CAMBIOS DE BASE

01

Como es sabido, los sistemas informáticos utilizan, para la representación interna de los datos, instrucciones y de la información que manejan, el sistema binario. Es decir, utilizan, exclusivamente, series de 0 y 1, de mayor o menor longitud según el caso. De manera que, si deseáramos ver el contenido de la memoria de un ordenador, tal y como se representa en el interior de la máquina, obtendríamos una interminable lista de 0 y 1.

Como es evidente, al ser humano se le hace imposible manejar, de forma eficaz y cómoda, tal forma de representación, por lo que es necesario buscar maneras equivalentes y alternativas de visualización. Esta es la razón por la cual, cuando se obtiene información de uso de un ordenador, se realizan sobre ella una serie de operaciones que permitan transformarla a una forma de representación más acorde con nosotros.

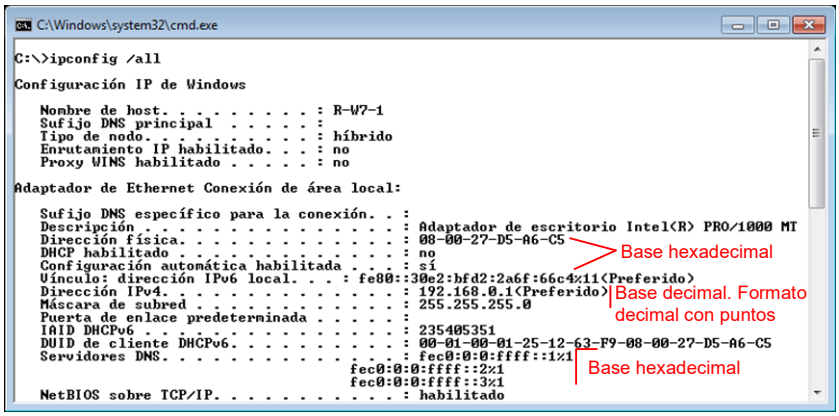


Figura 1

Sin lugar a duda, la operación más básica a realizar, sobre esa forma de representación interna del ordenador, es el cambio de base. De hecho, las utilidades del propio sistema nos facilitan el trabajo y ya nos muestran, en pantalla cuando así lo requerimos, la información interna de la máquina en formatos más convencionales, tal y como se muestra en la Figura 1, en la que puede verse el resultado de la ejecución del comando ***ipconfig /all***, en un sistema *Windows*.

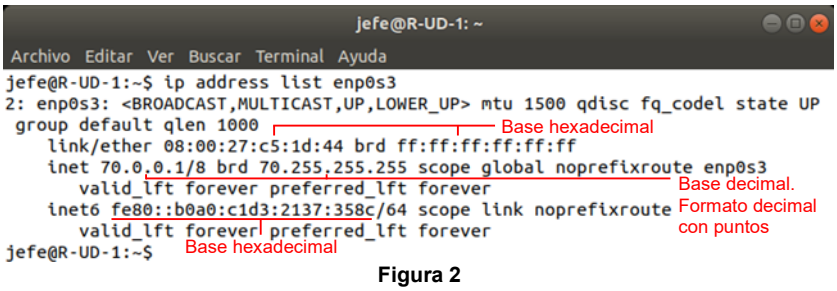


Figura 2

En ella puede comprobarse que se usan distintas formas de representación, pero en ningún caso se usa la forma binaria. Así, por ejemplo, se utiliza la base hexadecimal (base 16) para representar la "Dirección física", de la interfaz de red, (también denominada dirección MAC[1]), quedando como 08-00-27-D5-A6-C5, cuando, sin ningún tipo de transformación, se vería como:

00001000000000000010011111010101101001101100 0101

Caso aparte sería la "Dirección IPv4"[2], que se representa en base diez, en un formato denominado "*decimal con puntos*" (192.168.0.1), equivalente a la representación binaria: 11000000101010000000000000000001. Igual circunstancia se da en la "Máscara de subred"[3], 255.255.255.0, correspondiente a la representación binaria 11111111111111111111111100000000.

Lo mismo ocurre en sistemas *GNU/Linux*, tal y como se muestra en la Figura 2, utilizando el comando ***ip address list enp0s3***, similar al ***ipconfig /all*** de los sistemas *Windows*.

A la vista de lo anterior, es evidente el interés de aprender a realizar los cambios de base necesarios, en cada caso, con el fin de obtener representaciones más manejables.

Esto no es óbice para que, en algunas ocasiones, nos interese trabajar directamente en binario, para tener un absoluto control sobre cuanto hacemos.

Antes de entrar en las operaciones de cambio de base, es conveniente tener claros algunos conceptos, que repasaremos a continuación.

1.- SISTEMAS DE NUMERACIÓN.

Se define, sistema de numeración, como el conjunto de reglas que permiten nombrar y escribir cualquier número, a partir de un número finito de símbolos.

Así, por ejemplo, el sistema decimal permite escribir cualquier número, utilizando el conjunto de símbolos que le es propio, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, mediante las reglas correspondientes.

2.- BASE DE UN SISTEMA DE NUMERACIÓN.

Se denomina base de un sistema de numeración, al cardinal del conjunto de símbolos que utilice, dicho sistema de numeración, para la representación de los números.

Dicho de otra manera, es el número de símbolos (dígitos) que utiliza el sistema de numeración para la representación de los números.

También se define la base de un sistema de numeración, como el número por el que hay que multiplicar a una unidad inferior, para obtener la inmediatamente superior.

Así, por ejemplo, trabajando en el sistema decimal si multiplico 2 unidades por 10 (base del sistema decimal) obtendré 20, que corresponde a 2 decenas (unidad inmediatamente superior a la unidad) y si multiplico el valor obtenido (20) por la base (10), obtendré 2 centenas (200) que es la unidad inmediatamente superior a la decena.

En la Tabla 1 pueden verse las características de los sistemas de numeración más utilizados.

Sistema	Base	Nº. de símbolos	Conjunto de símbolos
Binario	2	2	{0, 1}
Octal	8	8	{0, 1, 2, 3, 4, 5, 6, 7}
Decimal	10	10	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
Hexadecimal	16	16	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Tabla 1

En el caso del sistema binario, a cada uno de sus dígitos, {0, 1}, se les denomina *bit*[4], acrónimo de ***bi**nary digi**t***, dígito binario.

Para indicar en qué sistema de numeración está representado un número, se le añade a su derecha, como subíndice, el valor de la base precedida de un paréntesis. Así, por ejemplo, 77 en base 8 se indicaría como $77_{(8}$ y en base hexadecimal sería $77_{(16}$. En caso de que se omita la indicación de la base, se asumirá que se trata de un número en base 10, aunque puede indicarse como en cualquier otra base. En nuestro ejemplo quedaría como $77_{(10}$.

3.- SISTEMA DE NUMERACIÓN POSICIONAL.

Se denomina sistema de numeración posicional, a aquel sistema de numeración que utiliza un conjunto de símbolos cuyo valor depende de su posición relativa con respecto al separador decimal, que en caso de ausencia se supone, implícitamente, a la derecha de la representación.

Las posiciones en la parte entera de una cifra se numeran de una en una, hacia la izquierda del separador decimal, siendo la adyacente al separador decimal la posición cero.

Se denomina **peso de una posición**, al valor obtenido al elevar la base del sistema de numeración a la posición de que se trate ($base^{(posición)}$), obteniéndose el valor del peso, de la posición correspondiente, en unidades.

Supongamos el número $222_{(10}$. Dado que no figura separador decimal, se asumirá que es un entero y por lo tanto el separador decimal se encontrará ocupando la primera posición por la derecha. En este caso, los tres símbolos (dígitos) son idénticos (dígito 2); sin embargo, analizando la posición que ocupa cada uno de ellos comprobaremos que su valor es distinto, veamos:

	Posición 2 Centenas	**Posición 1 Decenas**	**Posición 0(*) Unidades**
Dígitos	2	2	2
Peso de la posición (unidades) $base^{(posición)}$	10^2	10^1	10^0
Valor de cada dígito (unidades) (Dígito)·(peso de la posición)	$2 \cdot 10^2$	$2 \cdot 10^1$	$2 \cdot 10^0$
Unidades	200	20	2

(*) Recuérdese que todo número elevado a cero es la unidad.

