	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD COMPETENCIA	Sistemas de Numeración				

Índice

Sistemas de numeración (Resumen)	2
1. Introducción	2
2. Sistemas de numeración	2
2.1. Unidades.....	2
2.2. Sistema binario	3
2.2.1. Conversión binario a decimal	3
2.2.2. Conversión decimal a binario	3
2.2.3. Representación de números binarios con signo.	3
2.3. Sistema Octal	5
2.3.1. Conversión octal-binario:	5
2.3.2. Conversión decimal-octal:	5
2.4. Sistema Hexadecimal.....	6
2.4.1. Conversión binario-hexadecimal:	6
2.4.2. Conversión decimal-haxadecimal:.....	6
3. Lógica y aritmética en el sistema binario (álgebra de Boole).....	7
3.1. Lógica	7
3.1.1. Operación de no (NOT): Convierte un valor binario en el opuesto.....	7
3.1.2. Operación Y (AND):.....	7
3.1.3. Operación O (OR):	7
3.1.4. Operación O-Exclusiva (X-OR):	7
3.2. Operaciones.....	8
3.2.1. Aritmética	8
4. Códigos alfanuméricos	9

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD COMPETENCIA	Sistemas de Numeración				

Sistemas de numeración (Resumen)

Existen 10 tipos de personas: las que saben binario y las que no.

1. Introducción

Cuestión: ¿Qué lenguaje entiende un ordenador?

Un ordenador funciona ejecutando instrucciones que conforman un programa informático. Además esas instrucciones manejan datos que el ordenador tiene guardados en su memoria.

Pero para que el ordenador entienda esas instrucciones y esos datos deben estar escritos en el único lenguaje real que entienden: el lenguaje máquina. Dicho lenguaje máquina se escribe mediante código binario (unos y ceros).

http://es.wikipedia.org/wiki/Lenguaje_de_máquina

Pero ¿cómo se puede trabajar sólo con dos números (0 y 1)?

2. Sistemas de numeración

Cuestión: ¿Cómo contáis habitualmente? ¿Cuántos símbolos distintos se usan? ¿Cómo funciona? **Sistema**

decimal: Es un sistema posicional de 10 símbolos. Posicional indica que el símbolo más a la derecha es el de menor peso (unidades), el siguiente tiene un peso mayor (decenas), etc...

$$1342 = 2 + 40 + 300 + 1000 = 2 \times 10^0 + 4 \times 10^1 + 3 \times 10^2 + 1 \times 10^3$$

Otros sistemas posicionales funcionan igual que el decimal, pero usan un número distinto de símbolos.

Ese número de símbolos se denomina base.

Cuestión: ¿Conoces algún sistema NO posicional?

2.1. Unidades

Bit: unidad mínima de información. Es un 0 ó un 1.

Byte: agrupación de 8 bits. Se toma como unidad básica de medida de memoria.

Kbyte: 1024 bytes. En almacenamiento.

Mbyte: 1024 Kbytes

No se debe confundir unidades de información con velocidades de transmisión: **Kbps:** 1000 bits por segundo o 125 bytes por segundo, así una conexión de 256 Kbps (ADSL), da 32 Kbytes por segundo.

Cuestión: ¿Qué otras unidades conoces?

Según un estándar de 1998 se le cambia el nombre de Kb a KiB (Kibibyte), de Mb a MiB (Mebibyte), etc... dejando Kb para 1000 bytes o Mb para 1000000 de bytes según el SI (Sistema Internacional), pero con gran desacuerdo en la comunidad informática ya que se vio como estrategia comercial para que los fabricantes pudieran vender "números más grandes".

Es decir, si un disco duro tiene 100000000000 bytes, al dividirlo entre 1000000000 nos daría 100 Gb, sin embargo si lo dividimos entre 10243 nos daría en torno a 93Gb (o GiB) lo que no vende tanto. Entonces se "deja" el término más conocido (como GB) para uso comercial y el nuevo (GiB) para uso técnico.

