Práctica 2

1. Realizar un esquema de transformación para conseguir los siguientes cambios de base: de binario a decimal, octal y hexadecimal y viceversa.

- Para convertir un número **binario a uno decimal**, habrá que asignarle un valor a cada cifra del número binario. De derecha a izquierda, se le asignarán las potencias de 2 (partiendo de 2^0 de manera ascendente) hasta que no queden más cifras en el número binario. Después, se multiplicará cada cifra binaria con su correspondiente potencia y se sumarán todos los resultados. El número que resulte, será el número decimal de dicho número binario. Por ejemplo, el número $10110011_{(2}$ tiene 8 cifras (1 0 1 1 0 0 1 1), por lo que le asignaremos tantas potencias de 2 como cifras tenga, de derecha a izquierda de manera ascendente. Lo colocamos en forma de tabla y procedemos a multiplicar las cifras con su correspondiente potencia:

1	0	1	1	0	0	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64*0=0**	**32*1=32**	**16*1=16**	**8*0=0**	**4*0=0**	**2*1=2**	**1*1=1**

Por último, procedemos a sumar todos los resultados y, si hemos efectuado las operaciones de manera correcta, resultará el número decimal buscado:

$1 + 2 + 0 + 0 + 16 + 32 + 0 + 128 = 179_{(10}$

- **Viceversa**: Para convertir un número **decimal a uno binario**, aplicaremos el proceso contrario, pero tanteando. Es decir, se buscará en las potencias de 2 qué número es el que más se aproxima al número decimal que queremos transformar. Podrán aparecer dos números, uno menor y uno mayor, pero se escogerá el número menor, ya que las cifras faltantes se irán formando con el resto de potencias. Se hará la misma secuencia hasta que nuestro número se haya visto reducido a 0. Cuando el número se haya visto reducido a 0,

todas las potencias siguientes tendrían un 0 asignado. Por ejemplo, el número 97 se aproxima tanto a al número 64 como al número 128, pero cogeremos aquel que es menor, el 64. El número 64 sabemos que es 2^6, por lo que haremos otra tabla con todas las potencias de 2^6 (la máxima) a 2^0 (la mínima) en la primera fila y, en la segunda, pondremos 1 o 0 en función de si nos pasamos o no del número a convertir (97):

$64 = 2^6$	32	16	8	4	2	$1 = 2^0$
1	**1**	**0**	**0**	**0**	**0**	**1**

El número 64 es la mayor potencia que se corresponde con el número a convertir (97), por lo que se le asignará el primer 1. Hay que proceder a restarle dicha potencia al número decimal para poder continuar con las siguientes cifras (97 – 64 = 33), por lo que ahora necesitaremos una potencia de 2 menor a 64 (2^6) que no se pase del número que nos ha resultado, el 33. Como el número 32 puede cubrir dicha posición sin pasarse, le asignaremos un 1, y restaremos nuestro resultado con esta potencia (33 – 32 = 1). El número 16 no lo podemos poner ya que nos pasamos del número 1, así que se le asignará un 0. Esto mismo ocurre con los números 8, 4 y 2, por lo que también se asignaremos el 0 a cada une de ellos. El 1 (2^0) puede cubrir a este 1 resultante, así que pondremos un 1 y efectuamos la resta (1 – 1 = 0). Como ya ha resultado 0, juntaríamos todas estas cifras de 1 y 0 y ya habríamos acabado nuestra conversión: $1100001_{(2}$.

- Para convertir un número **binario a uno octal**, juntaremos el número binario en cifras de tres, de derecha a izquierda, ya que sabemos que 2^3 es 8 (el número de valores en base octal, del 0 al 7), una potencia de la base que estamos tratando. De derecha a izquierda nuevamente, iremos convirtiendo cada grupo de tres cifras a decimal, que será el mismo número en octal, ya que lo máximo que caben en tres cifras binarias es 7 en ambas bases. Una vez convertido cada grupo, juntaremos todas las cifras resultantes. En caso de que no podamos formar un grupo de tres porque no hay más números, se

añadirán tantos ceros por la izquierda sean necesarios para que sean tres cifras. Por ejemplo, tenemos el número binario 10111001 que queremos convertir a base octal. Para ello, procedemos a formar grupos de tres cifras: 10 111 001. Como resultan solo dos cifras a la izquierda, le añadimos un cero a la izquierda para formar tres: 010 111 001. Ahora hay que convertir dichos grupos en cifras en base decimal (que serán igual en base octal):

010 = 2 111 = 7 001 = 1

Resultan las cifras 2, 7 y 1, por lo que las juntamos y nuestro número buscado es el $271_{(8}$.

- **Viceversa**: Para convertir un número **octal a uno binario**, aplicaremos el proceso contrario, es decir, que a cada cifra en base octal le asignaremos su valor en base binaria, asignándole a cada una siempre tres cifras. Por ejemplo, el número $731_{(8}$ lo vamos a descomponer de manera binaria, ya que es muy sencillo convertir los números del 0 al 7 a dicha base, asignándole tres cifras binarias a cada cifra octal:

7 = 111 3 = 011 1 = 001

Juntamos todas nuestras cifras binarias y resulta nuestro número binario: $111011001_{(2}$.