De manera que, según lo visto anteriormente, podemos decir que el número 222: representa 2 centenas + 2 decenas + 2 unidades o lo que es lo mismo 200 unidades + 20 unidades + 2 unidades = 222 unidades.

Con lo cual es claro que el valor de cualquier dígito, en este caso el 2, lo da su posición relativa en el número, con respecto al separador decimal. En nuestro caso el dígito 2 de la izquierda tiene un valor 100 veces superior (10^2, centenas) que el situado más a la derecha (unidades, posición 0).

Obsérvese que lo único que se ha hecho fue aplicar la definición de base de un sistema de numeración vista con anterioridad, según la cual, la base de un sistema de numeración es el número por el que hay que multiplicar a una unidad inferior para obtener la inmediatamente superior. En este caso, dado que trabajamos en base 10, para pasar de unidades a centenas multiplicamos las unidades por 10, y para pasar de decenas a centenas multiplicamos las decenas por 10.

Los sistemas de numeración, comúnmente usados, son posicionales. No lo es, por ejemplo, el sistema de numeración romano, en el cual el 2011 (2 millares + 0 centenas + 1 decena + 1 unidad) se escribiría como MMXI, teniendo los dos símbolos M (que ocupan posiciones distintas) el valor de un millar de unidades, de forma que podría "traducirse" por: 1 millar de unidades + 1 millar de unidades + 10 unidades + 1 unidad.

Las grandes ventajas de los sistemas de numeración posicionales son la sencillez de sus leyes de formación (es muy sencillo aprender a numerar), al tiempo que facilitan enormemente el cálculo, frente a los no posicionales.

Si bien en informática no es corriente la representación directa de números fraccionarios (ya que se utilizan sistemas de representación más adecuados a su manejo por la circuitería digital), es interesante recordar que las posiciones a la derecha del separador decimal se numeran de uno en adelante, hacia la derecha, pero con signo negativo. Veamos un ejemplo con el número $123{,}456_{(10}$.

	Posición 2 Centenas	Posición 1 Decenas	Posición 0 Unidades	Separador decimal	Posición -1 Décimas	Posición -2 Centésimas	Posición -3 Milésimas
Dígitos	1	2	3	,	4	5	6
Peso de la posición (unidades) base(posición)	10^2	10^1	10^0	,	10^{-1}	10^{-2}	10^{-3}
Valor de cada dígito (unidades) (Dígito)·(peso de la posición)	$1\cdot10^2$	$2\cdot10^1$	$3\cdot10^0$	,	$4\cdot10^{-1}$	$5\cdot10^{-2}$	$6\cdot10^{-3}$
Unidades	100	20	3	,	0,4	0,05	0,006

Con lo que obtendríamos: 100 unidades + 20 unidades + 3 unidades + 0,4 unidades + 0,05 unidades + 0,006 unidades.

4.- TEOREMA FUNDAMENTEAL DE LOS SISTEMAS DE NUMERACIÓN POSICIONALES.

El teorema fundamental de la numeración[5], para sistemas posicionales, puede enunciarse como:

Dado un sistema de numeración posicional de base *b*, siendo *b*>1, cualquier número natural *N*, expresado en esa base *b* como $N_{(b} = x_{n-1}\cdots x_2x_1x_0$, puede descomponerse, en su representación polinomial o polinómica, de la forma:

$$N_{(b} = x_{n-1}\cdot b^{n-1} + \cdots + x_2\cdot b^2 + x_1\cdot b^1 + x_0\cdot b^0 = x_{n-1}\cdot b^{n-1} + \cdots + x_2\cdot b^2 + x_1\cdot b^1 + x_0 = N_{(10}$$

Es decir:

$$N_{(10} = \sum_{i=0}^{n-1} x_i \cdot b^i$$

Siendo *n* el número de cifras del número *N* en esa base *b*, donde los coeficientes x_i corresponden a las distintas cifras del número $N_{(b}$ y, como consecuencia, serán menores que la base *b*. Obteniéndose, como resultado de la suma indicada, la representación decimal, $N_{(10}$, del número *N* en base *b,* $N_{(b}$.

Nótese que, en este caso, el teorema se circunscribió a números enteros, por ser los que utilizaremos nosotros, pero es generalizable a números fraccionarios[5].

5.- CONVERSIÓN DE UN SISTEMA DE NUMERACIÓN A OTRO. NÚMEROS ENTEROS.

5.1.- CONVERSIÓN DE UN NÚMERO EN BASE *b* A LA BASE 10.

Para realizar este tipo de conversión simplemente hay que aplicar el teorema fundamental de los sistemas de numeración posicionales. Veamos algunos ejemplos.

a) Convertir el número $10101000_{(2}$ a la base decimal.

$10101000_{(2} = 1\cdot2^7 + 0\cdot2^6 + 1\cdot2^5 + 0\cdot2^4 + 1\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 0\cdot2^0 = 1\cdot128 + 0\cdot64 + 1\cdot32 + 0\cdot16 + 1\cdot8 + 0\cdot4 + 0\cdot2 + 0\cdot1=$

$= 128 + 32 + 8 = 168_{(10}$

b) Convertir el número $2467_{(8}$ a la base decimal.

$2467_{(8} = 2\cdot8^3 + 4\cdot8^2 + 6\cdot8^1 + 7\cdot8^0 = 2\cdot512 + 4\cdot64 + 6\cdot8 + 7\cdot1 = 1024 + 256 + 48 +7 = 1335_{(10}$

c) Convertir el número $D3C9_{(16}$ a la base decimal.

$D3C9_{(16} = D\cdot16^3 + 3\cdot16^2 + C\cdot16^1 + 9\cdot16^0 = D\cdot4096 + 3\cdot256 + C\cdot16 + 9\cdot1 = 13\cdot4096 + 3\cdot256 + 12\cdot16 + 9\cdot1 =$

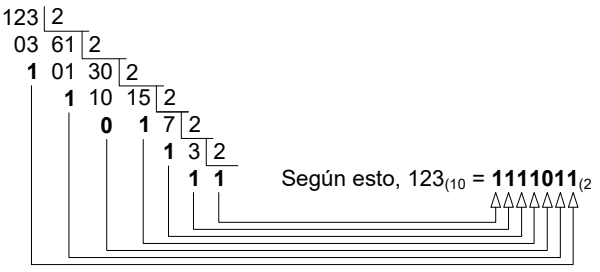
$= 53248 + 768 + 192 + 9 = 54217_{(10}$

5.2.- CONVERSIÓN DE UN NÚMERO EN BASE 10 A LA BASE *b*.

Para realizar esta conversión se utiliza el procedimiento inverso al visto anteriormente, se hacen las divisiones sucesivas del número entre la base *b*, hasta que el cociente sea menor que dicha base. El último cociente y los números obtenidos como restos componen el número en base *b*, pero colocados en orden inverso al que se han ido obteniendo, ocupando el último cociente la posición de mayor peso en la representación del número en la nueva base *b*.

Hagamos algunos ejemplos.

a) Convertir el número $123_{(10}$ a la base binaria.



Para comprobar la veracidad del cambio de base realizado, bastará con acudir al teorema fundamental de los sistemas de numeración posicionales y realizar el cambio opuesto:

$$1111011_{(2} = 1\cdot2^6 + 1\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 64 + 32 + 16 + 8 + 0 + 2 + 1 = 123_{(10}$$

Un buen ejercicio, para practicar el cambio de base de decimal a binario, es convertir las 16 primeras cifras del sistema decimal; ya que nos será muy útil cuando hablemos de las conversiones entre los sistemas binario, octal, decimal y hexadecimal. El resultado de este cambio de base se muestra en la Tabla 2.

