Práctica 1: Representación de la información

Caracteres

Organización del Computador I
DC - UBA

Verano 2018

Representación de caracteres

- La memoria contiene **bits**, ceros y unos dispuestos en algún orden que se pueden interpretar de múltiples formas.
- ¿Cómo hacemos para guardar en memoria la cadena de caracteres `orga1`?

ASCII

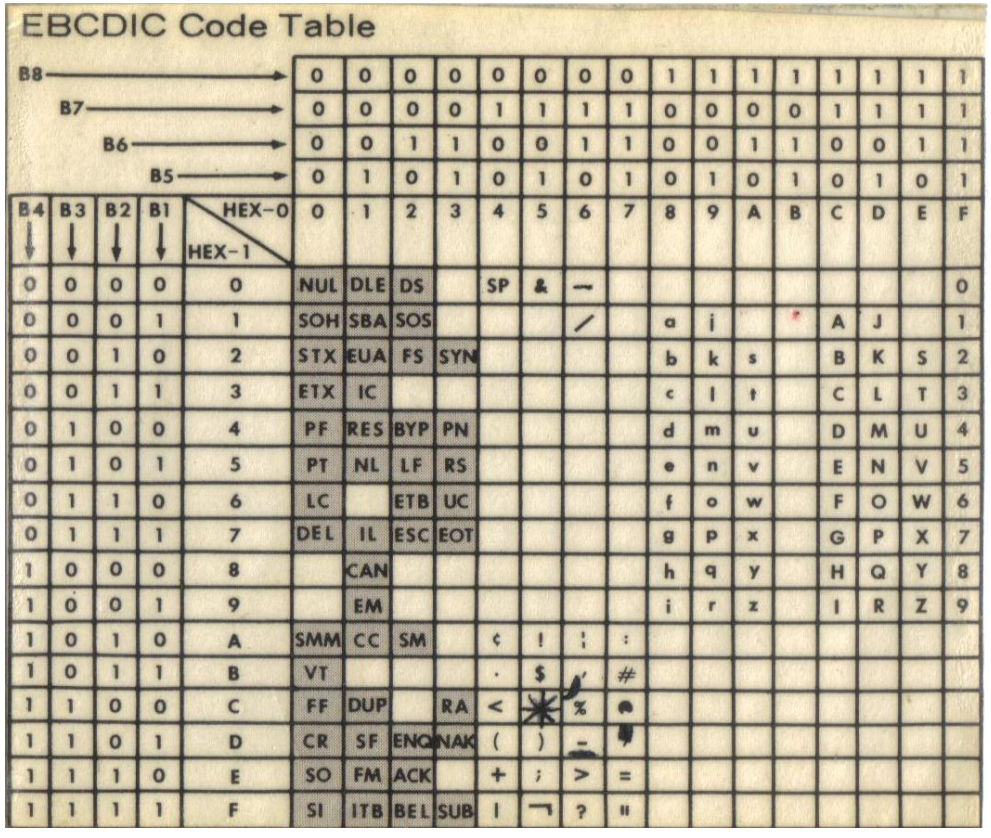
Respuesta 1: ASCII (American Standard for Information Interchange – 1963)

USASCII code chart

Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row / Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

EBCDIC

Respuesta 1 bis: EBCDIC (Extended Binary Coded Decimal Interchange Code – 1963, IBM)

