



75.562 · Fundamentos de Computadores · 2024-25

PEC1 - Primera prueba de evaluación continua

Apellidos: *López Henestrosa*

Nombre: José Carlos

Formato y fecha de entrega

- Para dudas y aclaraciones sobre el enunciado debéis dirigiros al consultor responsable de vuestra aula.
- Hay que entregar la solución en un fichero PDF utilizando una de las plantillas entregadas conjuntamente con este enunciado.
- Se debe entregar a través de la aplicación de **Entrega de la Actividad** correspondiente del apartado **Contenidos** de vuestra aula.
- La fecha límite de entrega es el **5 de marzo** (a las 24 horas).
- **Razonad la respuesta en todos los ejercicios. Las respuestas sin justificación no recibirán puntuación.**

Respuestas

Ejercicio 1 [15 %]

Dada la secuencia de bits 10101010, indicad a qué número decimal equivale según cada una de las interpretaciones siguientes:

- a) [5 %] Si se trata de un número binario natural.
(Sección 1.3. Cambios de base)

Para pasar de base 2 (binario) a base 10 (decimal), usamos el método basado en el TFN.

$$\begin{aligned} 10101010_{(2)} &= 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\ &= 1 \cdot 128 + 0 \cdot 64 + 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 170_{(10)} \end{aligned}$$

10101010₍₂₎ en base 10 es 170₍₁₀₎.



- b) [5 %] Si se trata de un número binario con 5 bits para la parte entera y 3 bits para la parte fraccionaria, sin signo.
(Sección 1.3. Cambios de base)

El número binario propuesto en el enunciado con 5 bits para la parte entera y 3 bits para la parte fraccionaria equivale a $10101,010_{(2)}$.

Para realizar la conversión a base 10, podemos seguir el mismo procedimiento aplicado en el apartado anterior, a través del cambio de base basado en el TFN.

$$\begin{aligned} 10101,010_{(2)} &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} \\ &= 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 + 0 \cdot 0,5 + 1 \cdot 0,25 + 0 \cdot 0,125 = 21,25_{(10)} \end{aligned}$$

$10101,010_{(2)}$ en base 10 es $21,25_{(10)}$.

- c) [5 %] De este último formato (5 bits para la parte entera y 3 para la parte fraccionaria, sin signo), ¿qué precisión proporciona este formato?
(Sección 2.1.2. Precisión)

En base 2, el número $10101,010_{(2)}$ tiene una precisión de 3 cifras decimales (0,001), ya que es la distancia entre dos valores consecutivos representables en este formato.

Por otro lado, la precisión en base 10 está determinada por la parte fraccionaria del número binario. En este caso, la parte fraccionaria $0,010_{(2)}$ tiene 3 bits de precisión. Por lo tanto, la menor diferencia representable en esta parte fraccionaria es $2^{-3} = 0,125$, pero el último bit significativo es $2^{-2} = 0,25$.

En base 10, esto implica que la precisión del número es de dos decimales exactos (0,25), lo que indica que el número se expresa con una **precisión de 2 cifras decimales** (0,01).



Ejercicio 2 [30 %]

- a) [10 %] Representad el número $307_{(10)}$ en Complemento a 2. ¿Cuál es la cantidad mínima de bits que hacen falta para representarlo?
(Sección 1.3. Cambios de base y
Sección 2.3.3. Representación en complemento a 2)

Para representar $307_{(10)}$ en complemento a 2, primero tenemos que determinar el rango de valores. Sabemos que, en general, el rango de enteros representables con n bits en Ca2 es, en decimal:

$$[-2^{n-1}, 2^{n-1} - 1]$$

Partiendo de este principio, debemos encontrar el mínimo valor de n tal que:

$$307 \leq 2^{n-1} - 1$$

Al probar con varios valores de n , encontramos que para $n = 10$ se cumple:

$$307 \leq 2^{10-1} - 1 \Rightarrow 307 \leq 511$$

Por lo tanto, podemos concluir que la cantidad mínima de bits que hacen falta para representar $307_{(10)}$ en complemento a 2 es **10 bits**.