Nosotros indistintamente que hablemos de GB o GiB tomaremos el estándar de 1024 como término de división.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD	COMPETENCIA	Sistemas de Numeración			

2.2. Sistema binario

Cuestión: ¿Qué pasaría si sólo tuviéramos dos símbolos? Intenta numerar del 1 al 10 con solo dos símbolos, el 0 y el 1.

Al numerar con solo dos símbolos, simplemente ocurre lo mismo que en el caso decimal, al acabarse los símbolos, volvemos al primer símbolo y añadimos uno nuevo a la izquierda: 0, 1, 10, 11, 100, 101,...

Cuestión: ¿Por qué se usa sólo el 1 y el 0?

El motivo principal es tecnológico. Es fácil realizar circuitos que tomen sólo dos estados, como por ejemplo +5Voltios y 0 Voltios (On/off). Además es más fuerte frente a cambios de tensión.

2.2.1. Conversión binario a decimal

Como el binario es también un sistema posicional, se hace lo mismo que con el decimal pero con base 2:

$$1101_2 = 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 = 1 + 0 + 4 + 8 = 13_{10}$$

$$10,11_2 = 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-2} = 0 + 2 + 1/2 + 1/4 = 2,75_{10}$$

2.2.2. Conversión decimal a binario

Se divide sucesivamente el número entre 2 y nos quedamos con los restos:

$$28:2 \quad \text{Cociente}=14 \quad \text{Resto}=0$$

$$14:2 \quad \text{Cociente}=7 \quad \text{Resto}=0$$

$$7:2 \quad \text{Cociente}=3 \quad \text{Resto}=1$$

$$3:2 \quad \text{Cociente}=1 \quad \text{Resto}=1$$

Se coge el último cociente y la ristra de restos a la inversa:

$$28_{10} = 11100_2$$

Para decimales se multiplica en lugar de dividir:

$$0,825 \times 2 = 1,65$$

$$0,65 \times 2 = 1,3$$

$$0,3 \times 2 = 0,6$$

$$0,6 \times 2 = 1,2...$$

Un número decimal con número de cifras decimales finitas no tiene por qué tener un número de cifras "decimales-binarias" finitas.

Por lo tanto:

$$0,825_{10} = 0,1101_2...$$

Ejercicios:

1. Pasar a binario los números: 5.5, 23.8, 48, 128.25 y 430 en base 10. En los que tienen parte decimal detecta cuando crees que debes parar de obtener cifras decimales.

2. Pasar a decimal los números: 1010.01, 111000, 1000010, 110011, 100.11 en base 2.

Comprueba que tus cálculos son correctos con la calculadora.

2.2.3. Representación de números binarios con signo.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD COMPETENCIA	Sistemas de Numeración				

Un caso particular son los números negativos, está claro que en un ordenador no podemos poner el signo – antes de del número en binario, entonces, ¿cómo se representan? Existen varias formas que se pueden consultar en https://es.wikipedia.org/wiki/Representaci%C3%B3n_de_n%C3%BAmeros_con_signo, nosotros nos vamos a centrar en 2:

2.2.3.1. Signo y magnitud

Esta representación nos permite codificar un número entero en binario con 8 bits (un byte). Esto nos otorga 1 bit para el signo y 7 bits para la magnitud. Con 8 bits, podemos representar, $2^8 = 256$ números. Los cuales, según este formato, van a estar repartidos entre 128 números positivos (bit de signo en 0) y 128 números negativos (bit de signo en 1). En la práctica hay doble representación del 0 con lo que el número de números que podemos representar queda en 255.

En este método se utiliza:

1. Un bit para representar el signo, habitualmente el bit del extremo izquierdo también conocido como bit más significativo (MSB)
0 => número positivo
1 => número negativo
2. Los n-1 bits restantes representan en valor absoluto el número.

Ejemplos de uso.

Representación del número -37_{10} en binario.

