1 = 19$$

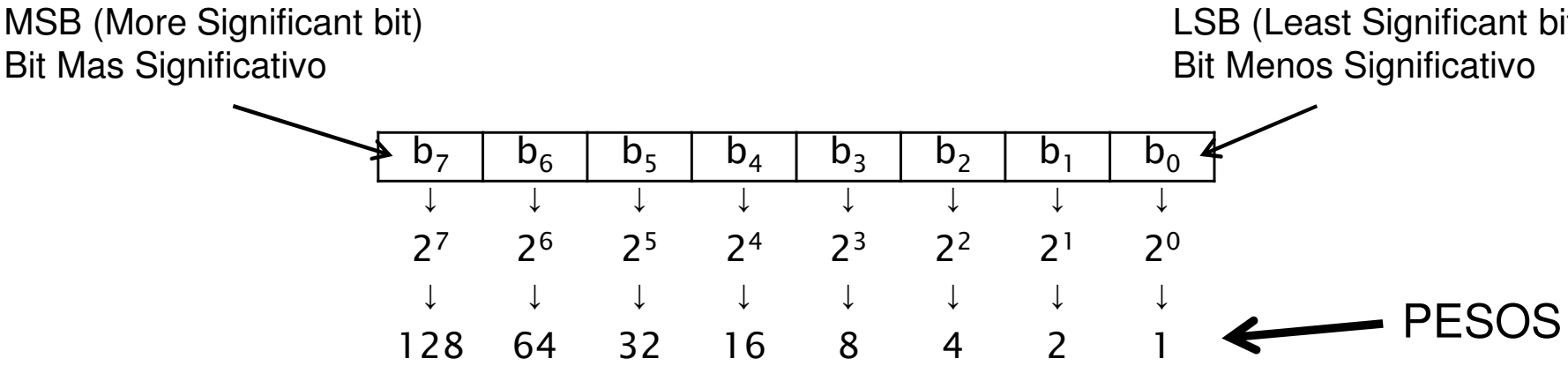
$$1101{,}001_2 = \mathbf{1} \times 2^3 + \mathbf{1} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{1} \times 2^0 + \mathbf{0} \times 2^{-1} + \mathbf{0} \times 2^{-2} + \mathbf{1} \times 2^{-3}$$
$$= 8 + 4 + 1 + 0{,}125$$
$$= 13{,}125$$

  Solo sumo los pesos donde hay un “1”

Información numérica

- **Sistema binario**

Consideramos un registro de 8 bits:



Largo fijo: Los sistemas digitales solo pueden manejar números con una cantidad fija de dígitos. Esto nos obliga a "completar" con 0´s:

Ejemplo para 8 bits: 10011_2 se representa como $0001\ 0011_2$

Información numérica

- **Sistema hexadecimal**: **– base 16:** (r=16, símbolos 0,1, ….,A,B,D,E,F)

  Ejemplo:

  Binario: $1101001001010011011111111001110010001110000 1_2$

  Hexadecimal: 0xD25 37F9 B8E1

Ambos números son iguales

Los números binarios son muy prácticos para los sistemas digitales, pero son muy difíciles de visualizar. Suele ser engorroso trabajar con binarios.

Ni los sistemas digitales ni nosotros trabajamos en hexadecimal, pero es una muy buena herramienta para visualizar y documentar números con múltiples bits.