Número decimal	Representación binaria	Número decimal	Representación binaria	Número decimal	Representación binaria	Número decimal	Representación binaria
0	0	4	100	8	1000	12	1100
1	1	5	101	9	1001	13	1101
2	10	6	110	10	1010	14	1110
3	11	7	111	11	1011	15	1111

Tabla 2

b) Convertir el número $500_{(10}$ a la base octal.

500 : 8 = 62 → resto **4**
62 : 8 = **7** → resto **6**

Según esto, $500_{(10} = \mathbf{764}_{(8}$

Para verificar el cambio acudiremos al teorema fundamental de los sistemas de numeración posicionales:

$$764_{(8} = 7\cdot8^2 + 6\cdot8^1 + 4\cdot8^0 = 448 + 48 + 4 = 500_{(10}$$

c) Convertir el número $1994_{(10}$ a la base hexadecimal.

1994 : 16 = 124 → resto **10 (A)**
124 : 16 = **7** → resto **12 (C)**

Según esto, $1994_{(10} = \mathbf{7CA}_{(16}$

Cuya verificación nos lleva a:

$$7CA_{(16} = 7\cdot16^2 + C\cdot16^1 + A\cdot16^0 = 7\cdot16^2 + 12\cdot16^1 + 10\cdot16^0 = 1792 + 192 + 10 = 1994_{(10}$$

5.3.- CONVERSIÓN DE UN NÚMERO EN UNA BASE CUALQUIERA *b* A OTRA BASE CUALQUIERA *b'*.

5.3.1.- Método general

Dado que no existe ningún método que permita realizar este cambio de base directamente (de *b* a *b'*, siendo *b* y *b'* dos bases cualesquiera), utilizaremos una técnica que se fundamenta en utilizar, como base auxiliar intermedia la base decimal; es decir, pasaremos el número en base *b* a base 10, y de ésta, a la base *b'*. Utilizando, para estos cambios de base, los métodos ya conocidos.

$$b \rightarrow 10 \rightarrow b'$$

Desarrollemos un ejemplo.

Convertir el número $4572_{(8}$ a la base hexadecimal.

Según el procedimiento descrito anteriormente debemos realizar los cambios de base siguientes:

$$4572_{(8} \rightarrow X_{(10} \rightarrow Y_{(16}$$

a) Cambio de la base octal a la base decimal, utilizando el teorema fundamental de los sistemas de numeración posicionales:

$$4572_{(8} = 4\cdot8^3 + 5\cdot8^2 + 7\cdot8^1 + 2\cdot8^0 = 4\cdot512 + 5\cdot64 + 7\cdot8 + 2\cdot1 = 2048 + 320 + 56 + 2 = 2426_{(10}$$

b) Cambio de la base decimal a la base hexadecimal, utilizando el método de las divisiones sucesivas:

2426 : 16 = 151 → resto **10 (A)**
151 : 16 = **9** → resto **7**

Según esto, $2426_{(10} = \mathbf{97A}_{(16}$

Con lo cual ya sabemos que

$$4572_{(8} = 2426_{(10} = 97A_{(16}$$

Que era lo que se nos pedía.

5.3.2.- Método singular.

En informática, únicamente se utilizan las bases: binaria, octal, decimal y hexadecimal lo que permite utilizar un método singular para realizar los cambios de base entre ellas. Este método utiliza la base binaria como base intermedia, en lugar de la base decimal del método general.

Este sistema singular es más rápido y cómodo que el general, pero solo es aplicable a los cambios entre sistemas de numeración cuyas bases sean potencias de 2, tal y como ocurre entre los sistemas binario (2^1), octal (2^3) y hexadecimal (2^4), mientras que el método general sirve para cambios entre las bases *b* y *b'*, sean cuales quieran esas bases.

Antes de ver este método debemos hablar de la capacidad de representación de un conjunto de bits, es decir, de cuántos símbolos distintos es posible representar con un número determinado de bits. Para entenderlo empezaremos por analizar el caso, práctico, más sencillo. El sistema octal.

5.3.2.1.- Representación binaria de los símbolos del sistema octal.

Como es conocido, el sistema octal utiliza un conjunto de representación de 8 símbolos (de ahí que sea base 8), en concreto utiliza el conjunto: {0, 1, 2, 3, 4, 5, 6, 7}. La cuestión que se plantea es averiguar cuál es el número mínimo de bits (símbolos del sistema binario) que se necesitan para representar, de forma inequívoca, cada uno de los símbolos del sistema octal.

El problema podemos plantearlo en los siguientes términos: Necesito representar 8 símbolos (los dígitos del sistema octal) utilizando para ello los 2 símbolos (bits) del sistema binario ({0, 1}), ¿cuántos símbolos binarios (bits) necesitaré, como mínimo, para poder hacerlo?

Para afrontar la resolución de este problema hay un par de consideraciones que deben tenerse en cuenta:

a) A la hora de realizar la posible codificación, el orden de los bits es determinante; por ejemplo, las codificaciones 010 y 001 serán distintas.

b) Como resulta evidente, los bits 0 y 1 pueden repetirse cuantas veces sean necesarias.

Estas consideraciones nos llevan a deducir que nos encontramos ante un cálculo de variaciones con repetición[6], según esto:

El número de variaciones posibles (V), utilizando un conjunto de n símbolos, tomando un número r de esos símbolos será:

$$V = n^r$$

En nuestro caso, $V = 8$ (símbolos del sistema octal que necesitamos representar), $n = 2$ (número de símbolos del sistema binario) y r es lo que necesitamos averiguar, es decir el número, mínimo, de símbolos binarios necesarios para representar los 8 símbolos del sistema octal. Con lo cual,

$$V = n^r \Rightarrow 8 = 2^r \Rightarrow \log 8 = \log 2^r \Rightarrow \log 8 = r \cdot \log 2 \Rightarrow r = \log 8 / \log 2 = \log 2^3 / \log 2 = 3 \cdot \log 2 / \log 2 = 3 \cdot 1 = 3$$

Según esto, para representar los 8 símbolos del sistema octal, utilizando los 2 símbolos del sistema binario (bits), necesitaremos un mínimo de 3 bits. Resultado absolutamente coherente con el obtenido al pasar las 16 primeras cifras del sistema decimal al binario (Tabla 2), obsérvese que la codificación del 7 decimal, dígito de mayor valor del sistema octal, requiere tres bits para su representación binaria.

Para comprobar que esto es correcto, completaremos la tabla de codificaciones binarias de los símbolos octales utilizando tres bits, Tabla 3.

Símbolo octal	Representación binaria (3 bits)(*)	Símbolo octal	Representación binaria (3 bits)(*)
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

(*) Se obtienen pasando el correspondiente dígito decimal a la base binaria, por el método conocido, y completando la codificación binaria resultante a tres bits, con ceros por la izquierda. Suele denominarse representación normalizada a 3 bits.

Tabla 3

Tal y como vemos, Tabla 3, la solución obtenida es correcta, con tres bits es posible representar los ocho símbolos del sistema octal.

5.3.2.2.- Representación binaria de los símbolos del sistema hexadecimal.

Haciendo un razonamiento idéntico al realizado para el sistema octal, pero sabiendo que en este caso el número de símbolos a representar es de 16 (base hexadecimal), obtenemos que:

$$V = n^r \Rightarrow 16 = 2^r \Rightarrow \log 16 = \log 2^r \Rightarrow \log 16 = r \cdot \log 2 \Rightarrow r = \log 16 / \log 2 = \log 2^4 / \log 2 = 4 \cdot \log 2 / \log 2 = 4 \cdot 1 = 4$$

Según esto, necesitaremos un mínimo de 4 bits (*nibble*[7]) para representar los 16 símbolos del sistema hexadecimal; con lo cual la representación de los símbolos del sistema hexadecimal será la mostrada en la Tabla 4.

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

(*) Se obtienen pasando el correspondiente dígito decimal a la base binaria, por el método conocido, y completando la codificación binaria resultante a cuatro bits, con ceros por la izquierda. Suele denominarse representación normalizada a 4 bits.