Pasaríamos 37_{10} a binario y añadiríamos un bit de signo 1100101_2

Representación del número 101100101_2 en decimal

Separaríamos el bit más significativo (bit de signo), y convertiríamos el resto del número a su valor decimal, en este caso tendríamos $1100101_2 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^2 + 1 \times 2^0 = 101$, teniendo en cuenta el bit de signo el resultado sería -101

2.2.3.2. Complemento a uno

El complemento a uno de un número binario nos permite obtener la representación binaria de números negativos. Se obtiene al cambiar cada uno de los dígitos del número binario N por su complementario, esto es, cambiar los unos por ceros y los ceros por unos.

Este método utiliza:

1. Un bit para representar el signo. Ese bit a menudo es el bit más significativo y, por convención: un 0 denota un número positivo, y un 1 denota un número negativo;
2. Los (n-1)-bits restantes para representar el significando que es el valor del número en valor absoluto para el caso de números positivos, o bien, en el complemento a uno del valor absoluto del número, en caso de ser negativo.

Ejemplos de uso

Representación del número -37_{10} en binario.

- Como el número es negativo, llevará el bit de signo a 1
- El número a continuación del bit de signo, deberá expresarse en complemento a uno. Es decir- $37_{01} = |-37_{10}| = 37_{10} = 100101_2$. El complemento a uno de este número es 011010_2
- Llegamos a la cifra completa, 11011010_2 . Donde el 1 en el bit más significativo indica un número negativo, y el resto es el complemento a uno del valor absoluto del número.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD COMPETENCIA	Sistemas de Numeración				

Representación en decimal de 11010101_2 , número binario en Ca_1

- Ya que el bit más significativo es 1, sabemos que es un número negativo
- Cogemos el resto del número y complementamos a uno, en este caso quedaría 0101010_2
- Hallamos su valor en decimal $0101010_2 = 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^1 = 42_{10}$
- Dado que el bit de signo es negativo, el número en decimal sería -42_{10}
- Si el número hubiese sido positivo, bit de signo 0, no se hubiese complementado el número. Ejemplo $01010101_2 = 85_{10}$

2.3. Sistema Octal

Actividad: Trata de contar hasta 20 sólo con 8 símbolos

Efectivamente el sistema octal es un sistema con 8 símbolos, del 0 al 7. Cuando se llega al 7_8 , el siguiente es el 10_8 que equivale al 8 decimal.

Tabla de octal-binario

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

2.3.1. Conversión octal-binario:

El motivo de uso de este sistema es la facilidad de conversión al binario, ya que cada símbolo equivale a grupos de 3 bits, y se puede usar la tabla anterior:

Para convertir $325,6_8 = 011\ 010\ 101, 110 = 11010101,11_2$

A la inversa: $1011,01_2 = 001\ 011, 010 = 13,2_8$

2.3.2. Conversión decimal-octal:

Para pasar de decimal a octal usamos el método de la división:

$1036:8$ Cociente=129 resto=4
 $129:8$ Cociente=16 resto=1
 $16:8$ Cociente=2 resto=0

Por tanto $1036_{10} = 2014_8$

Y a la inversa el método de la base, que esta vez es 8:

$$354_8 = 4 \times 8^0 + 5 \times 8^1 + 3 \times 8^2 = 4 + 40 + 192 = 236_{10}$$

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD COMPETENCIA	Sistemas de Numeración				

2.4. Sistema Hexadecimal

Ahora usamos 16 símbolos, como los números no nos llegan, tomamos las primeras letras del abecedario.

Tabla de hexadecimal-binario

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

2.4.1. Conversión binario-hexadecimal:

Se agrupan de 4 en 4 bits y se ve la tabla anterior:

$$9A_{16} = 1001\ 1010 = 10011010_2$$

2.4.2. Conversión decimal-hexadecimal:

Se puede usar las reglas de la división y de la base, pero es más cómodo hacer una conversión previa al binario.