Información numérica

- **Sistema hexadecimal**

$2^0 = 16^0$; $2^4 = 16^1$; $2^8 = 16^2$; ..... $2^{4n} = 16^n$ (n entero)

Ejemplo:

$10\ 1101_2 = \mathbf{0} \times 2^7 + \mathbf{0} \times 2^6 + \mathbf{1} \times 2^5 + \mathbf{0} \times 2^4 + \mathbf{1} \times 2^3 + \mathbf{1} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{1} \times 2^0$

$= (0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0) \times 2^4 + (1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) \times 2^0$

$= 2 \times 16^1 + D \times 16^0$

$= 0x2D$

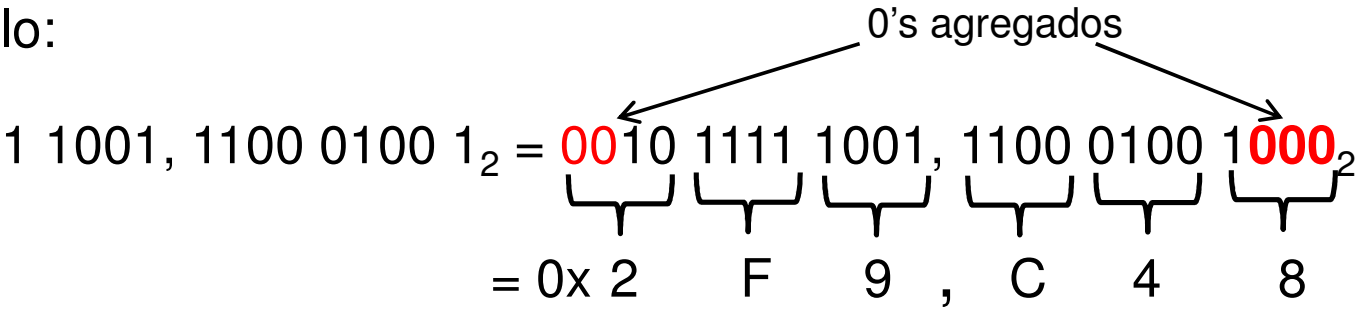
Método: Si la cantidad de dígitos binarios no es múltiplo de 4 se completa con 0's a la izquierda.
Se separa en grupos de 4 bits y cada grupo se sustituye por su equivalente hexadecimal.

Información numérica

- **Sistema hexadecimal**

Este método también sirve si el número binario tiene parte fraccionaria. Basta contar los grupos de 4 bits hacia la izquierda y hacia la derecha de la coma, agregando los 0's necesarios .

Ejemplo:



Nota: Vale lo mismo para números octales, basta hacer grupos de a 3 bits.

Información numérica

- **Cambio de base**

De base 2 a base 10:

Ejemplo: $1\ 1011\ 0110_2 = 256 + 128 + 32 + 16 + 4 + 2 = 438$

Sumo solo los pesos donde hay un 1

De base 16 a base 10:

Ejemplo: $0x1A2,E = 1 \times 16^2 + 10 \times 16^1 + 2 \times 16^0 + 14 \times 16^{-1}$

$= 1 \times 256 + 10 \times 16 + 2 \times 1 + 14 \times 0,0625 = 418,875$

Información numérica

- **Cambio de base**

De base 10 a base 2:

Voy a tener: $N_{10} = (\text{Parte entera})_{10} + (\text{Parte fraccionaria})_{10}$

$N_{10} = \quad I_{10} , F_{10}$

Pues podría tener infinitos dígitos

$N_{10} = I_{10} , F_{10} \rightarrow N_2 = I_2 , F_2 = b_{n-1}\, b_{n-2} \ldots b_1\, b_0 \, , \, b_{-1}\, b_{-2} \ldots b_{-m} \ldots\ldots$

PARTE ENTERA: $I_{10} = I_2 = b_{n-1}\, 2^{n-1} + b_{n-2}\, 2^{n-2} \ldots b_1\, 2^{-1} + b_0\, 2^0$

PARTE FRACCIONARIA: $F_{10} = F_2 = b_{-1} 2^{-1} + b_{-2}\, 2^{-2} + \ldots + b_{-m}\, 2^{-m} + \ldots$

Recordar: Si A / B = Q y resto R → A = B x Q + R

Información numérica

- **Cambio de base**

De base 10 a base 2:

PARTE ENTERA: $I_{10} = I_2 = b_{n-1}\ 2^{n-1} + b_{n-2}\ 2^{n-2} \ldots\ b_1\ 2^1 + b_0$