- b) [10 %] Representad el número $-307_{(10)}$ en Complemento a 2. ¿Cuál es la cantidad mínima de bits que hacen falta para representarlo?
(Sección 2.3.4. Cambio de signo en Complemento a 2)

El proceso es similar al del apartado anterior. Para representar $-307_{(10)}$ en complemento a 2, primero tenemos que determinar el rango de valores. Sabemos que, en general, el rango de enteros representables con n bits en Ca2 es, en decimal:

$$[-2^{n-1}, 2^{n-1} - 1]$$

Partiendo de este principio, debemos encontrar el mínimo valor de n tal que:

$$-2^{n-1} \leq -307$$

Al probar con varios valores de n , encontramos que para $n = 10$ se cumple:


$$-2^{10-1} \leq -307 \Rightarrow -512 \leq -307$$

Por lo tanto, podemos concluir que la cantidad mínima de bits que hacen falta para representar $-307_{(10)}$ en complemento a 2 es **10 bits**.



- c) [5 %] Representad el número $-307_{(10)}$ en Signo y magnitud. ¿Cuál es la cantidad mínima de bits que hacen falta para representarlo?
(Sección 2.3.1. Representación de enteros en signo y magnitud en base 2)

Para representar $-307_{(10)}$ en signo y magnitud, primero tenemos que convertir el número de base 10 (sin el signo) a base 2. En este caso, realizaré la conversión mediante el método de la división entera por 2 para obtener el valor binario:

DIVISIÓN	COCIENTE	RESTO	
$307 \div 2$	153	1	 <p>Leemos los restos de abajo hacia arriba para obtener el número en base 2</p>
$153 \div 2$	76	1	
$76 \div 2$	38	0	
$38 \div 2$	19	0	
$19 \div 2$	9	1	
$9 \div 2$	4	1	
$4 \div 2$	2	0	
$2 \div 2$	1	0	
$1 \div 2$	0	1	


$307_{(10)}$ en base 2 es $100110011_{(2)}$.

Una vez tenemos el número en base 2, agregamos el bit más significativo (MSB), el cual indica el signo. Como estamos ante un número negativo, tenemos que agregar 1 como bit de signo al número en base 2, por lo que el resultado es: $1100110011_{(SM2)}$.



- d) [5 %] Representad el número $307_{(10)}$ en hexadecimal.
(Sección 1.3.3. Cambio de base entre b y b^n)

Primero pasamos 132 a base 2 usando el método de la división entera por 2:

DIVISIÓN	COCIENTE	RESTO	 Leemos los restos de abajo hacia arriba para obtener el número en base 2
$307 \div 2$	153	1	
$153 \div 2$	76	1	
$76 \div 2$	38	0	
$38 \div 2$	19	0	
$19 \div 2$	9	1	
$9 \div 2$	4	1	
$4 \div 2$	2	0	
$2 \div 2$	1	0	
$1 \div 2$	0	1	

$307_{(10)}$ en base 2 es $100110011_{(2)}$.

Ahora, para pasar de base 2 a base 16, aplicamos el método de cambio de base b (2) a base b^n (2^4). Así, tenemos que agrupar los dígitos binarios de 4 en 4 y de derecha a izquierda para después transformar cada grupo de 4 bits en un dígito hexadecimal. En consecuencia:

$$307_{(10)} = 0001\ 0011\ 0011_{(2)} = 133_{(16)}$$



Ejercicio 3 [30 %]

Haced la suma de los números binarios 110110 y 011001 de 6 bits, considerando los siguientes escenarios en el formato de representación dado. En cada caso, indicad si se produce desbordamiento.