- Para convertir un número **binario a uno hexadecimal**, seguiremos la misma mecánica que de binario a octal, solo que en vez de juntar cifras de tres de derecha a izquierda, juntaremos cuatro, ya que sabemos que 2^4 es 16 (el número de valores en base hexadecimal, del 0 a la F [$15_{(10}$]). De derecha a izquierda nuevamente, iremos convirtiendo cada grupo de cuatro cifras a decimal, teniendo en cuenta que A, B, C, D, E y F se corresponden con 10, 11, 12, 13, 14 y 15, respectivamente. Una vez convertido cada grupo, juntaremos todas los cifras (números y/o letras) resultantes. En caso de que no podamos formar un grupo de cuatro porque no hay más números, se añadirán tantos ceros por la izquierda sean necesarios para que sean cuatro cifras. Por ejemplo, tenemos

el número binario 1010111001 que queremos convertir a base hexadecimal. Para ello, procedemos a formar grupos de cuatro cifras: 10 1011 1001. Como resultan solo dos cifras a la izquierda, le añadimos dos ceros a la izquierda para formar cuatro: 0010 1011 1001. Ahora hay que convertir dichos grupos en cifras en base hexadecimal:

0010 = 2 1011 = B 1001 = 9

Resultan las cifras 2, B (recordemos que se corresponde con $11_{(10}$) y 9, por lo que las juntamos y nuestro número buscado es el $2B9_{(16}$.

- **Viceversa**: Para convertir un número **hexadecimal a uno binario**, aplicaremos el proceso contrario, es decir, que a cada cifra en base hexadecimal le asignaremos su valor en base binaria, asignándole a cada una siempre cuatro cifras. Por ejemplo, el número $C1A_{(16}$ lo vamos a descomponer de manera binaria, asignándole cuatro cifras binarias a cada cifra hexadecimal:

C = 1100 1 = 0001 A = 1010

Juntamos todas nuestras cifras binarias y resulta nuestro número binario: $110000011010_{(2}$.

2. Realizar los siguientes cambios a binario.

a) $1030_{(10} \rightarrow 10000000110_{(2}$
b) $7301_{(8} \rightarrow 111\ 011\ 000\ 001_{(2}$
c) $FE0_{(16} \rightarrow 1111\ 1110\ 0000_{(2}$

3. Transformar el número binario 1111100000 en:

a) Decimal → $32+64+128+256+512 = 992_{(10}$
b) Octal → $1740_{(8}$
c) Hexadecimal → $3E0_{(16}$

4. Transformar los números hexadecimales 45A0, CF y 3020 en:

Hexadecimal	**Decimal**	**Binario**	**Octal**
45A0	17824	1000101101000000	42640
CF	207	11001111	317
3020	12320	1100000010000 0	30040

5. Transformar los números octales 457, 53, y 302 en:

Octal	**Decimal**	**Binario**	**Hexadecimal**
457	303	100101111	12F
53	43	101011	2B
302	194	11000010	C2

6. Con 8 bits:

a) ¿Cuántos números distintos se pueden escribir con 8 bits?
   - Con 8 bits se podrán escribir 256 números distintos (del 0 [$00000000_{(2}$] al 255 [$11111111_{(2}$]).

b) Escribir el número más pequeño que se puede escribir con 8 bits y convertirlo a decimal.
   - El número más pequeño que se puede escribir con 8 bits es $00000000_{(2}$, el cual también es 0 en base decimal.

c) Escribir el número más grande que se puede escribir con 8 bits y convertirlo a decimal.
   - 255 es el número más grande que se podrá escribir con 8 bits, ya que 8 bits es igual a $11111111_{(2}$, lo que equivale a 255 en base decimal.

7. En qué consiste el código ASCII extendido y Unicode.

- El código ASCII extendido es un estándar de codificación de caracteres que consiste en la representación de texto de 255 caracteres distintos gracias a la combinación de 8 bits.
  - Tabla de caracteres ASCII: http://www.asciitable.com/

- El código Unicode es un estándar de codificación de caracteres que contiene a su vez los estándares de UTF-8, UTF-16 y UTF-32, entre otros. Estos estándares pueden representar muchos más caracteres que ASCII extendido gracias a que usan un byte para la menor codificación y para la mayor, cuatro bytes (en el caso de UTF-8, para el resto esta cifra va en aumento a medida que hablamos de un UTF mayor), mientras que ASCII extendido solo usa un bit. En la actualidad Unicode (versión 11.0) logra soportar más de 137 000 caracteres, pudiendo en el futuro seguir ampliando este número gracias a sus combinaciones tan bien planificadas. Añadir también que Unicode puede entender ASCII extendido, ya que parte de esos 255 caracteres, con el mismo valor binario.
  - Tabla de caracteres Unicode (UTF-8): https://www.utf8-chartable.de/unicode-utf8-table.pl

8. Realizar una tabla de correspondencia como la siguiente entre los sistemas
decimal, binario, octal y hexadecimal.

Deci-mal	**Bina-rio**	**Octal**	**Hexa-deci-mal**	**Deci-mal**	**Bina-rio**	**Octal**	**Hexa-deci-mal**
0	00000	0	0	16	10000	20	10
1	00001	1	1	17	10001	21	11
2	00010	2	2	18	10010	22	12
3	00011	3	3	19	10011	23	13
4	00100	4	4	20	10100	24	14
5	00101	5	5	21	10101	25	15
6	00110	6	6	22	10110	26	16
7	00111	7	7	23	10111	27	17
8	01000	10	8	24	11000	30	18
9	01001	11	9	25	11001	31	19
10	01010	12	A	26	11010	32	1A
11	01011	13	B	27	11011	33	1B
12	01100	14	C	28	11100	34	1C
13	01101	15	D	29	11101	35	1D
14	01110	16	E	30	11110	36	1E
15	01111	17	F	31	11111	37	1F