Si divido I_{10} entre 2 obtengo cociente Q_1 y resto b_0:

$$I_{10} = 2 \cdot Q_1 + b_0$$

$$Q_1 = b_{n-1}\ 2^{n-2} + b_{n-2}\ 2^{n-3} \ldots\ b_1$$

Si divido Q_1 entre 2 obtengo cociente Q_2 y resto b_1:

$$Q_1 = 2 \cdot Q_2 + b_1$$

$$Q_2 = b_{n-1}\ 2^{n-3} + \ldots\ldots + b_2$$

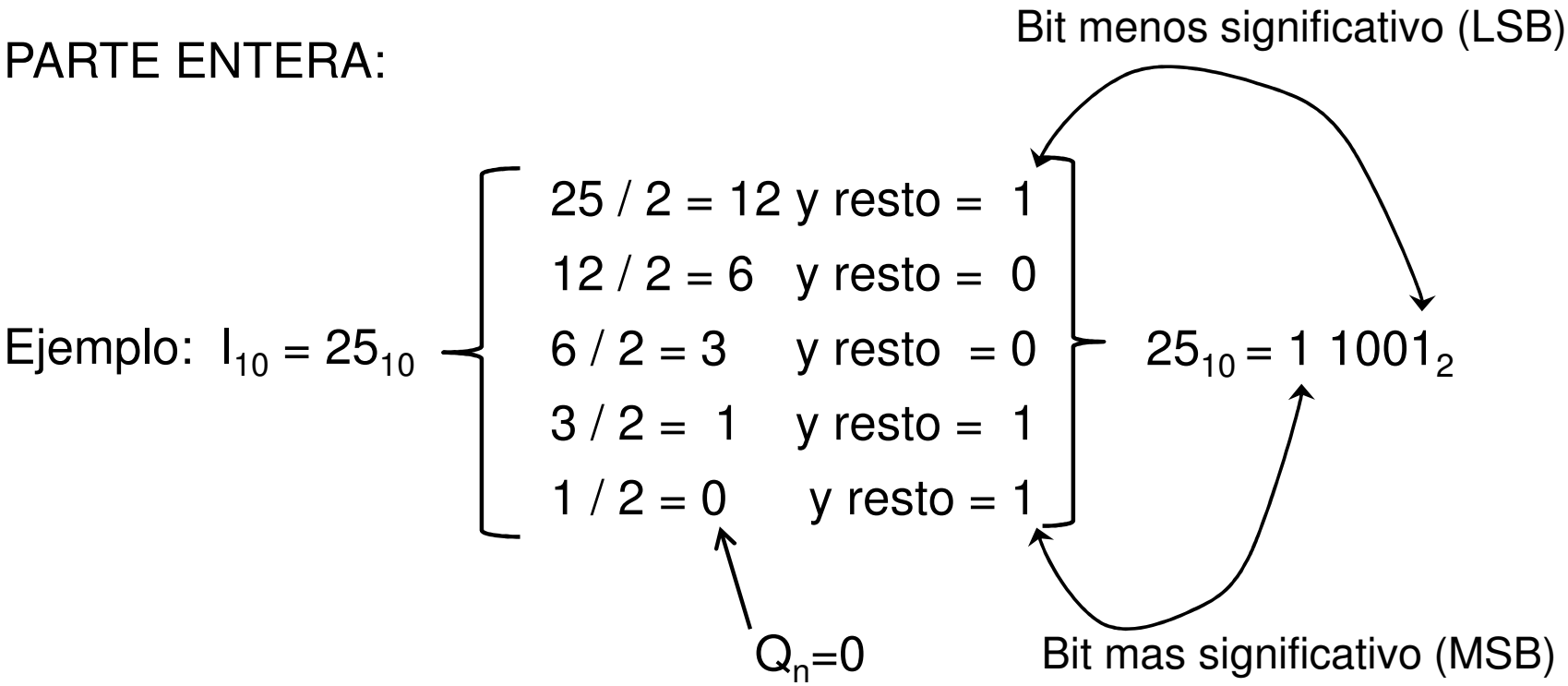
Aplico esto sucesivamente hasta que $Q_n = 0$
El número binario está dado por los restos.