- a) [10 %] Son números enteros en Complemento a 2. Hemos preguntado a una herramienta de IA generativa que nos resuelva el ejercicio y nos da la siguiente solución:

(Sección 2.3.6. Suma en complemento a 2 y
Sección 2.3.7. Resta en complemento a 2

*Vamos a realizar la suma de los números binarios **110110** y **011001** de **6 bits** en el contexto de los enteros representados en **Complemento a 2**.*

Paso 1: Convertir los números binarios a decimal

Número 1: 110110 (en Complemento a 2)

En complemento a 2, si el primer bit (más significativo) es 1, significa que el número es negativo. Primero, encontramos su valor en complemento a 2.

- 110110: El primer bit es 1, por lo que el número es negativo.
- Tomamos el complemento a 2: Invertimos los bits y sumamos 1:
 - 110110 → 001001 (invertir bits)
 - 001001 + 1 = 001010
- El número en decimal es 10, por lo que 110110 representa -10 en decimal.

Número 2: 011001 (en Complemento a 2)

El primer bit es 0, lo que significa que el número es positivo. Directamente lo convertimos a decimal:

- $011001 \rightarrow 1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0$
 $= 32 + 16 + 2 = 50.$

Paso 2: Realizar la suma en decimal

Ahora que tenemos los números en decimal, sumamos: $-10 + 50 = 40$

Paso 3: Convertir el resultado de vuelta a binario

El resultado es 40 en decimal. Convertimos este número a binario:

$40 = 101000_2$

Paso 4: Verificar el desbordamiento

En una representación de 6 bits en complemento a 2, los valores posibles van de -32 a 31 (ya que el rango en complemento a 2 con 6 bits es de $-2^{(n-1)}$ a $2^{(n-1)} - 1$).



El resultado de 40 está fuera de este rango, por lo que se produce un desbordamiento.

Razonad si la solución propuesta es correcta.

- Si es correcta, realizad la operación con un desarrollo vuestro para obtener el mismo resultado.
- Si es incorrecta, indicad el paso a partir del cual es incorrecta y obtened el resultado correcto.

La solución propuesta por la IA generativa es errónea a partir del paso 1, concretamente cuando indica que $011001_{(Ca2)}$ en base 10 es 50. La forma correcta de convertir $011001_{(Ca2)}$ a base 10 es la siguiente:

$$011001_{(Ca2)} = 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 16 + 8 + 1 = 25$$

En base a la anterior corrección, vamos a retomar la estructura propuesta por la IA para continuar la resolución del ejercicio.

Paso 2: Realizar la suma en decimal

Ahora que tenemos los números en decimal, sumamos: $-10 + 25 = 15$

Paso 3: Convertir el resultado de vuelta a binario

El resultado es 15 en decimal. Convertimos este número a binario:

$$15_{(10)} = 001111_{(Ca2)}$$

Paso 4: Verificar el desbordamiento

En una representación de 6 bits en complemento a 2, los valores posibles van de -32 a 31 (ya que el rango en complemento a 2 con 6 bits es de $-2^{(n-1)}$ a $2^{(n-1)} - 1$).

$15_{(10)}$ está dentro de este rango, por lo que **no se produce un desbordamiento.**

b) [10 %] Son números enteros en signo y magnitud.



(Sección 2.3.2. Suma y resta en signo y magnitud)

Sabemos que, en signo y magnitud, el bit más significativo (MSB) indica el signo y que los otros bits representan la magnitud absoluta del número.

Dados los números $110110_{(SM2)}$ y $011011_{(SM2)}$, sabemos lo siguiente:

- $110110_{(SM2)}$
 - **MSB:** 1 (signo negativo)
 - **Magnitud:** 10110
- $011011_{(SM2)}$
 - **MSB:** 0 (signo positivo)
 - **Magnitud:** 11011

Como se trata de un número positivo y otro negativo, hay que analizar las magnitudes para saber cuál es la mayor, restar la magnitud pequeña de la grande y aplicar al resultado el signo de la magnitud mayor.