Tabla 4

Al igual que en el caso del octal, puede comprobarse la coherencia del resultado obtenido consultando la Tabla 2, correspondiente a la representación binaria de las 16 primeras cifras del sistema decimal.

5.3.2.3.- Procedimiento del método singular.

Vista la representación binaria de los sistemas octal y hexadecimal, ya nos es posible utilizar el método particular de cambio de base entre los sistemas octal y hexadecimal.

Básicamente este sistema consiste en utilizar, como paso intermedio, la representación en base binaria, con tres bits (para el sistema octal) o con 4 bits (para el sistema hexadecimal), de cada uno de los dígitos del número original, y agrupar el conjunto de bits resultante, de cuatro en cuatro o de tres en tres, empezando por la derecha y completando el último grupo con ceros a la izquierda, para obtener la representación en el sistema hexadecimal u octal, según interese, de cada uno de los dígitos.

Desarrollaremos algún ejemplo.

a) Convertir el número $765_{(8}$ a la base hexadecimal.

En la tabla siguiente se recoge el procedimiento utilizado para realizar este cambio de base.

Número en base octal	$765_{(8}$		
Dígitos octales del número	7	6	5
Representación binaria (con tres bits) de cada dígito octal	111	110	101
Número en base binaria	$111110101_{(2}$		
Grupos de cuatro bits, empezando por la derecha y rellenando con ceros a la izquierda hasta completar el último grupo, si fuera necesario	0001	1111	0101
Símbolos hexadecimales correspondientes a cada grupo de 4 bits	1	F	5
Número en base hexadecimal	$1F5_{(16}$		

Para comprobar la bondad de este procedimiento, bastará con realizar la conversión mediante el método general visto y comprobar que los resultados son idénticos, tal y como se muestra a continuación:

1. Cambio de la base octal a la base decimal, utilizando el teorema fundamental de los sistemas de numeración posicionales:

$$765_{(8} = 7\cdot8^2 + 6\cdot8^1 + 5\cdot8^0 = 7\cdot64 + 6\cdot8 + 5\cdot1 = 448 + 48 + 5 = 501_{(10}$$

2. Cambio de la base decimal a la base hexadecimal, utilizando el método de las divisiones sucesivas:

   501 : 16 = 31 → resto **5**

   31 : 16 = **1** → resto **15 (F)**

   Según esto, $501_{(10}$ = **1F5**$_{(16}$

Con lo cual ya sabemos que

$$765_{(8} = 501_{(10} = 1F5_{(16}$$

b) Convertir el número $F1B_{(16}$ a la base octal.

Se utiliza el mismo procedimiento visto en el ejemplo anterior, pero representando cada dígito hexadecimal por su correspondiente codificación de 4 bits y, posteriormente, agrupando el conjunto de bits obtenido de tres en tres, empezando por la derecha y completando a ceros por la izquierda, para concluir representando el dígito octal correspondiente a cada uno de los grupos de tres bits obtenidos. Este procedimiento se muestra en la tabla siguiente.

Número en base hexadecimal	$F1B_{(16}$		
Dígitos hexadecimales del número	F	1	B
Representación binaria (con cuatro bits) de cada dígito hexadecimal	1111	0001	1011
Número en base binaria	$111100011011_{(2}$		
Grupos de tres bits, empezando por la derecha y rellenando con ceros a la izquierda hasta completar el último grupo, si fuera necesario	111	100	011	011
Símbolos octales correspondientes a cada grupo de 3 bits	7	4	3	3
Número en base octal	$7433_{(8}$			

Utilizando el método general, deberemos llegar a idéntico resultado:

1. Cambio de la base hexadecimal a la base decimal, utilizando el teorema fundamental de los sistemas de numeración posicionales:

$$F1B_{(16} = F\cdot16^2 + 1\cdot16^1 + B\cdot16^0 = 15\cdot256 + 1\cdot16 + 11\cdot1 = 3840 + 16 + 11 = 3867_{(10}$$

2. Cambio de la base decimal a la base octal, utilizando el método de las divisiones sucesivas:

   3867 : 8 = 483 → resto **3**

   483 : 8 = 60 → resto **3**

   60 : 8 = **7** → resto **4**

   Según esto, $3867_{(10}$ = **7433**$_{(8}$

Con lo cual ya sabemos que

$$F1B_{(16} = 3867_{(10} = 7433_{(8}$$

Se propone, como ejercicio adicional, el comprobar la codificación binaria de ambos ejemplos; partiendo de las codificaciones decimales y utilizando el método de las divisiones sucesivas.

De igual manera puede verificarse el resultado del ejemplo desarrollado en el apartado 5.3.1, utilizando este método singular.

5.4.- CONVERSIÓN RÁPIDA DE LA BASE BINARIA A LA BASE DECIMAL.

Conocido el teorema fundamental de los sistemas de numeración posicionales y, quizá, como resultado de la práctica, es posible desarrollar un sistema más rápido y cómodo para la conversión de la base binaria a la base decimal.

En este procedimiento, lo único que es necesario recordar es el peso de cada posición en un número binario, lo cual es muy sencillo pues siempre, obviamente, es el doble que la unidad inmediatamente inferior. Veamos una tabla con algunos de estos pesos.

Posición.....	....	10	9	8	7	6	5	4	3	2	1	0
Peso de cada posición.....	....	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valor del peso.....	....	1024	512	256	128	64	32	16	8	4	2	1

Según esto, para convertir cualquier número en base binaria a su equivalente en base decimal, tan solo debemos sumar los valores de los pesos de las posiciones ocupadas por bits 1, en la representación binaria del número.

Veamos un ejemplo. Convertir el número $10111001110_{(2}$ a su equivalente en base decimal.

Para ello debemos darnos cuenta de que los bits 1 ocupan las posiciones 1, 2, 3, 6, 7, 8 y 10 con lo cual la representación decimal de ese número binario, debe ser la suma de los pesos de esas posiciones:

$$10111001110_{(2} = 2^1 + 2^2 + 2^3 + 2^6 + 2^7 + 2^8 + 2^{10} = 2 + 4 + 8 + 64 + 128 + 256 + 1024 = 1486_{(10}$$

6.- EJERCICIOS PROPUESTOS:

1.- Convertir los números $11101_{(2}$, $11101_{(8}$ y $11101_{(16}$ a sus correspondientes números equivalentes en base decimal.

2.- Convertir el número $101010111_{(2}$ a sus números equivalentes en base octal, decimal y hexadecimal.

3.- Convertir el número $73624_{(8}$ a sus números equivalentes en base binaria, decimal y hexadecimal.

4.- Convertir el número $AB713_{(16}$ a sus números equivalentes en base binario, octal y decimal.

5.- En las representaciones de las direcciones IP, y de las máscaras de subred, se suele utilizar el formato decimal con puntos (tal y como se muestra en la Figura 1 y la Figura 2), que se obtiene agrupando los bits, de la representación binaria, de 8 en 8 (a un conjunto de 8 bits se le denomina byte[8] u octeto) y representando cada uno de los grupos por su correspondiente valor decimal, separados por el carácter punto.

Dado que las direcciones IP y las máscaras de subred (hablando de la versión 4 del protocolo IP[2] (Protocolo de Internet, ***I**nternet **P**rotocol*) se componen de 32 bits (4 bytes); lo normal es indicar que el formato decimal con puntos se obtiene representado cada byte de la dirección IP, o de la máscara de subred, por su correspondiente codificación decimal, separados por un punto.

Según esto, cuál sería la representación, en formato decimal con puntos, de la IP 10101001111111100110000101001110 y de la máscara de subred 11111111111111110000000000000000.

