# Guia de trabajos prácticos N° 1

## Sistemas de numeración, operaciones lógicas y unidades de medida de la información

1. Completar la siguiente tabla realizando las conversiones necesarias entre los sistemas numéricos: decimal, binario, octal y hexadecimal.

| Decimal | Binario | Octal | Hexadecimal |
|---|---|---|---|
| 25 | | | |
| 50 | | | |
| 100 | | | |
| 54783 | | | |
| | 0111 0100 1010 0011 | | |
| | | 3052 | |
| | | | 206A |
| | _ _ _ _ 0010 _ _ _ _ 1011 | | 4 _ 5 _ |
| | _ _ _ _ 0010 _ _ _ _ _ _ _ _ | _ _ _ 377 | 6 _ _ _ |

2. Realizar las siguientes sumas en binario.

| | | | | | |
|---|---|---|---|---|---|
| 00001011 | 00001011 | 11110100 | 11001011 | 01010011 | 10001000 |
| + 00001100 | + 11111101 | + 11100110 | + 00110101 | + 01100010 | + 11110110 |

3. Tomando en cuenta las operaciones del ejercicio anterior.
   a. Considerando que todos los números son binarios con signo complemento a 2, convierta los operandos y resultados a decimal.
   b. Verifique que los resultados coinciden e identifique los casos de error.
   c. Elabore una regla para detectar los casos de error.

4. Convertir los siguientes números binarios a números negativos en complemento a 2. Elija una cantidad apropiada de bits única que soporte todos los casos.

| Decimal | Binario (en valor absoluto) | Complemento a 2 |
|---|---|---|
| -462 | | |
| -13677 | | |
| -4095 | | |
| -4096 | | |

5. Realizar las siguientes operaciones lógicas **and** (**&**), **or** (**|**) y **xor** (**^**) entre números hexadecimales:

a. FFFF **&** AAAA =
b. 25A5 **&** 007F =
c. 25A5 **&** FF80 =
d. 2FE4 **&** 2FE4 =
e. FFFF **|** AAAA =
f. 25A5 **|** 007F =
g. 25A5 **|** FF80 =
h. 87AC **|** 87AC =
i. FFFF **^** AAAA =
j. 25A5 **^** 007F =
k. 25A5 **^** FF80 =
l. 25A5 **^** 25A5 =

6. Resuelva los siguientes desplazamientos, **shr** (>>) y **shl** (<<), con propagación de signo:
   a. $0001\ 1101\ 1010\ 1100_2$ >> 4 =
   b. $0001\ 1101\ 1010\ 1100_2$ << 4 =
   c. $1001\ 1101\ 1010\ 1100_2$ >> 2 =
   d. $1001\ 1101\ 1010\ 1100_2$ << 2 =
   e. $24AF_{16}$ >> 8 =
   f. $24AF_{16}$ << 16 =
   g. $247_8$ >> 7 =
   h. $247_8$ << 3 =
   i. $1_{10}$ << 1 =
   j. $1_{10}$ << 2 =
   k. $1_{10}$ << 3 =
   l. $1_{10}$ << 4 =
   m. $8_{10}$ >> 1 =

7. Aplique las siguientes máscaras usando la operación **and** (&) para completar la tabla.

| **n** | n **&** 0F00 | n **&** 000F | n **&** 0008 | n **&** 0004 | n **&** 0002 | n **&** 0001 |
|---|---|---|---|---|---|---|
| 1A6E | | | | | | |
| 3DE3 | | | | | | |
| 9BCB | | | | | | |
| DAD0 | | | | | | |

8. Aplique las siguientes máscaras usando la operación **or** (|) para completar la tabla.

| **n** | n **|** 0F00 | n **|** 000F | n **|** 0008 | n **|** 0004 | n **|** 0002 | n **|** 0001 |
|---|---|---|---|---|---|---|
| 1A6E | | | | | | |
| 3DE3 | | | | | | |
| 9BCB | | | | | | |
| DAD0 | | | | | | |

9. Dado un número ***n*** de 16 bits, combine máscaras y desplazamientos necesarios para obtener:
   a. El byte de la izquierda.
   b. El byte de la derecha.
   c. Devolver 1 si el número es impar, y 0 si es par.
   d. Devolver -1 si el número es negativo, y 0 si es positivo.
   e. Devolver el número que representan los primeros 12 bits
   f. Devolver el número que representan los últimos 4 bits

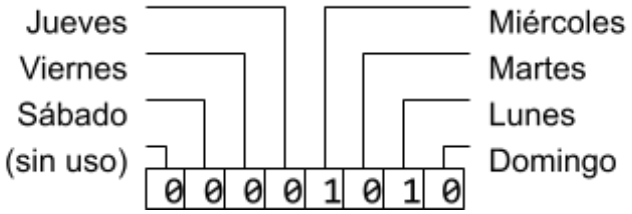
   Por ejemplo si n = 02A3; entonces a) 0002, b) 00A3, c) 0001, d) 0000, e) 002A, f) 0003