Información numérica

- **Cambio de base**

De base 10 a base 2:

PARTE ENTERA:



Información numérica

- **Cambio de base**

De base 10 a base 2:

PARTE ENTERA:

OTRO METODO: Por tanteo, utilizando los pesos.

Tomemos I_{10}= 25

Busco el PESO que no supere a I_{10}. $25 \geq 2^4 = 16$ ➔ 25 -16 = 9

Hago lo mismo con la diferencia: $9 \geq 2^3 = 8$ ➔ 9 – 8 = 1

Termino cuando la diferencia da cero $1 \geq 2^0 = 1$ ➔ 1- 1 = 0

Los 1's están en la posición indicada por los exponentes: $1\ 1001_2$

Observar que : 25 = 16 + 8 + 1 , la suma de sus pesos.

Información numérica

- **Cambio de base**

De base 10 a base 2:

PARTE FRACCIONARIA: $F_{10} = F_2 = b_{-1}2^{-1} + b_{-2}\,2^{-2} + \ldots\ldots + b_{-m}\,2^{-m} + \ldots..$

Multiplico F_{10} por 2: $2 \,.\, F_{10} = b_{-1} + b_{-2}\,2^{-1} + \ldots\ldots + b_{-m}\,2^{-m+1} + \ldots\ldots\ldots\ldots$ C_1

Multiplico C_1 por 2: $2 \,.\, C_1 = b_{-2} + b_{-3}\,2^{-1} + \ldots\ldots + b_{-m}\,2^{-m+2} + \ldots\ldots\ldots\ldots$ C_2

$C_i < 1$ ➔ al multiplicar si resultado >1 , $b_i = 1$, de lo contrario $b_i = 0$

Aplico esto sucesivamente.

Si algún $C_i = 0$➔ representación exacta y terminé.

Si no encuentro $C_i = 0$➔ trunco en el dígito m y tengo un error.

$F_{10} = b_{-1}2^{-1} + b_{-2}\,2^{-2} + \ldots\ldots + b_{-m}\,2^{-m} + C_{-m}\,2^{-m}$ ← Error cometido

Representación con m bits

Información numérica

- **Cambio de base**

De base 10 a base 2:

PARTE FRACCIONARIA:

Ejemplo: $F_{10} = 0,125$ ➔

2 x 0,125 = 0 + 0,25
2 x 0,25 = 0 + 0,5
2 x 0,5 = 1 + 0

$0,125 = 0,001_2$

Sin error

Ejemplo: $F_{10} = 0,3_0$ ➔

2 x 0,3 = 0 + 0,6
2 x 0,60 = 1 + 0,2
2 x 0,2 = 0 + 0,4
2 x 0,4 = 0 + 0,8
2 x 0,8 = 1 + 0,6

No sigo más

$0,3 = 0,01001_2$

Error = $0,6 \times 2^{-5}$

El error se genera al truncar

Información numérica

- **Cambio de base**

De base 10 a base 2:

PARTE FRACCIONARIA:

En lugar de multiplicar por 2, multiplico por 2^n, lo que "me trae" **n** bits.

Tomo n=6: ($2^6 = 64$)

Ejemplo: $F_{10} = 0,125$ ➔ $64 \times 0,125 = 8 = 001000_2$ (en 6 bits)
➔ $0,125 = 0,001000$
Sin error pues 8 no tiene parte fraccionaria

Ejemplo: $F_{10} = 0,3_0$ ➔ $64 \times 0,3 = 19,2$ y $19 = 010011$ (en 6 bits)
➔ $0,3 = 0,010011$
Error = $0,2 \times 2^{-6}$