6.- Utilizando una herramienta específica, se ha determinado que la dirección física (dirección MAC[1] o dirección *hardware*), compuesta de 48 bits o 6 bytes (en sistemas IPv4), asociada a la interfaz de red de un equipo es la: 000010000000000000100111011011110111110001001000. Cuál será la representación de la dirección MAC indicada, usando el formato habitual; que representa los *nibbles* (se denomina *nibble* a un conjunto de 4 bits) de cada byte en hexadecimal, separando los dos *nibbles*, que componen cada byte de la dirección, por un guion (-), Figura 1, o por un carácter dos puntos (:), tal y como se muestra en la Figura 2.

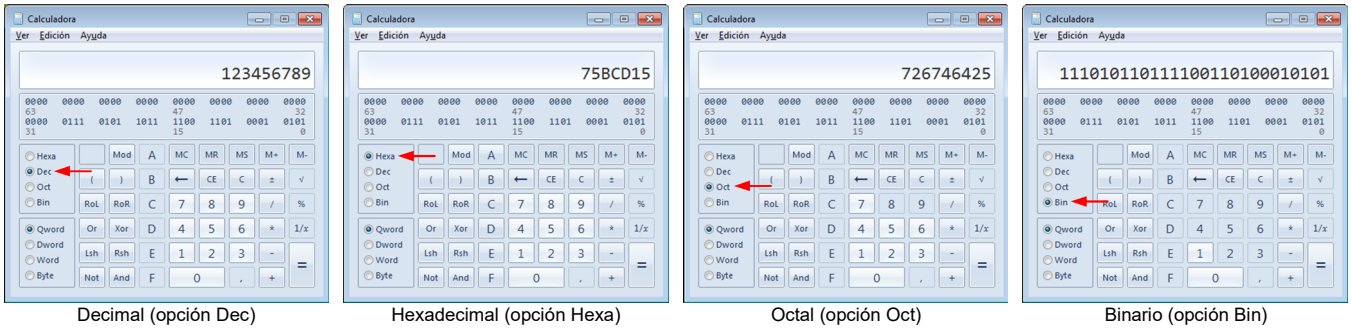
7.- ANEXO-I. USO DE LAS CALCULADORAS.

Todos los sistemas operativos, de entorno gráfico, incorporan, entre las aplicaciones que traen por defecto, calculadoras más o menos sofisticadas, que facilitan enormemente la tarea de realizar los cambios de base de las distintas informaciones que se manejan. Veremos aquí el caso de los sistemas *Microsoft Windows* y el *Ubuntu Desktop 20.04 LTS*.

7.1.- Sistemas *Microsoft Windows*.

En estos sistemas, por defecto, se tiene acceso a la calculadora estándar, que no incorpora herramientas avanzadas para la realización de cambios de base.

Para poder acceder a estas funciones es necesario trabajar con la denominada calculadora de programador, accesible a través de la opción “Ver” de la barra de menús de la propia calculadora, una vez desplegadas las opciones correspondientes se seleccionará la opción “Programador” que nos presentará la calculadora mostrada en la Figura 3.



Decimal (opción Dec) Hexadecimal (opción Hexa) Octal (opción Oct) Binario (opción Bin)

Figura 3

Esta calculadora, por defecto, trabaja en base decimal, pero bastará con seleccionar la opción deseada en el bloque de las bases de numeración, tal y como se muestra en la Figura 3, para obtener la representación, de la cifra en pantalla, en la nueva base de numeración. Adviértase que, independientemente de la base numérica seleccionada, bajo la pantalla se muestra la codificación binaria del número en pantalla. Nótese, que se incluye la numeración correspondiente a las posiciones de los bits de la representación, del 0 al 63.

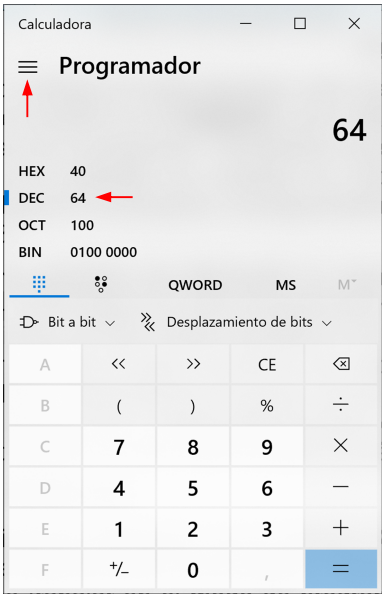
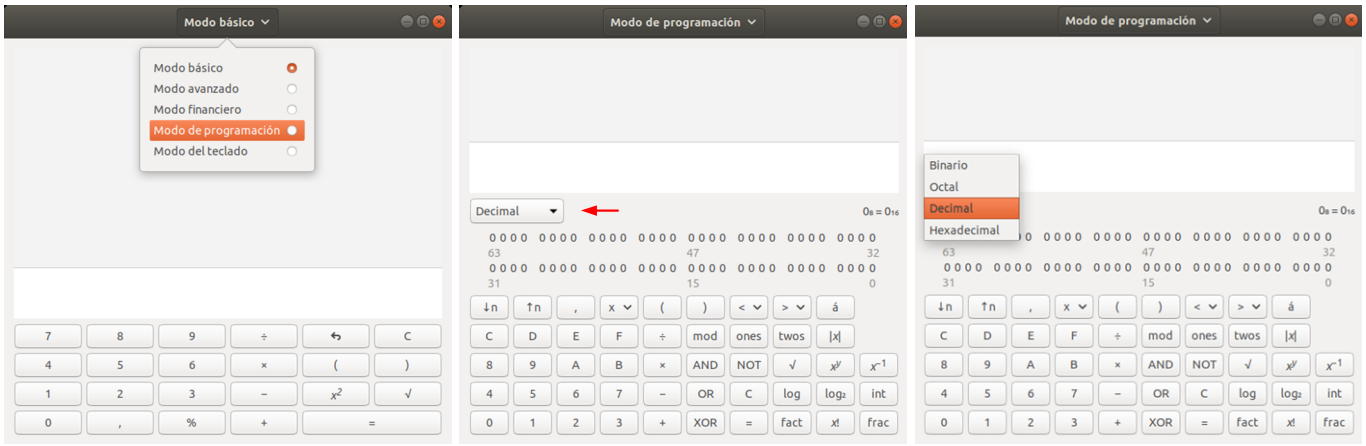


Figura 4

En la versión 10, del sistema operativo de *Microsoft*, la calculadora cambia de aspecto, tal y como se muestra en Figura 4. Para llegar al modo programador, mostrado en la Figura 4, bastará con acudir a su menú, en las líneas situadas a la izquierda del modo de trabajo, y seleccionar el deseado. Para elegir la base numérica de trabajo se hará clic sobre la que se requiera (HEX, DEC, OCT y BIN), apareciendo la opción activa con una marca de color en su lado izquierdo, en la Figura 4 sería la base decimal. Adviértase que, independientemente de la base de trabajo, se muestra la codificación, del valor en pantalla, en las cuatro bases numéricas.

7.2.- Sistemas *Ubuntu Desktop 20.04 LTS*.

Al igual que ocurría en los sistemas *Windows*, en el *Ubuntu Desktop 20.04 LTS* se accede a la calculadora, por defecto, en el denominado modo básico. Para poder acceder a las herramientas de cambio de base debemos trabajar en el modo de programación, para lo cual abriremos el combo del modo de trabajo, de la barra de menús de la calculadora, Figura 5a, y seleccionamos el modo de programación, que nos presentará la calculadora mostrada en la Figura 5b que, al igual que ocurría en los sistemas *Windows*, trabaja, por defecto, en base decimal.