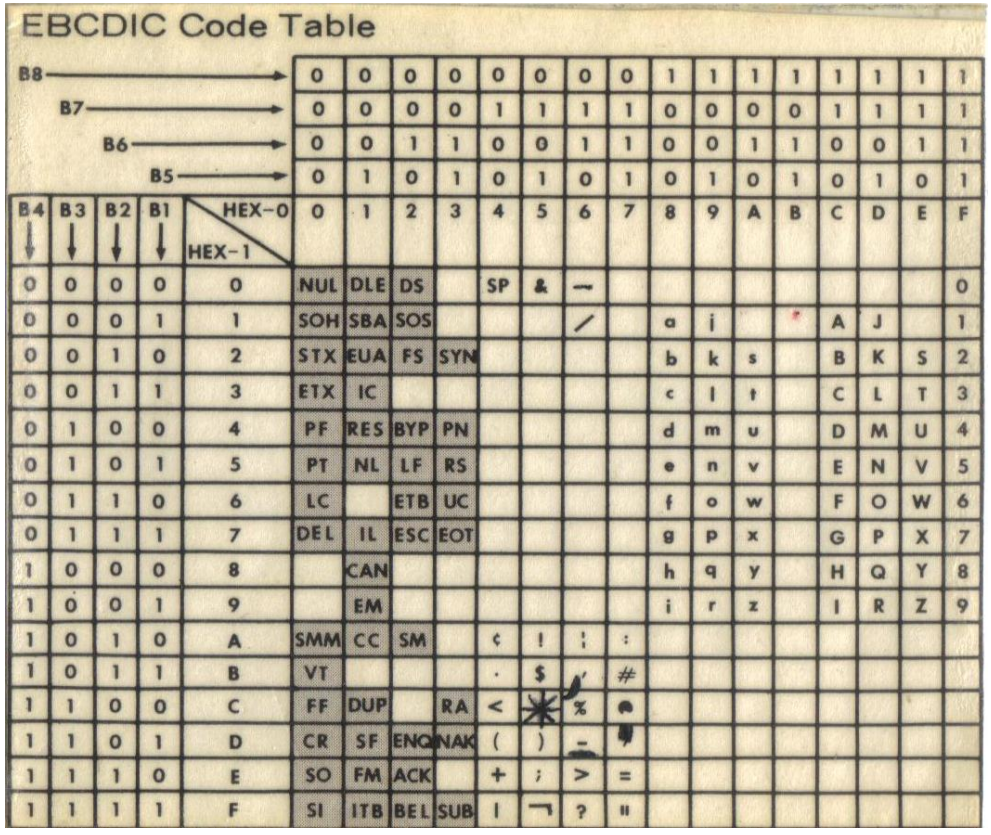


EBCDIC Code Table

B8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
B7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
B6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
B5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
B4 B3 B2 B1 HEX-1 \ HEX-0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 0 0 0 0	NUL	DLE	DS		SP	&	—									0
0 0 0 1 1	SOH	SBA	SOS				/		a	j			A	J		1
0 0 1 0 2	STX	EUA	FS	SYN					b	k	s		B	K	S	2
0 0 1 1 3	ETX	IC							c	l	t		C	L	T	3
0 1 0 0 4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0 1 0 1 5	PT	NL	LF	RS					e	n	v		E	N	V	5
0 1 1 0 6	LC		ETB	UC					f	o	w		F	O	W	6
0 1 1 1 7	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1 0 0 0 8		CAN							h	q	y		H	Q	Y	8
1 0 0 1 9		EM							i	r	z		I	R	Z	9
1 0 1 0 A	SMM	CC	SM		¢	!	¦	:								
1 0 1 1 B	VT				.	$	,	#								
1 1 0 0 C	FF	DUP		RA	<	*	%	@								
1 1 0 1 D	CR	SF	ENQ	NAK	(	)	_	'								
1 1 1 0 E	SO	FM	ACK		+	;	>	=								
1 1 1 1 F	SI	ITB	BEL	SUB	|	¬	?	"								

EBCDIC

Respuesta 1 bis: EBCDIC (Extended Binary Coded Decimal Interchange Code – 1963, IBM)