Ejercicios:

1. Indica cuales de los siguientes números no podrían ser octales especificando el motivo: 1, 78, 142, A9, 35294, 100110, 8.
2. Pasa de octal a binario y a decimal: 5, 23, 50, 111, 1026.
3. Pasa de hexadecimal a binario, a decimal y a octal: D, FF, 13, 9B, 1E3
4. Pasa de octal a hexadecimal: 561, 132, 10.
5. Pasa de binario a octal y la hexadecimal: 10001, 11111001, 101010001, 11
6. Pasa de decimal a binario, hexadecimal y octal: 1024, 200, 123

Crea un sistema en base 4 con ejemplos de conversión a decimal y viceversa. Encuentra también una forma sencilla de conversión a binario.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD COMPETENCIA	Sistemas de Numeración				

3. Lógica y aritmética en el sistema binario (álgebra de Boole)

3.1. Lógica

Una operación lógica es una operación sobre 1 bit o entre 2 bits.

Son operaciones del estilo verdadero-falso de la lógica clásica:

3.1.1. Operación de no (NOT): Convierte un valor binario en el opuesto.

Se representa cómo NOT

not 0=1

not 1=0

En lugar del not, puedes poner una línea encima del bit.

3.1.2. Operación Y (AND):

El resultado es 1 si los dos operadores son 1. (Para más de 2 operadores, todos deben ser 1)

Op1	Op 2	AND
0	0	0
0	1	0
1	0	0
1	1	1

3.1.3. Operación O (OR):

El resultado es 1 si por lo menos uno de los dos operadores es 1 (Para más operadores, llega con que uno sea 1)

Op1	Op 2	OR
0	0	0
0	1	1
1	0	1
1	1	1

3.1.4. Operación O-Exclusiva (X-OR):

El resultado es 1 sólo si uno de los dos operandos es 1 y el otro es 0 (para varios operadores, el número de 1's debe ser impar)

Op1	Op 2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD COMPETENCIA	Sistemas de Numeración				

3.2. Operaciones

Se puede operar sobre varios bits.

	110		110		110		
Y	101		O	101		XOR	101
	---		---		---		
	100		111		011		

Ejemplo codificación XOR: Dato (110), clave (101), dato encriptado (011), desencriptación (Xor del encriptado con la clave).

```

      011
XOR  101
    ---
    110 -> Se recupera el dato original

```

Ejemplo: Las operaciones lógicas que se pueden hacer en Java mediante los operadores de bit.

3.2.1. Aritmética

Las sumas, restas, multiplicaciones y divisiones se realizan igual que en decimal pero teniendo en cuenta que sólo tenemos dos símbolos. Lo vemos con ejemplos:

Suma: 12+7=19

```

1100
+111
----
10011

```


Resta: 12-7=5.

Si la cifra del minuendo es menor que la del sustraendo se coloca la diferencia y lleva 1 que se suma a la cifra del sustraendo siguiente.

```

Minuendo      1 1 0 0
Sustraendo    - 1 1 1 1
-----
              0 1 0 1

```


	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD COMPETENCIA	Sistemas de Numeración				

Multiplicación: $12 * 7 = 84$

```

      1100
    x 111
    -----
      1100
     1100
    1100
    -----
   1010100

```

División: $12/3=4$

```

  1100 | 11
  ----
  0000 100

```

$26/2=13$

```

  11010 | 10
  -----
  010    1101
    0010
      00

```

En este caso también se podría eliminar los 0's de la derecha

Ejercicios:

1. Opera:

- $(A9_{16} \text{ AND NOT } 1F_{16}) \text{ OR } 66_8$
- $\text{NOT } 10_{10} \text{ XOR } 5_{16}$

2. Pasa de decimal a binario y haz las siguientes operaciones:

- $120+100$
- $34+99$
- $112-99$
- $50-25$
- $20*5$
- $50/2$
- $64/4$

4. Códigos alfanuméricos

Cuestión: ¿Cómo representan letras los computadores?