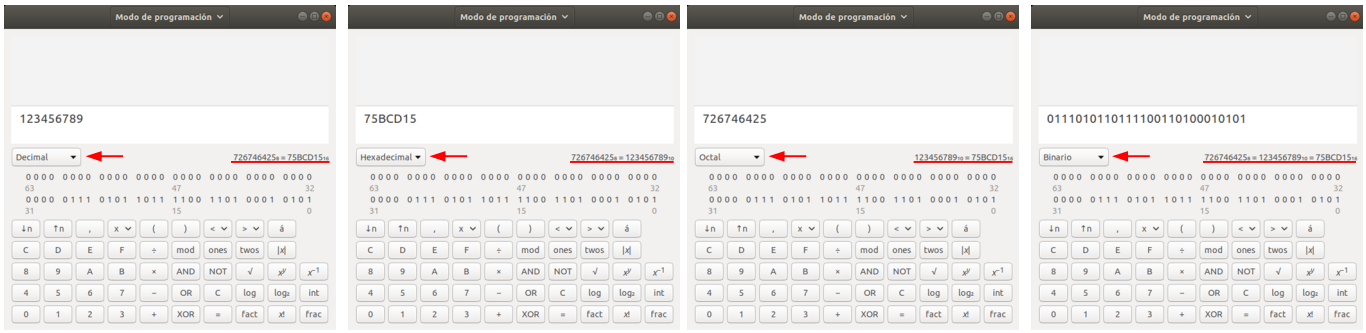


a b c

Figura 5

Para indicar la base del número que se va a introducir, se selecciona la opción correspondiente en el combo, Figura 5c, y se introduce el número cuya codificación se desea conocer en otras bases. Bajo la pantalla, en la parte derecha, se mostrarán las

nuevas codificaciones octal, decimal o hexadecimal, según proceda, del número de la pantalla. Por debajo de estas codificaciones podrá verse la representación binaria con 64 bits (posiciones 0 a 63). Tal y como se muestra en los ejemplos de la Figura 6.



Número decimal **Número hexadecimal** **Número octal** **Número binario**

Figura 6

7.3.- Dispositivos móviles.

En la actualidad, suele resultar muy útil disponer de una calculadora, con las funciones de cambio de base, en el dispositivo móvil de uso habitual, ya que nos evita el tener que llevar una calculadora de mano.

Existen variedad de modelos, y versiones, desarrolladas para todos los sistemas operativos utilizados en los dispositivos móviles actuales.

En la Figura 7 se muestra un ejemplo de calculadora, con las funciones indicadas (DEC, HEX, OCT y BIN), sobre un dispositivo con sistema operativo *Android*.

Figura 7

8.- ANEXO-II. UNIDADES HABITUALES.

- A un conjunto de 4 bits se le denomina ***nibble***[7] o **cuarteto**, que corresponde a la representación, en binario, de un dígito hexadecimal.
- A un conjunto de 8 bits se le denomina **byte**[8] u **octeto**, un byte está formado por dos *nibbles*, el superior (los 4 bits de la izquierda) y el inferior (los cuatro bits de menor peso), Figura 8. Así, por ejemplo, la dirección física (o MAC) 08-00-27-D5-A6-C5, de la Figura 1, estará compuesta por 6 octetos, separados por guiones (es muy habitual utilizar como separador de los octetos el carácter dos puntos, Figura 2), cada uno de los cuales se representa mediante los dos caracteres hexadecimales que codifican cada uno de sus *nibbles*.

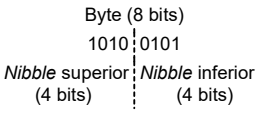


Figura 8

- En sistemas de almacenamiento de información se utilizan las siguientes unidades, múltiplos del byte (la letra B, representa al byte y la b al bit):

8 bits (8 b)	1024 B	1024 kB	1024 MB	1024 GB	1024 TB
1 Byte (1 B)	1 kilobyte (1 kB)	1 Megabyte (1 MB)	1 Gigabyte (1 GB)	1 Terabyte (1 TB)	1 Petabyte (1 PB)

Obsérvese que se utilizan potencias de dos para la relación entre las unidades. Así por, ejemplo, 1 kB son 2^{10} bytes (1024 bytes) por ser la potencia de dos que más se próxima al valor 1000. Esto hace que con el uso de estas unidades surja un problema de ambigüedad, ya que al utilizar los mismos símbolos y prefijos que el Sistema Internacional de Unidades[9] (SI), debieran utilizarse las correspondientes potencias de 10, para la conversión entre las unidades, de manera que 1 kB debería corresponder a 10^3 B, es decir 1000 bytes, y no 1024.

Para evitar este equívoco, la Comisión Electrotécnica Internacional (IEC[10], ***International Electrotechnical Commission***), propuso nuevos prefijos para las unidades que usan potencias de dos, la propuesta contempla el uso de los mismos prefijos que en el SI, añadiéndole el término "*binario*", abreviado como "***bi***". Así, por ejemplo, el gigabyte pasaría a denominarse *gi**bi**byte* y se representaría como ***GiB***[11].

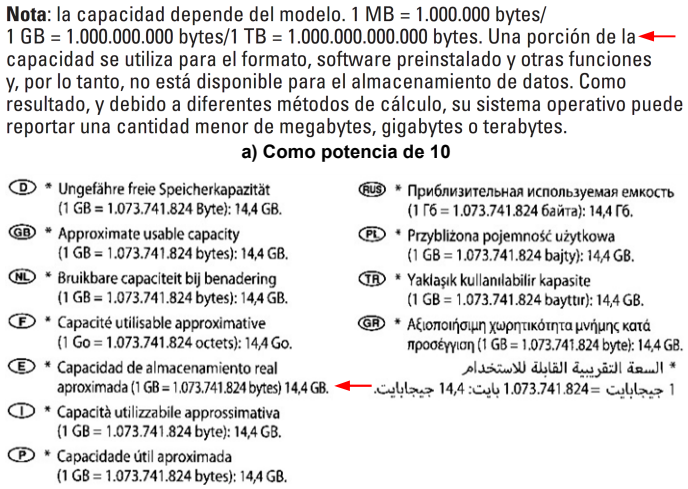


Figura 9

En la práctica, la ambigüedad es grande ya que cada vez se utilizan más las potencias de 10, en lugar de las potencias de 2, de manera que no siempre es sencillo saber si se refieren a, por ejemplo, kB decimales (1000 B) o kB binarios (1024 B). Tanto es así, que, en muchos casos, sobre todo si se utilizan potencias de 10, el propio fabricante del equipo indica que relación se utiliza en las unidades, tal y como se muestra en la Figura 9a. Aunque lo aconsejable, para evitar cualquier tipo de ambigüedad, es que, aun utilizando la potencia de dos, se indique de forma inequívoca que unidad se usa, tal y como se muestra en la Figura 9b.

Un buen ejemplo del uso de ambas unidades se muestra en la Figura 10, en donde puede verse como la cantidad de memoria RAM se expresa en GiB, mientras que la capacidad del disco duro se indica en GB.

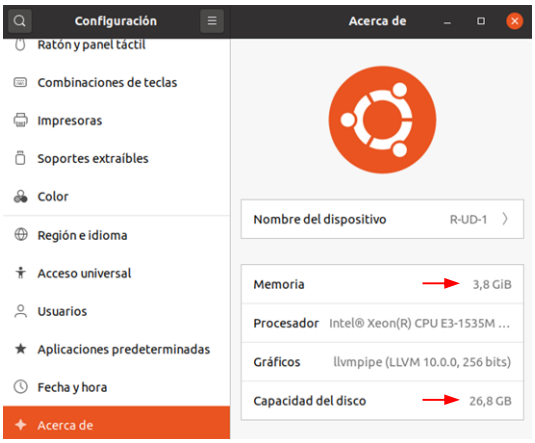


Figura 10

- En sistemas de transferencia de información, transmisiones, suelen utilizarse, como unidades, los siguientes múltiplos del bit:

1 bit	1000 bits	1000 kb	1000 Mb	1000 Gb	1000 Tb	
1 b	1 kilobit (1 kb)	1 Megabit (1 Mb)	1 Gigabit (1 Gb)	1 Terabit (1 Tb)	1 Petabit (1 Pb)	

En estos sistemas, estas unidades suelen referirse a la unidad de tiempo (segundo, s) dando como resultado:

1 bit/s	1000 bits/s	1000 kb/s	1000 Mb/s	1000 Gb/s	1000 Tb/s
1 bit/s (1 b/s; 1 bps)	1 kilobit/s (1 kb/s; 1 kbps)	1 Megabit/s (1 Mb/s; 1 Mbps)	1 Gigabit/s (1 Gb/s; 1 Gbps)	1 Terabit/s (1 Tb/s; 1 Tbps)	1 Petabit/s (1 Pb/s; 1 Pbps)

Indicando que en un segundo se transfieren, o transmiten, las cantidades de bits señaladas en cada caso. Así, por ejemplo, cuando se habla de una velocidad de transmisión de 1 Gbs^{-1}, se indica que se trasmiten 1.000.000.000 bits por segundo, lo que implica que para la transmisión de cada bit se emplearán 0,000000001 segundos, es decir $1 \cdot 10^{-9}$ s. Lo que indica que se transmite un bit por cada nanosegundo (1 nanosegundo = $1 \cdot 10^{-9}$ s). Todo ello, siempre y cuando la transmisión se realice en serie[12], típicamente, en su momento, utilizando el estándar TIA/EIA-854[13].