EBCDIC Code Table

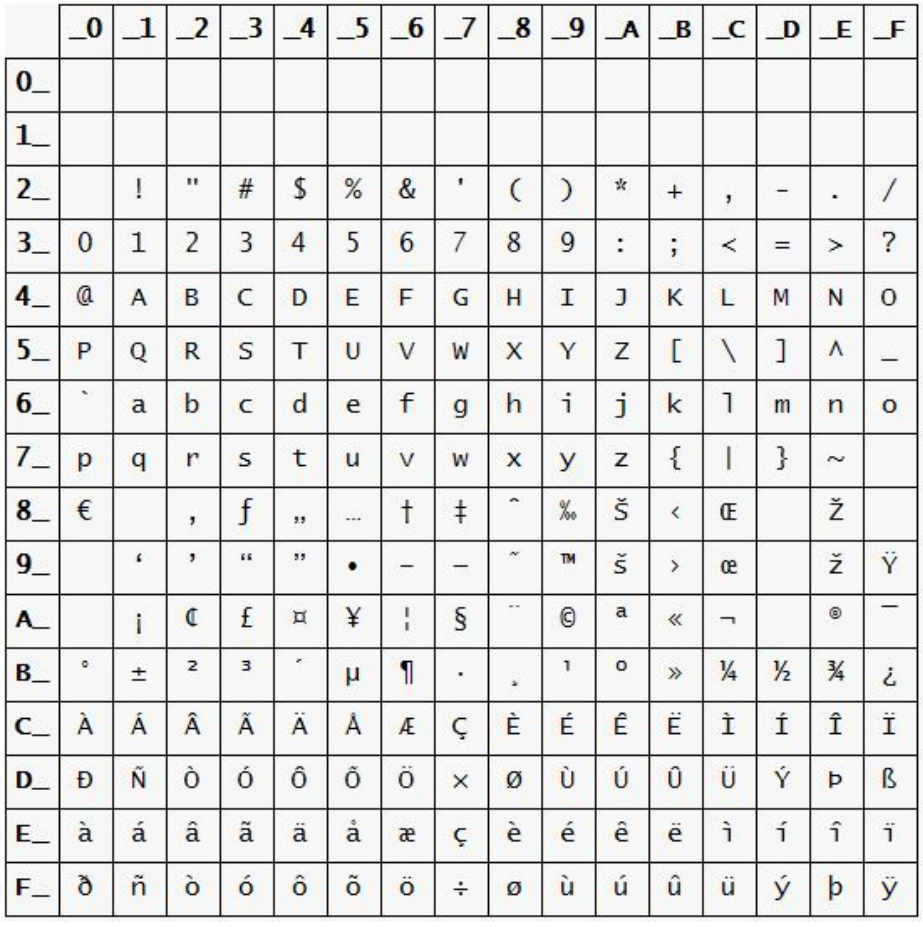
B8					0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
B7					0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	
B6					0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
B5					0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
B4	B3	B2	B1	HEX-0 / HEX-1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	0	0	0	0	NUL	DLE	DS		SP	&	—										0
0	0	0	1	1	SOH	SBA	SOS				/		a	j			A	J			1
0	0	1	0	2	STX	EUA	FS	SYN					b	k	s		B	K	S		2
0	0	1	1	3	ETX	IC							c	l	t		C	L	T		3
0	1	0	0	4	PF	RES	BYP	PN					d	m	u		D	M	U		4
0	1	0	1	5	PT	NL	LF	RS					e	n	v		E	N	V		5
0	1	1	0	6	LC		ETB	UC					f	o	w		F	O	W		6
0	1	1	1	7	DEL	IL	ESC	EOT					g	p	x		G	P	X		7
1	0	0	0	8		CAN							h	q	y		H	Q	Y		8
1	0	0	1	9		EM							i	r	z		I	R	Z		9
1	0	1	0	A	SMM	CC	SM		¢	!	¦	:									
1	0	1	1	B	VT				.	$	,	#									
1	1	0	0	C	FF	DUP		RA	<	*	%	@									
1	1	0	1	D	CR	SF	ENQ	NAK	(	)	_	'									
1	1	1	0	E	SO	FM	ACK		+	;	>	=									
1	1	1	1	F	SI	ITB	BEL	SUB	|	¬	?	"									

Problema: ¿y si quiero representar `Organización del Computador I`?

ISO-8859-1 (Latin-1)