Como los computadores trabajan sólo con números, hay que realizar lo que se llama una "codificación", es decir, acordar de alguna forma que ciertos valores equivalen a determinado símbolo alfabético.

COLEXIO VIVAS .S.L	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD COMPETENCIA	Sistemas de Numeración				

El estándar más usado al principio de la computación fue el código ASCII (código de 7 bits). Este incluía básicamente los principales símbolos alfanuméricos americanos.

Para ver dicho sistema de codificación visita esta web:

<http://www.neurophys.wisc.edu/comp/docs/ascii/>

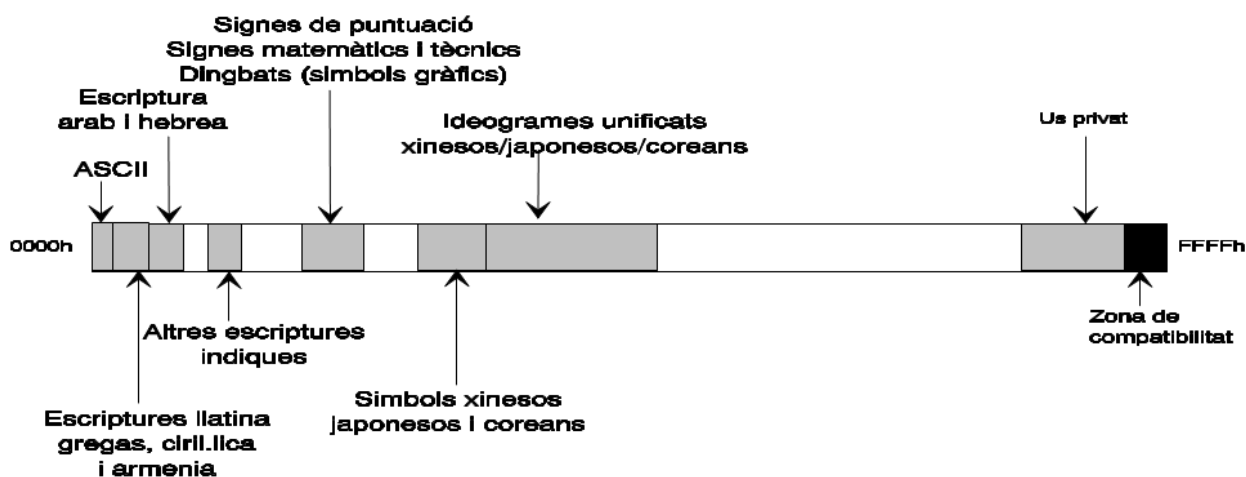
Este sistema de codificación se quedó corto ya que había muchos lenguajes con su propia simbología (como por ejemplo nosotros tenemos ñ, Á, ü, €, etc...) lo que llevó a extender el ASCII a 8 bits con el que se podía representar 256 caracteres. Cuando en un teclado pulsase una tecla, envía el número correspondiente al código ASCII del símbolo de la tecla pulsada.

Actividad: Ver Mapa de Caracteres (Charmap) en el Windows

Como puede verse, el ASCII extendido tiene la problemática de que no es totalmente estándar en los últimos 128 símbolos ya que cambia según la región: hay un ASCII para Europa occidental, otro para Rusia, etc...

Y por esto es que actualmente está empezando a usarse el UNICODE en sus distintas versiones, que es un estándar que permite almacenar cualquier símbolo de cualquier lenguaje del mundo. Así no sólo se permiten los caracteres latinos, sino toda la simbología mundial de los distintos idiomas (ruso, japonés, hebreo, árabe, chino, ...) e incluye también otras simbologías como la matemática o la musical.

Está en principio compuesto de los valores de 0x0 al 0x10FFFF (111411210 posibilidades) por lo que cada carácter ocupa para el uso diario demasiado lo que lleva a crear distintas formas de codificación.



