

## 2. Codificación

### Fundamentos de Informática

Especialidad de Electrónica – 2009-2010

Ismael Etxeberria Agiriano

07/10/2009



Escuela Universitaria  
de Ingeniería  
Vitoria-Gasteiz

Ingeniaritzako  
Unibertsitate Eskola  
Vitoria-Gasteiz



Universidad  
del País Vasco

Euskal Herriko  
Unibertsitatea

# Índice

## 2. Codificación

1. Definiciones
2. Bases numeración
3. Números negativos
4. Conversiones
5. Caracteres

# 1. Definiciones

- **Ordenador**
  - Máquina capaz de **procesar información**
- **Programa**
  - Secuencia de **órdenes/instrucciones** que manipulan **datos**
  - Los **programas** y los **datos** se guardan en dispositivos de **memoria** de tipos variados
  - Internamente los programas y los datos se guardan en base **binaria**, independientemente del soporte físico
- **Algoritmo**
  - **Secuencia finita y ordenada** de pasos que describe la resolución de un problema informático
  - **Determinista**: en las mismas condiciones ha de dar siempre el mismo resultado
  - Receta: ingredientes, utensilios y método de preparación

## 2. Bases de numeración

- Estamos habituados a utilizar la **base decimal** o **base 10**
  - Dígitos del **0 al 9**
  - Dígitos decimales válidos: **0 1 2 3 4 5 6 7 8 9**
- El ordenador utiliza la **base binaria** o **base 2**
  - Dígitos del **0 al 1**
  - Dígito binario: **bit**
  - Agrupación de 8 bits: **octeto** o **byte**
  - Dígitos binarios válidos: **0 1**
- Una base relacionada a la binaria es la **base octal** o **base 8**
  - Dígitos del **0 al 7**
  - Dígitos octales válidos: **0 1 2 3 4 5 6 7**
- Otra base relacionada es la **base hexadecimal** o **base 16**
  - Necesitamos 6 dígitos nuevos: **a b c d e f**
  - Dígitos del **0 al f**
  - Dígitos hexadecimales válidos: **0 1 2 3 4 5 6 7 8 9 a b c d e f**

- **Capacidad binaria (prefijos binarios)**
  - Expresada en potencias de  $2^{10}$
  - Tradicionalmente 1 Kilobyte =  $2^{10}$  bytes = 1024 bytes
  - Confusión con el Sistema Internacional
  - Kilobinario: kibi
  - 1 **Kibi**byte =  $2^{10}$  bytes = 1024 bytes  $\approx$  1000 bytes = 1 **Kilobyte**

Prefijo SI	Símbolo	Valor	Prefijo binario	Símbolo	Pot 2
Kilo	K	$10^3$	Kibi	Ki	$2^{10}$
Mega	M	$10^6$	Mebi	Mi	$2^{20}$
Giga	G	$10^9$	Gibi	Gi	$2^{30}$
Tera	T	$10^{12}$	Tebi	Ti	$2^{40}$
Peta	P	$10^{15}$	Pebi	Pi	$2^{50}$
Exa	E	$10^{18}$	Exbi	Ei	$2^{60}$
Zetta	Z	$10^{21}$	Zebi	Zi	$2^{70}$
Yotta	Y	$10^{24}$	Yobi	Yi	$2^{80}$

- **Base decimal**

- Estamos habituados a esta base
- Operamos con comodidad
- Para referirnos a otras bases utilizamos la base decimal
  - El dígito 2 no existe en binario
  - En base 7 el número 7 se representa como 10
  - En base 16 el número 16 se representa como 10
  - Si expresamos cualquier base en esa base diríamos siempre “base 10”
- Para contar en cualquier base vamos incrementando el dígito de menor peso (el de menos valor) hasta que no nos quedan más dígitos; entonces lo pondremos a cero e incrementaremos el siguiente, y así sucesivamente
- **Ejemplo:** El número 3092 representa:

$$\begin{aligned} 3 \cdot 10^3 &+ 0 \cdot 10^2 + 9 \cdot 10^1 + 2 \cdot 10^0 = \\ 3 \cdot 1000 &+ 0 \cdot 100 + 9 \cdot 10 + 2 \cdot 1 = \\ 3000 &+ 0 + 90 + 2 = 3092 \end{aligned}$$