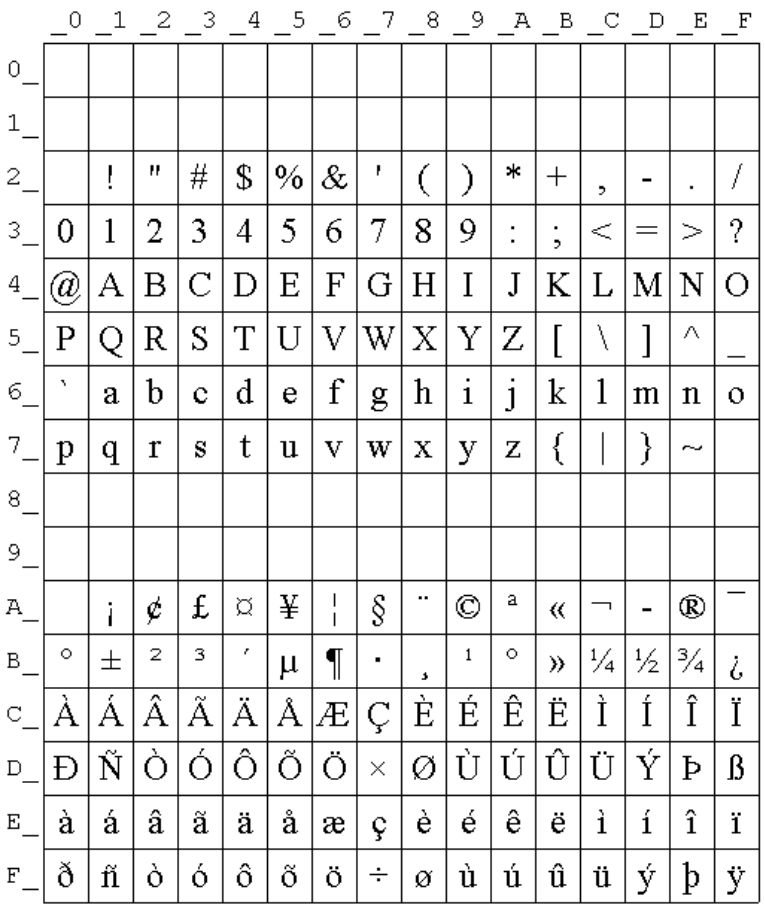
Respuesta 2: Latin-1

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_																
1_																
2_		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8_																
9_																
A_		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B_	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C_	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D_	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E_	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F_	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ



ISO-8859-1 (Latin-1)

Respuesta 2: Latin-1



	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_																
1_																
2_		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8_																
9_																
A_		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B_	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C_	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D_	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E_	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F_	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

¿Y ahora si quiero codificar 你什么名字?

Unicode

- Para unificar la codificación de caracteres en todas las lenguas del mundo se definió un standard, llamado **Unicode**, que establece una numeración uniforme para 136.690 caracteres (versión 10.0, Junio 2017) en 139 *scripts* diferentes

Unicode

- Para unificar la codificación de caracteres en todas las lenguas del mundo se definió un standard, llamado **Unicode**, que establece una numeración uniforme para 136.690 caracteres (versión 10.0, Junio 2017) en 139 *scripts* diferentes
- A esta lista de caracteres se la conoce como UCS (Universal Character Set)

Unicode

- Para unificar la codificación de caracteres en todas las lenguas del mundo se definió un standard, llamado **Unicode**, que establece una numeración uniforme para 136.690 caracteres (versión 10.0, Junio 2017) en 139 *scripts* diferentes
- A esta lista de caracteres se la conoce como UCS (Universal Character Set)
- Algunos *scripts* incluidos: Árabe, Armenio, Bengalí, Bopomofo, Cirílico, Devanagari, Georgiano, Griego, Gujarati, Gurmukhi, Hangul, Hebreo, Hiragana, Kannada, Katakana, Lao, Latín, Malayo, Oriya, Tamil, Telugu, Thai, Tibetano, CJK (chino, japonés y coreano), Sánscrito védico, Javanés, Khmer, Mongol, Tibetano, Braille, Runas, ~~Klingon~~, etc.

Unicode

- Para unificar la codificación de caracteres en todas las lenguas del mundo se definió un standard, llamado **Unicode**, que establece una numeración uniforme para 136.690 caracteres (versión 10.0, Junio 2017) en 139 *scripts* diferentes
- A esta lista de caracteres se la conoce como UCS (Universal Character Set)
- Algunos *scripts* incluidos: Árabe, Armenio, Bengalí, Bopomofo, Cirílico, Devanagari, Georgiano, Griego, Gujarati, Gurmukhi, Hangul, Hebreo, Hiragana, Kannada, Katakana, Lao, Latín, Malayo, Oriya, Tamil, Telugu, Thai, Tibetano, CJK (chino, japonés y coreano), Sánscrito védico, Javanés, Khmer, Mongol, Tibetano, Braille, Runas, ~~Klingon~~, etc.
- Unicode establece el **mapa** de caracteres, pero no cómo se codifican.

UCS y UTF

- UCS define una manera de codificar los caracteres, utilizando el número del mismo en el mapa. Para esto necesitamos al menos 17 bits ($2^{17} = 131072$).
- Las codificaciones de UCS son de **longitud fija**, se indica en el nombre la cantidad de bytes de la codificación.