Para codificar unicode se usan varios sistemas. UTF-8 y UTF-16 son hoy en día los más extendidos. En el caso de UTF-8 se usan bloques de 8 bits para representar símbolos. Alguno les llega un bloque (en concreto el ASCII-US es totalmente compatible con este sistema). Y en caso de que el valor sea mayor de

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
	MÓDULO	Sistemas Informáticos				CURSO: 1
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
	UNIDAD COMPETENCIA	Sistemas de Numeración				

256 entonces se envían varios bloques. El UTF-8 es el más usado en programación hoy en día. En el caso del UTF-16 es similar pero cada bloque es de 16 bits.

Veamos la siguiente tabla (fuente Wikipedia):

Rango de Código UNICODE hexadecimal	UTF-16	UTF-8	Notas
000000 - 00007F	00000000 0xxxxxxx	0xxxxxxx	Rango equivalente a US-ASCII. Símbolos de un único byte donde el bit más significativo es 0
000080 - 0007FF	00000xxx xxxxxxx	110xxxxx 10xxxxxx	Símbolos de dos bytes. El primer byte comienza con 110, el segundo byte comienza con 10
000800 - 00FFFF	xxxxxxx xxxxxxx	1110xxxx 10xxxxxx 10xxxxxx	Símbolos de tres bytes. El primer byte comienza con 1110, los bytes siguientes comienzan con 10
010000 - 10FFFF	110110xx xxxxxxx 110111xx xxxxxxx	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	Símbolos de cuatro bytes. El primer byte comienza con 11110, los bytes siguientes comienzan con 10

Por ejemplo (se usa el prefijo 0x como indicador de byte hexadecimal):

"a" (U+0061) se almacena en UTF-8 cómo 0x61. En UTF-16 se almacena cómo 0x00 0x61

"ñ" (U+00F1) se almacena en UTF-8 cómo 0xc3 0xb1. En UTF-16 se almacena cómo 0x00 0xf1

Habitualmente se escribe el unicode con el llamado code point U+ y se añaden 4 cifras hexadecimales en el caso valores menores o iguales a 16 bits y el número necesario de cifras en el caso de números mayores.

Actividad: Comprueba que la ñ cumple la tabla anterior.

¿Cómo se codificaría € en UTF-16 y UTF-8?

Esto debemos tenerlo en cuenta al programar aplicaciones. Muchas de ellas utilizan de forma nativa UNICODE por lo que estas dos líneas en Java no sería iguales:

```
int size1 = "niño".length(); // Cuenta caracteres

int size2 = "niño".getBytes("UTF-8").length;

int size3 = "niño".getBytes("UTF-16LE").length; //Sin LE mete FFFF indicativo utf16
```

Ejemplo: Tipo char en java

RAMA:	Informática	CICLO:	Desarrollo de aplicaciones multiplataforma		
MÓDULO	Sistemas Informáticos				CURSO: 1
PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	Sept-2020
UNIDAD	COMPETENCIA	Sistemas de Numeración			

Ejercicios finales:

1. Opera:

- a) $(FE_{16} \text{ or } 17_8) \text{ and } 200_{10}$
- b) $100_{10} \text{ XOR } 1A_{16}$

2. Se quiere enviar por una red insegura el mensaje "JAVA 1.7" encriptada byte a byte con el método XOR (recuerda que un carácter ASCII el UTF-8 ocupa un byte). La clave que se usará y la FF_{16} . Indica el código numérico hexadecimal que resulta de la encriptación realizada (es decir, los números que se han enviar por la red).

3. a) Indica la longitud en bytes y la codificación de la palabra Ñandú en UTF-8 y UTF-16

- b) Realiza lo mismo con la letra griega ω y la cirílica Я
- c) Si un usuario envía en ASCII de Europa Occidental la palabra Ñandú ¿Qué recibe un usuario con unicode?

4. Haz las siguientes operación pasando los números previamente a binario (están en decimal). Compruébalo convirtiendo los resultados a decimal.

- a) $128+100$
- b) $90-35$
- c) 25×5
- d) $33/4$