Adviértase que en todos los casos se utiliza la letra *B* para referirse al byte y la *b* para hacer referencia al *bit*. Así, por ejemplo, cabría indicar que 8 Mb/s equivaldrían a 1 MB/s.

La expresión de las unidades en la forma, por ejemplo, Mbps no es correcta, aunque en la práctica se utilice en entornos no técnicos. Las formas correctas, en este ejemplo, serían Mb/s o Mbs^{-1}.

Obsérvese que en el caso de las unidades de transferencia de información se utiliza, exclusivamente, el SI, es decir unidades múltiplos de 10.

9.- EJERCICIOS PROPUESTOS RESUELTOS.

1.- Convertir los números $11101_{(2}$, $11101_{(8}$ y $11101_{(16}$ a sus correspondientes números equivalentes en base decimal.

$11101_{(2}$ Para realizar esta conversión acudiremos al teorema fundamental de los sistemas de numeración posicionales, según el cual:

$$11101_{(2} = 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = 16 + 8 + 4 + 0 + 1 = 29_{(10}$$

$11101_{(8}$ También para realizar esta conversión acudiremos al teorema fundamental de los sistemas de numeración posicionales, según el cual:

$$11101_{(8} = 1\cdot8^4 + 1\cdot8^3 + 1\cdot8^2 + 0\cdot8^1 + 1\cdot8^0 = 1\cdot4096 + 1\cdot512 + 1\cdot64 + 0\cdot8 + 1\cdot1 = 4096 + 512 + 64 + 0 + 1 =$$
$$= 4673_{(10}$$

$11101_{(16}$ De nuevo haremos uso del teorema fundamental de los sistemas de numeración posicionales.

$$11101_{(16} = 1\cdot16^4 + 1\cdot16^3 + 1\cdot16^2 + 0\cdot16^1 + 1\cdot16^0 = 1\cdot65536 + 1\cdot4096 + 1\cdot256 + 0\cdot16 + 1\cdot1 =$$
$$= 65536 + 4096 + 256 + 0 + 1 = 69889_{(10}$$

En este problema es muy importante reflexionar sobre la absoluta necesidad de tener siempre la certeza de la base de numeración sobre la que estamos trabajando. Normalmente, las herramientas que dan información del sistema la transforman a alguna de las bases comúnmente utilizadas, pero suelen asumir que el usuario sabe exactamente lo que obtiene, de manera que es muy fácil que, en una misma pantalla de información, se utilicen distintas bases de numeración sin ningún tipo de indicación. Esto puede comprobarse en la Figura 1 y la Figura 2, en donde se obtiene información en base hexadecimal y decimal sin especificar tal circunstancia.

Con este ejercicio se ve, bien a las claras, que un error a la hora de atribuir la base de numeración, a una información, genera resultados absolutamente distintos. En este caso, el sistema nos mostraría la cifra 11101, sin indicación de su base, y nosotros deberíamos saber, dependiendo de la información a la que corresponda, en qué base nos la está dando. Si nos equivocamos, podemos, por ejemplo, acceder a una posición de memoria miles de veces por encima de la deseada.

2.- Convertir el número $101010111_{(2}$ a sus números equivalentes en base octal, decimal y hexadecimal.

Conversión a decimal: Los bits a 1, ocupan las posiciones 0, 1, 2, 4, 6 y 8, con lo cual:

$$101010111_{(2} = 2^0 + 2^1 + 2^2 + 2^4 + 2^6 + 2^8 = 1 + 2 + 4 + 16 + 64 + 256 = 343_{(10}$$

Una vez obtenida la codificación decimal, podremos acudir al procedimiento general para obtener las representaciones octal y hexadecimal, de las siguientes formas:

Conversión a octal	Conversión a hexadecimal
343 : 8 = 42 → resto **7**	343 : 16 = 21 → resto **7**
42 : 8 = **5** → resto **2**	21 : 16 = **1** → resto **5**
Según esto, $343_{(10} = 527_{(8}$	Según esto, $343_{(10} = 157_{(16}$

Dado que se trata de una conversión entre los sistemas: binario, octal y hexadecimal, también podemos aplicar el método específico para estas conversiones, con lo cual:

	Conversión a octal			..	Conversión a hexadecimal		
Representación binaria	$101010111_{(2}$			..	$101010111_{(2}$		
Grupos de tres bits (octal) o 4 bits (hexadecimal).	101	010	111	..	0001	0101	0111
Dígitos representados por cada grupo de bits	5	2	7	..	1	5	7
Representación octal y hexadecimal.....	$527_{(8}$			..	$157_{(16}$		

Recuérdese que los bits se agrupan empezando por la derecha y completando con ceros a la izquierda, en su caso, el último de los grupos obtenido.

3.- Convertir el número $73624_{(8}$ a sus números equivalentes en base binaria, decimal y hexadecimal.

Conversión a Decimal: Haciendo uso del teorema fundamental de los sistemas de numeración posicionales, tenemos:

$$73624_{(8} = 7\cdot8^4 + 3\cdot8^3 + 6\cdot8^2 + 2\cdot8^1 + 4\cdot8^0 = 7\cdot4096 + 3\cdot512 + 6\cdot64 + 2\cdot8 + 4\cdot1 = 28672 + 1536 + 384 + 16 + 4 = 30612_{(10}$$

Una vez obtenida la codificación decimal, podremos acudir al procedimiento general para obtener las representaciones binaria y hexadecimal, de las siguientes formas:

Conversión a binario	Conversión a hexadecimal
30612 : 2 = 15306 → resto **0**	30612 : 16 = 1913 → resto **4**
15306 : 2 = 7653 → resto **0**	1913 : 16 = 119 → resto **9**
7653 : 2 = 3826 → resto **1**	119 : 16 = **7** → resto **7**
3826 : 2 = 1913 → resto **0**	Según esto, $30612_{(10} = 7794_{(16}$
1913 : 2 = 956 → resto **1**	

956 : 2 = 478 → resto **0**
478 : 2 = 239 → resto **0**
239 : 2 = 119 → resto **1**
119 : 2 = 59 → resto **1**
59 : 2 = 29 → resto **1**
29 : 2 = 14 → resto **1**
14 : 2 = 7 → resto **0**
7 : 2 = 3 → resto **1**
3 : 2 = **1** → resto **1**
Según esto, $30612_{(10} = 111011110010100_{(2}$

Dado que se trata de una conversión entre los sistemas: octal, binario y hexadecimal, también podemos aplicar el método específico para estas conversiones, con lo cual:

Número en base octal	$73624_{(8}$				
Dígitos octales del número	7	3	6	2	4
Representación binaria (con tres bits) de cada dígito octal	111	011	110	010	100
Número en base binaria	$111011110010100_{(2}$				
Grupos de cuatro bits, empezando por la derecha y rellenando con ceros a la izquierda hasta completar el último grupo, si fuera necesario	0111	0111	1001	0100	
Símbolos hexadecimales correspondientes a cada grupo de 4 bits	7	7	9	4	
Número en base hexadecimal	$7794_{(16}$				

4.- Convertir el número $AB713_{(16}$ a sus números equivalentes en base binaria, octal y decimal.