UCS y UTF

- UCS define una manera de codificar los caracteres, utilizando el número del mismo en el mapa. Para esto necesitamos al menos 17 bits ($2^{17} = 131072$).
- Las codificaciones de UCS son de **longitud fija**, se indica en el nombre la cantidad de bytes de la codificación.
- Inicialmente UCS-2, luego UCS-4.

UCS y UTF

- UCS define una manera de codificar los caracteres, utilizando el número del mismo en el mapa. Para esto necesitamos al menos 17 bits ($2^{17} = 131072$).
- Las codificaciones de UCS son de **longitud fija**, se indica en el nombre la cantidad de bytes de la codificación.
- Inicialmente UCS-2, luego UCS-4.
- **Problema**: el texto "hola mundo" se codificaría en UCS-4 como:

```
00 00 00 68 | 00 00 00 6F | 00 00 00 6C | 00 00 00 61
00 00 00 20 | 00 00 00 6D | 00 00 00 75 | 00 00 00 6E
00 00 00 64 | 00 00 00 6F
```

UCS y UTF

- UCS define una manera de codificar los caracteres, utilizando el número del mismo en el mapa. Para esto necesitamos al menos 17 bits ($2^{17} = 131072$).
- Las codificaciones de UCS son de **longitud fija**, se indica en el nombre la cantidad de bytes de la codificación.
- Inicialmente UCS-2, luego UCS-4.
- **Problema**: el texto "hola mundo" se codificaría en UCS-4 como:

```
00 00 00 68 | 00 00 00 6F | 00 00 00 6C | 00 00 00 61
00 00 00 20 | 00 00 00 6D | 00 00 00 75 | 00 00 00 6E
00 00 00 64 | 00 00 00 6F
```

Mientras que en ASCII la codificación es:

```
68 6F 6C 61 | 20 6D 75 6E | 64 6F
```

UCS y UTF

- UCS define una manera de codificar los caracteres, utilizando el número del mismo en el mapa. Para esto necesitamos al menos 17 bits ($2^{17} = 131072$).
- Las codificaciones de UCS son de **longitud fija**, se indica en el nombre la cantidad de bytes de la codificación.
- Inicialmente UCS-2, luego UCS-4.
- **Problema**: el texto "hola mundo" se codificaría en UCS-4 como:

```
00 00 00 68 | 00 00 00 6F | 00 00 00 6C | 00 00 00 61
00 00 00 20 | 00 00 00 6D | 00 00 00 75 | 00 00 00 6E
00 00 00 64 | 00 00 00 6F
```

Mientras que en ASCII la codificación es:

```
68 6F 6C 61 | 20 6D 75 6E | 64 6F
```

- **Solución**: usar codificaciones de longitud variable (**UTF** = **U**CS **T**ransformation **F**ormat)

UTF

UTF define diversos sistemas de transformación del código de un caracter a su representación en memoria. Para indicar el tipo, se suele utilizar un número que representa la cantidad de **bits** que tiene cada *unidad de código* (code unit) del formato.

Los formatos UTF más utilizados son UTF-32, UTF-16 y **UTF-8**.

UTF

UTF define diversos sistemas de transformación del código de un caracter a su representación en memoria. Para indicar el tipo, se suele utilizar un número que representa la cantidad de **bits** que tiene cada *unidad de código* (code unit) del formato.

Los formatos UTF más utilizados son UTF-32, UTF-16 y **UTF-8**.

UTF-32 equivalente a UCS-4.

UTF

UTF define diversos sistemas de transformación del código de un caracter a su representación en memoria. Para indicar el tipo, se suele utilizar un número que representa la cantidad de **bits** que tiene cada *unidad de código* (code unit) del formato.

Los formatos UTF más utilizados son UTF-32, UTF-16 y **UTF-8**.

UTF-32 equivalente a UCS-4.

UTF-16 toma como base UCS-2, utiliza 2 code units de 16 bits para codificar caracteres cuyo código tiene un valor superior a `U+10000`.

UTF

UTF define diversos sistemas de transformación del código de un caracter a su representación en memoria. Para indicar el tipo, se suele utilizar un número que representa la cantidad de **bits** que tiene cada *unidad de código* (code unit) del formato.