10. Realice un programa en C para verificar los resultados del ejercicio anterior.

11. Realizar una función en C, que reciba como parámetro un número entero (`int nro`) y un string (`char* s`), y rellene el string con los caracteres '1' o '0' con la representación binaria del número entero. Debe realizar la función utilizando máscaras y desplazamientos.
    a. Verifique el resultado comparado con su representación hexadecimal
    `printf("%X = %s", nro, s);`

12. Realizar un programa en C para verificar que la PC utiliza complemento a 2 para la representación de números negativos.

13. Realizar una función en C que recibe un byte (`char`), donde cada bit representa un día de la semana (ver imágen), e imprima la descripción los días activos.




    Si el bit tiene un 1 quiere decir que el día está activo, en caso contrario tiene un 0.
    Por ejemplo: si ingresa 10 = `0x0A` = `0b00001010` entonces escribe "Lunes" y "Miércoles".

14. Realizar una función "**`weekday_set`**" en C que reciba como parámetros: un puntero a char y un número entero (de 0 a 6), y que configure en 1 el bit correspondiente al día de la semana (siendo 0 domingo y 6 sábado) como en el ejercicio anterior. Del mismo modo, escribir una función "**`weekday_reset`**", con los mismos parámetros, para configurar el bit en 0.

15. Realizar una función en C en la cual ingresa un valor entero de 2 bytes (`short int`), donde se codifica una fecha, e imprima la misma en formato ISO 8601 (`YYYY-MM-DD`).
    Los 2 bytes (16 bits) se utilizan del siguiente modo para codificar la fecha: los 5 bits más significativos para el día (de 1 a 31), seguidos de 4 bits para el mes (de 1 a 12) y los 7 bits menos significativos para el año (de 0 a 99). Si el año es mayor a 50, se asume que está entre 1950 y 1999, si es menor a o igual a 50, se considera como del 2000 al 2050.

16. Realizar una función en C que reciba día, mes y año como parámetros enteros y devuelva un short int con la fecha codificada al igual que en el ejercicio anterior.

17. Realizar una función en C para pasar un string ASCII a mayúsculas, utilizando máscaras y aprovechando la codificación ASCII (no utilizar la función de librerías como ***toupper()*** o ***strupr()***).

18. Realizar una función en C para pasar un string ASCII a un integer, utilizando máscaras (no utilizar funciones de librerías como ***atoi()***).

19. Realizar un programa (`op.exe`) en C que reciba 3 argumentos: número de operación (obligatorio), valor `A` (obligatorio) y valor `B` (opcional) e imprima el resultado de la operación: `0 => A + B; 1 => A & B; 2 => A; 3=> ~A`. El programa no debe tener sentencia **`if`** ni **`switch`** para elegir la función, sino que debe utilizar punteros a funciones.

    Ejemplos cómo se debería ver por consola al ejecutar el programa con los parámetros:

```
C:\...\tp01\>op 0 12 9
21

C:\...\tp01\>op 1 12 9
8

C:\...\tp01\>op 2 12
12

C:\...\tp01\>op 3
error falta parámetro A

C:\...\tp01\>op 10
error operación inválida
```

20. Realizar programa en C para cifrar/descifrar un archivo `TXT` con una palabra clave pasada por parámetro (NOTA: el archivo debe abrirse como binario). El programa debe recibir por parámetros la palabra clave, el archivo origen y el archivo destino. El cifrado debe hacerse byte a byte con cada byte del archivo y la palabra utilizando **`xor`** (^) (la palabra se utiliza cíclicamente).
    Probar el programa creando un archivo de texto y pasarlo dos veces por el programa utilizando la misma palabra clave.

21. ¿Cuántos bloques de memoria de 4 kiB entran en una memoria de 4GiB?

22. Una variable de 4 bytes que representa un puntero a memoria:
    a. ¿Cuántas celdas de memoria puede direccionar?
    b. Si cada celda es de 2 bytes, ¿cuál es el tamaño de la memoria direccionable por esa variable?

23. Con una velocidad de transferencias de 100 GigaBits/seg cuanto tiempo se tardaría en llenar un disco de 1 TiB? (asumiendo que es la tasa efectiva de una transmisión constante y sin interrupciones utilizando las unidades en decimal del sistema internacional)
    a. ¿Y si el disco fuese de 100GiB?

24. Para hacer el streaming de una película de alta resolución que dura 2 horas y ocupa 4 GiB, ¿cuál es, en teoría, la tasa mínima de transferencia efectiva que debo tener de ancho de banda para reproducirla por completo sin ningún corte?