- Conversión de base **b** a base **decimal**
  - Para expresar números en **base b** necesitamos **b** dígitos distintos:
    - Dígitos del  $v_0$  al  $v_{b-1}$
    - Dígitos válidos:  $v_0 v_1 v_2 \dots v_i \dots v_{b-2} v_{b-1}$
  - Un número  $d_{n-1} \dots d_i \dots d_1 d_0$  en base **b** de **n** cifras representa:
 
$$d_{n-1} \cdot b^{n-1} + \dots + d_i \cdot b^i + \dots + d_1 \cdot b^1 + d_0 \cdot b^0$$
  - Denominamos **peso** a la posición *i* de estos dígitos de 0 a  $n - 1$
  - Si la base  $b > 10$  (siempre razonando en decimal) algunos de estos dígitos pueden representar números mayores que 9 pero no por ello difíciles de representar (a, b, c, ...)
- **Ejemplo:** el **8af1** hexadecimal ( $b = 16$ ) se corresponde:

$$8 \cdot 16^3 + a \cdot 16^2 + f \cdot 16^1 + 1 \cdot 16^0 =$$

$$8 \cdot 4096 + 10 \cdot 256 + 15 \cdot 16 + 1 \cdot 1 =$$

$$32768 + 2560 + 240 + 1 = 35569 \text{ en decimal}$$

- De **decimal** a base **b**
  - Vamos dividiendo por la base y en el resto obtenemos los dígitos en orden inverso de menor peso a mayor peso
  - Lo más sencillo para recordar el método es suponer que la base  $b$  es 10.
- **Ejemplo:** el 8302 a decimal ( $b = 10$ ) se corresponde:

$$\begin{array}{r} 8302 \quad | \quad 10 \\ 30 \quad 830 \quad | \quad 10 \\ 02 \quad 30 \quad 83 \quad | \quad 10 \\ 2 \quad 0 \quad 3 \quad 8 \quad | \quad 10 \\ \hline \quad \quad \quad 8 \quad 0 \end{array}$$

Se lee en este orden



- **Ejemplo:** el 415 en binario ( $b = 2$ ) se corresponde:

415  $\underline{|2}$   
 015    207  $\underline{|2}$   
   1    007    103  $\underline{|2}$   
           1    03    51  $\underline{|2}$   
                   1    11    25  $\underline{|2}$   
                           1    12    12  $\underline{|2}$   
                                   1    0    6  $\underline{|2}$   
   0    3  $\underline{|2}$   
   1    1  $\underline{|2}$   
   1    0

*Se lee en este orden*

1 1 0 0 1 1 1 1 1 ¡Compruébalo!

- **Comprobación:** el  $110011111_2$  en decimal (b = 10) se corresponde:

$$1 \cdot 2^8 + 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 =$$

$$256 + 128 + 0 + 0 + 16 + 8 + 4 + 2 + 1 =$$

$$415_{10}$$

- **Octal**

– El  $110 \ 011 \ 111_2$  se corresponde:

6      3      7<sub>8</sub>

- **Hexadecimal**

– El  $1 \ 1001 \ 1111_2$  se corresponde:

1      9      f<sub>16</sub>

¡Compruébalos!

### 3. Números negativos

- ¿Signo?
  - Enteros con signo: **signed**
  - Enteros sin signo: **unsigned**
- ¿Cómo representar los números negativos?
  - Signo y magnitud
    - Se utiliza para los números en coma flotante IEEE 754
  - **Complemento a 1**
  - **Complemento a 2**
  - Desviada (biased)
    - Para el exponente de los números en coma flotante IEEE 754
    - BCD – Exceso a 8

- **Complemento a nueve (decimal)**
  - Antes de ver el complemento a 2
  - Como ejemplo, razonar en base 10 es más fácil
  - Lo que falta a cada dígito para llegar a 9
  - Importante el número de dígitos
- **Ejemplo**
  - Para  $x = 23$  con 4 dígitos
  - Si ponemos los ceros a la izquierda: 0023
  - Dígito a dígito obtenemos lo que falta para 9:
    - $9 - 0 \quad 9 - 0 \quad 9 - 2 \quad 9 - 3 = 9976$

- **Complemento a uno (binario)**
  - Lo que falta a cada dígito para llegar a 1
  - Importante el número de dígitos
- **Ejemplo**
  - Para  $x = 23$  en binario `10111` con 8 dígitos (en binario)
  - Si ponemos los ceros a la izquierda: `00010111`
  - Dígito a dígito obtenemos lo que falta para 1:
    - `00010111`
    - `11101000`

- **Complemento a dos (binario)**

- Para  $n$  dígitos podemos definir  $C_2^x$ , complemento a 2 de un número entero  $x$  como:

$$C_2^x = 2^n - x$$

- **Ejemplo**

- Para  $x = 23$  en binario  $10111$  con 8 dígitos (en binario)

$$2^8 - 10111 = 256 - 23 = 233$$

100000000 ( $2^8$ : 9 dígitos)

$$\begin{array}{r} - \quad 10111 \\ \hline 11101001 \end{array}$$

¿coinciden?