Los formatos UTF más utilizados son UTF-32, UTF-16 y **UTF-8**.

UTF-32 equivalente a UCS-4.

UTF-16 toma como base UCS-2, utiliza 2 code units de 16 bits para codificar caracteres cuyo código tiene un valor superior a `U+10000`.

UTF-8 toma como base ASCII, codificación de longitud variable entre 1 y 4 code units de 8 bits.

Codificación en UTF-8

En UTF-8 se utiliza la siguiente tabla para codificar caracteres Unicode:

Bits	Inicio	Fin	Byte 1	Byte 2	Byte 3	Byte 4
1–7	U+0000	U+007F	0xxxxxxx			
8–11	U+0080	U+07FF	110xxxxx	10xxxxxx		
12–16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
17–21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Vamos a codificar el texto 名字, que corresponde a los caracteres Unicode
U+540D U+5B57[1]

[1] http://www.fileformat.info/info/unicode/char/search.htm

Codificación en UTF-8 (cont.)

Primero escribimos ambos números en binario:

Codificación en UTF-8 (cont.)

Primero escribimos ambos números en binario:

$$\texttt{540D} = \texttt{0101 0100 0000 1101}$$
$$\texttt{5B57} = \texttt{0101 1011 0101 0111}$$

Codificación en UTF-8 (cont.)

Primero escribimos ambos números en binario:

`540D` = `0101 0100 0000 1101`
`5B57` = `0101 1011 0101 0111`

Como ocupan 15 bits, tenemos que usar la codificación de 3 bytes. La escribimos:

Codificación en UTF-8 (cont.)

Primero escribimos ambos números en binario:

`540D` = `0101 0100 0000 1101`
`5B57` = `0101 1011 0101 0111`

Como ocupan 15 bits, tenemos que usar la codificación de 3 bytes. La escribimos:

`540D` ⇒ `1110 0101 1001 0000 1000 1101` ⇒ `E5 90 8D`
`5B57` ⇒ `1110 0101 1010 1101 1001 0111` ⇒ `E5 AD 97`

Decodificación de UTF-8

Supongamos que tenemos la siguiente cadena de bytes y la queremos interpretar como UTF-8:

```
63 C3 B6 E4 BA 9C F0 9D 84 9E
```

Decodificación de UTF-8

Supongamos que tenemos la siguiente cadena de bytes y la queremos interpretar como UTF-8:

`63 C3 B6 E4 BA 9C F0 9D 84 9E`

- 63, en binario 0110 0011 $\Rightarrow$ U+0063

Decodificación de UTF-8

Supongamos que tenemos la siguiente cadena de bytes y la queremos interpretar como UTF-8:

`63 C3 B6 E4 BA 9C F0 9D 84 9E`

- 63, en binario 0110 0011 $\Rightarrow$ U+0063
- C3, en binario 1100 0011.

Decodificación de UTF-8

Supongamos que tenemos la siguiente cadena de bytes y la queremos interpretar como UTF-8:

`63 C3 B6 E4 BA 9C F0 9D 84 9E`

- 63, en binario `0110 0011` $\Rightarrow$ `U+0063`
- C3, en binario `1100 0011`. Tenemos que leer 1 byte más.

Decodificación de UTF-8

Supongamos que tenemos la siguiente cadena de bytes y la queremos interpretar como UTF-8:

`63 C3 B6 E4 BA 9C F0 9D 84 9E`

- 63, en binario 0110 0011 $\Rightarrow$ U+0063
- C3, en binario 1100 0011. Tenemos que leer 1 byte más.
- B6, en binario 1011 0110 $\Rightarrow$ U+00F6

Decodificación de UTF-8

Supongamos que tenemos la siguiente cadena de bytes y la queremos interpretar como UTF-8:

`63 C3 B6 E4 BA 9C F0 9D 84 9E`

- 63, en binario 0110 0011 $\Rightarrow$ U+0063
- C3, en binario 1100 0011. Tenemos que leer 1 byte más.
- B6, en binario 1011 0110 $\Rightarrow$ U+00F6
- E4, en binario 1110 0100. Tenemos que leer 2 bytes más.