Conversión a decimal: Haciendo uso del teorema fundamental de los sistemas de numeración posicionales, tenemos:

$$AB713_{(16} = A\cdot16^4 + B\cdot16^3 + 7\cdot16^2 + 1\cdot16^1 + 3\cdot16^0 = 10\cdot16^4 + 11\cdot16^3 + 7\cdot16^2 + 1\cdot16^1 + 3\cdot16^0 =$$

$$= 10\cdot65536 + 11\cdot4096 + 7\cdot 256 + 1\cdot16 + 3\cdot1 = 655360 + 45056 + 1792 + 16 + 3 = 702227_{(10}$$

Una vez obtenida la codificación decimal, podremos acudir al procedimiento general para obtener las representaciones binaria y octal, de las siguientes formas:

Conversión a binario

702227 : 2 = 351113 → resto **1**
351113 : 2 = 175556 → resto **1**
175556 : 2 = 87778 → resto **0**
87778 : 2 = 43889 → resto **0**
43889 : 2 = 21944 → resto **1**
21944 : 2 = 10972 → resto **0**
10972 : 2 = 5486 → resto **0**
5486 : 2 = 2743 → resto **0**
2743 : 2 = 1371 → resto **1**
1371 : 2 = 685 → resto **1**
685 : 2 = 342 → resto **1**
342 : 2 = 171 → resto **0**
171 : 2 = 85 → resto **1**
85 : 2 = 42 → resto **1**
42 : 2 = 21 → resto **0**
21 : 2 = 10 → resto **1**
10 : 2 = 5 → resto **0**
5 : 2 = 2 → resto **1**
2 : 2 = **1** → resto **0**
Según esto, $702227_{(10} = 10101011011100010011_{(2}$

Conversión a octal

702227 : 8 = 87778 → resto **3**
87778 : 8 = 10972 → resto **2**
10972 : 8 = 1371 → resto **4**
1371 : 8 = 171 → resto **3**
171 : 8 = 21 → resto **3**
21 : 8 = **2** → resto **5**
Según esto, $702227_{(10} = 2533423_{(8}$

Utilizando el procedimiento singular para las conversiones entre las bases: binaria, octal y hexadecimal, se obtiene:

Número en base hexadecimal	$AB713_{(16}$				
Dígitos hexadecimales del número	A	B	7	1	3
Representación binaria (con cuatro bits) de cada dígito hexadecimal	1010	1011	0111	0001	0011
Número en base binaria	$10101011011100010011_{(2}$				

Grupos de tres bits, empezando por la derecha y rellenando con ceros a la izquierda hasta completar el último grupo, si fuera necesario	010	101	011	011	100	010	011
Símbolos octales correspondientes a cada grupo de 3 bits	2	5	3	3	4	2	3
Número en base octal	$2533423_{(8}$						

5.- En las representaciones de las direcciones IP, y de las máscaras de subred, se suele utilizar el formato decimal con puntos (tal y como se muestra en la Figura 1 y la Figura 2), que se obtiene agrupando los bits, de la representación binaria, de 8 en 8 (a un conjunto de 8 bits se le denomina byte u octeto) y representado cada uno de los grupos por su correspondiente valor decimal, separados por el carácter punto.

Dado que las direcciones IP y las máscaras de subred (hablando de la versión 4 del protocolo IP (Protocolo de Internet, *Internet Protocol*) se componen de 32 bits (4 bytes); lo normal es indicar que el formato decimal con puntos se obtiene representado cada byte de la dirección IP, o de la máscara de subred, por su correspondiente codificación decimal, separados por un punto.

Según esto, cuál sería la representación, en formato decimal con puntos, de la IP 10101001111111100110000101001110 y de la máscara de subred 11111111111111110000000000000000.

Empezaremos por la IP:

IP en base binaria	$10101001111111100110000101001110_{(2}$			
Grupos de ocho bits (bytes)	10101001	11111110	01100001	01001110
Codificación decimal de cada uno de los bytes (Aplicando el teorema fundamental a cada byte)	169	254	97	78
IP en formato decimal con puntos	169.254.97.78			

Haciendo lo mismo con la máscara de subred se obtiene:

Máscara de subred en base binaria	$11111111111111110000000000000000_{(2}$			
Grupos de ocho bits (bytes)	11111111	11111111	00000000	00000000
Codificación decimal de cada uno de los bytes (Aplicando el teorema fundamental a cada byte)	255	255	0	0
Máscara de subred en formato decimal con puntos	255.255.0.0			

6.- Utilizando una herramienta específica, se ha determinado que la dirección física (dirección MAC o dirección hardware), compuesta de 48 bits o 6 bytes (en sistemas IPv4), asociada a la interfaz de red de un equipo es la: 000010000000000000100111011011110111110001001000. Cuál será la representación de la dirección MAC indicada, usando el formato habitual; que representa los *nibbles* (se denomina *nibble* a un conjunto de 4 bits) de cada byte en hexadecimal, separando los dos *nibbles*, que componen cada byte de la dirección, por un guion (-), Figura 1, o por un carácter dos puntos (:), tal y como se muestra en la Figura 2.

Empezaremos por separar cada uno de los seis bytes que componen la dirección MAC, utilizando para ello, por ejemplo, el carácter "**-**":

00001000-00000000-00100111-01101111-01111100-01001000

A continuación, marcaremos los dos *nibbles* (4 bits) que componen cada byte:

00001000-**0000**0000-**0010**0111-**0110**1111-**0111**1100-**0100**1000

Para terminar, representaremos cada uno de los *nibbles* por su correspondiente codificación hexadecimal:

08-00-27-6F-7C-48 o bien 08:00:27:6F:7C:48, utilizando como separador de los bytes el carácter ":".

10.- NOTAS FINALES.

[1] Siglas de ***M**edia **A**ccess **C**ontrol*, control de acceso al medio. Véase: https://es.wikipedia.org/wiki/Direcci%C3%B3n_MAC

[2] Siglas de ***I**nternet **P**rotocol **v**ersion **4***, Protocolo de Internet versión 4. Véase: https://es.wikipedia.org/wiki/IPv4

[3] Para máscara de subred véase: https://support.microsoft.com/es-es/help/164015/understanding-tcp-ip-addressing-and-subnetting-basics

[4] Para bit véase: https://es.wikipedia.org/wiki/Bit

[5] Para el teorema fundamental de los sistemas de numeración posicionales véase: https://es.wikipedia.org/wiki/Sistema_de_numeraci%C3%B3n#:~:text=Teorema%20fundamental%20de%20la%20numeraci%C3%B3n,-Este%20teorema%20establece&text=El%20valor%20total%20del%20n%C3%BAmero,la%20ejecuci%C3%B3n%20de%20operaciones%20aritm%C3%A9ticas.

[6] Para variaciones con repetición véase: https://es.wikipedia.org/wiki/Variaci%C3%B3n_(combinatoria)

[7] Para *nibble* véase: https://es.wikipedia.org/wiki/Nibble

[8] Para byte véase: https://es.wikipedia.org/wiki/Byte

[9] Para Sistema Internacional de Unidades véase: https://es.wikipedia.org/wiki/Sistema_Internacional_de_Unidades

[10] Para Comisión Electrotécnica Internacional véase: https://www.iec.ch/

[11] Para GiB véase: https://es.wikipedia.org/wiki/Gibibyte

[12] Para transmisión serie véase: https://es.wikipedia.org/wiki/Comunicaci%C3%B3n_serie

[13] Para el estándar TIA/EIA-854 (***T**elecommunications **I**ndustry **A**ssociation*, Asociación de la Industria de Telecomunicaciones, y ***E**lectronic **I**ndustries **A**lliance*, Alianza de Industrias Electrónicas), sobre cables de pares trenzados categoría 6, véase: https://www.flukenetworks.com/knowledge-base/application-or-standards-articles-copper/1000base-tx-over-category-6