- **Complemento a dos (binario)**

- Puede obtenerse alternativamente sumando 1 al complemento a 1

- **Ejemplo**

- Para  $x = 23$  en binario 10111 con 8 dígitos (en binario)
- Ponemos los ceros a la izquierda y complementamos:

00010111

11101000 Complemento a 1

---

+ 1

11101001 Complemento a 2 (compárese con la anterior)

- **Complemento a dos (binario)**

- Comenzando por la derecha (el dígito de menos peso), copiar el número original (de derecha a izquierda) hasta encontrar el primer 1, después se niegan (complementan) los dígitos restantes

- **Ejemplo**

- Para  $x = 23$  en binario 10111 con 8 dígitos (en binario)
- Ponemos los ceros a la izquierda y complementamos:

00010111

11101001 Complemento a 2 (compárese con la anterior)

- **Ejemplo**

01011100

10100100 Complemento a 2

- **Complemento a dos (binario)**

- El *bit más significativo* **con todo su peso** es negativo
- Si el *bit más significativo* es **0** el número será **positivo**; si es **1** será **negativo**
- Salvo el último, todos los ceros y todos los unos por la izquierda no serán significativos

- **Ejemplos**

- El 01, el 001, el 0001, ... todos representan el 01
- Con 1 bit el 1 representa el -1:  $-1 \cdot 2^0 = -1$
- Con 2 bits el 11 representa el -1:  $-1 \cdot 2^1 + 1 \cdot 2^0 = -2 + 1$
- Con 3 bits el 111 representa el -1:  $-1 \cdot 2^2 + 1 \cdot 2^1 + 1 = -4 + 2 + 1$
- Con 4 bits el 1111 representa el -1:  $-8 + 4 + 2 + 1$
- Con 6 bits el 111111 representa el -1:  $-32 + 16 + 8 + 4 + 2 + 1$

## 4. Conversiones de tipos

- A menudo vamos a necesitar convertir datos de tipos distintos y a menudo esto se hará automáticamente
- Para **aumentar el tamaño** de un entero se extiende el signo, es decir, se rellena por la izquierda con el bit más significativo
- Si aumentamos de tamaño un entero siempre seguiremos representando el mismo número
- Para **disminuir el tamaño** de un entero eliminamos los bytes sobrantes de mayor peso
- Si disminuimos de tamaño un entero podemos cambiar de número e incluso pasar de positivo a negativo y viceversa
- De real a entero **se trunca**, no se redondea

## 5. Codificación de caracteres

- La más tradicional es el código ASCII (7 bits)

Dec	Hex	C	Car	Descripción	Description
0	0	NUL		carácter nulo	null
1	1	SOH		comienzo de cabecera	start of heading
2	2	STX		comienzo de texto	start of text
3	3	ETX		fin de texto	end of text
4	4	EOT		fin de transmisión	end of transmission
5	5	ENQ		petición	enquiry
6	6	ACK		reconocimiento	acknowledge
7	7	BEL		timbre	bell
8	8	BS		retroceso	backspace
9	9	TAB		tabulador horizontal	horizontal tab
10	a	LF/NL		salto de línea	line feed/new line
11	b	VF		tabulador vertical	vertical tab
12	c	FF/NP		salto de página	form feed/new page
13	d	CR		retorno de carro	carriage return
14	e	SO		cambiar conjunto de caracteres	shift out
15	f	SI		volver al conjunto de caracteres	shift in
16	10	DLE		escape de enlace de datos	data link escape
17	11	DC1		control de dispositivo 1	device control 1
18	12	DC2		control de dispositivo 2	device control 2
19	13	DC3		control de dispositivo 3	device control 3
20	14	DC4		control de dispositivo 4	device control 4
21	15	NAK		reconocimiento negativo	negative acknowledge
22	16	SYN		espera síncrona	synchronous idle
23	17	ETB		fin de bloque de transmisión	end of transmission block
24	18	CAN		cancelar	cancel
25	19	EM		fin de medio	end of medium
26	1a	SUB		substitución	substitute
27	1b	ESC		escape	escape
28	1c	FS		separador de fichero	file separator
29	1d	GS		separador de grupo	group separator
30	1e	RS		separador de registro	record separator
31	1f	US		separador de unidad	unit separator
127	7f	DEL		suprimir	delete

Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car
32	20	espacio	48	30	0	64	40	@	80	50	P	96	60	`	112	70	p
33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
36	24	\$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
42	2a	*	58	3a	:	74	4a	J	90	5a	Z	106	6a	j	122	7a	z
43	2b	+	59	3b	;	75	4b	K	91	5b	[	107	6b	k	123	7b	{
44	2c	,	60	3c	<	76	4c	L	92	5c	\	108	6c	l	124	7c	
45	2d	-	61	3d	=	77	4d	M	93	5d	]	109	6d	m	125	7d	}
46	2e	.	62	3e	>	78	4e	N	94	5e	^	110	6e	n	126	7e	~
47	2f	/	63	3f	?	79	4f	O	95	5f	_	111	6f	o			





Escuela Universitaria  
de Ingeniería  
Vitoria-Gasteiz

Ingeniaritzako  
Unibertsitate Eskola  
Vitoria-Gasteiz

eman ta zabal zazu



Universidad  
del País Vasco

Euskal Herriko  
Unibertsitatea