Decodificación de UTF-8

Supongamos que tenemos la siguiente cadena de bytes y la queremos interpretar como UTF-8:

`63 C3 B6 E4 BA 9C F0 9D 84 9E`

- 63, en binario 0110 0011 $\Rightarrow$ U+0063
- C3, en binario 1100 0011. Tenemos que leer 1 byte más.
- B6, en binario 1011 0110 $\Rightarrow$ U+00F6
- E4, en binario 1110 0100. Tenemos que leer 2 bytes más.
- BA 9C, en binario 1011 1010 1001 1100 $\Rightarrow$ U+4E9C

Decodificación de UTF-8

Supongamos que tenemos la siguiente cadena de bytes y la queremos interpretar como UTF-8:

`63 C3 B6 E4 BA 9C F0 9D 84 9E`

- 63, en binario 0110 0011 $\Rightarrow$ U+0063
- C3, en binario 1100 0011. Tenemos que leer 1 byte más.
- B6, en binario 1011 0110 $\Rightarrow$ U+00F6
- E4, en binario 1110 0100. Tenemos que leer 2 bytes más.
- BA 9C, en binario 1011 1010 1001 1100 $\Rightarrow$ U+4E9C
- F0, en binario 1111 0000. Tenemos que leer 3 bytes más.

Decodificación de UTF-8

Supongamos que tenemos la siguiente cadena de bytes y la queremos interpretar como UTF-8:

```
63 C3 B6 E4 BA 9C F0 9D 84 9E
```

- 63, en binario 0110 0011 ⇒ U+0063
- C3, en binario 1100 0011. Tenemos que leer 1 byte más.
- B6, en binario 1011 0110 ⇒ U+00F6
- E4, en binario 1110 0100. Tenemos que leer 2 bytes más.
- BA 9C, en binario 1011 1010 1001 1100 ⇒ U+4E9C
- F0, en binario 1111 0000. Tenemos que leer 3 bytes más.
- 9D 84 9E, en binario 1001 1101 1000 0100 1001 1110 ⇒ U+1D11E

Buscando en la tablita

Ahora que tenemos los caracteres en Unicode, hay que buscar en la tabla qué quieren decir. Nos quedó:

```
U+0063 U+00F6 U+4E9C U+1D11E
```

Buscando en la tablita

Ahora que tenemos los caracteres en Unicode, hay que buscar en la tabla qué quieren decir. Nos quedó:

```
U+0063 U+00F6 U+4E9C U+1D11E
```

Una página que permite buscar caracteres fácilmente teniendo el code point es `http://codepoints.net/`

```
U+0063 LATIN SMALL LETTER C
U+00F6 LATIN SMALL LETTER O WITH DIAERESIS
U+4E9C CJK UNIFIED IDEOGRAPH-4E9C
U+1D11E MUSICAL SYMBOL G CLEF
```

Buscando en la tablita

Ahora que tenemos los caracteres en Unicode, hay que buscar en la tabla qué quieren decir. Nos quedó:

```
U+0063 U+00F6 U+4E9C U+1D11E
```

Una página que permite buscar caracteres fácilmente teniendo el code point es `http://codepoints.net/`

```
U+0063 LATIN SMALL LETTER C
U+00F6 LATIN SMALL LETTER O WITH DIAERESIS
U+4E9C CJK UNIFIED IDEOGRAPH-4E9C
U+1D11E MUSICAL SYMBOL G CLEF
```

El texto final es: cö亜𝄞

Unicode en la vida computacional

- El uso más preponderante de UTF-8 se encuentra en la web.
- En Marzo de 2015, el 83,3 % de las páginas web se encontraban codificadas en UTF-8.

Unicode en la vida computacional

- El uso más preponderante de UTF-8 se encuentra en la web.
- En Marzo de 2015, el 83,3 % de las páginas web se encontraban codificadas en UTF-8.
- A pesar de esto, todavía hay ocasiones en las que la codificación de un texto no queda claro cuál es.

Unicode en la vida computacional

- El uso más preponderante de UTF-8 se encuentra en la web.
- En Marzo de 2015, el 83,3 % de las páginas web se encontraban codificadas en UTF-8.
- A pesar de esto, todavía hay ocasiones en las que la codificación de un texto no queda claro cuál es.
- UTF-32 se utiliza para guardar cadenas de caracteres en memoria, UTF-8 para almacenar y transferir por Internet (en Windows se usa UTF-16 también).