En este caso, restamos las magnitudes en este orden: $11011_{(2)} - 10110_{(2)}$, ya que $11011 > 10110$ (en base decimal, $27 > 22$).

$$\begin{array}{r}
 11011 \text{ (1)} \\
 1 \text{ (2)} \\
 - 10110 \text{ (3)} \\
 \hline
 00101 \text{ (4)}
 \end{array}$$

Donde:

- **(1):** Magnitud de $011011_{(SM2)}$.
- **(2):** Acarreo.
- **(3):** Magnitud de $110110_{(SM2)}$.
- **(4):** Resultado de la suma.

Como la magnitud mayor proviene del número positivo ($011011_{(SM2)}$), el resultado también será positivo.

Ahora que tenemos la magnitud y el signo, desglosamos el resultado para determinar el número entero binario en signo y magnitud.

- **MSB:** 0 (el resultado es positivo)
- **Magnitud:** 00101 (resultado de la operación)

La suma de $110110_{(SM2)}$ y $011011_{(SM2)}$ en signo y magnitud es $000101_{(SM2)}$.



Por último, convertimos el número a decimal para determinar si se produce desbordamiento:

$$000101_{(SM2)} = 0 \cdot 2^4 + 0 \cdot 2^4 + 1 \cdot 2^4 + 0 \cdot 2^4 + 1 \cdot 2^4 = 0 + 0 + 4 + 0 + 1 = 5_{(10)}$$

En una representación de 6 bits en complemento a 2, los valores posibles van de -32 a 31 (ya que el rango en complemento a 2 con 6 bits es de $-2^{(n-1)}$ a $2^{(n-1)} - 1$).

$5_{(10)}$ está dentro de este rango, por lo que **no se produce un desbordamiento**.

- c) [10 %] Son números naturales.
(Sección 2.2. Números naturales)

Dados los números naturales en base 2 $110110_{(2)}$ y $011001_{(2)}$, realizamos la suma:

$$\begin{array}{r} 11 \quad (1) \\ 110110 \quad (2) \\ + 011001 \quad (3) \\ \hline 1001111 \quad (4) \end{array}$$

Donde:

- **(1)**: Acarreo
- **(2)**: Magnitud de $011011_{(SM2)}$
- **(3)**: Magnitud de $110110_{(SM2)}$
- **(4)**: Resultado de la suma

Como podemos apreciar, el resultado de la operación es $1001111_{(2)}$, el cual es un número de 7 bits. Estamos trabajando con 6 bits y, si ignoramos el acarreo extra y tomamos solo los 6 bits de la derecha, vemos que el resultado sería $001111_{(2)}$, lo cual no representa el valor correcto. Por lo tanto, **sí hubo desbordamiento**.



Ejercicio 4 [25 %]

Dado el formato de coma flotante siguiente:

S	Exponente			Mantisa		
13	12		8	7		0

Donde:

- el bit de signo S vale 0 para los números positivos, y 1 para los negativos
- método de aproximación por truncamiento
- el exponente se codifica en exceso a 16, y
- la mantisa está normalizada de la forma 1,M y con bit implícito.

(Sección 3.3.2. Representación en coma flotante)

a) [10 %] ¿A qué número decimal corresponde la secuencia de bits 01010011001100?

Los números en coma flotante toman la forma $\pm R \cdot b^e$ donde se indica el signo, R es un número fraccionario que recibe el nombre de **mantisa**, b es la **base** de la numeración y e es un número entero que recibe el nombre de **exponente**.

La identificación del formato de la numeración nos indica que el primer bit de la cadena de bits indica el signo. En este caso, el primer bit es un 0 y, por tanto, el número es **positivo**.

Respecto a la mantisa, vemos que ocupa las 8 posiciones menos significativas de la cadena de bits. Por lo tanto, se corresponde con la cadena: 11001100. Ahora bien, el formato indica que hay bit implícito y está normalizada de la forma 1,M. Esto implica que solo muestra la parte variable de las mantisas normalizadas y se asume la parte fija como conocida y definida en el formato de representación. En este ejercicio, el número representado por la mantisa es **1,11001100₍₂₎**.

Respecto al exponente, el formato nos indica que está representado por 5 bits y, además, se nos indica que se ha codificado con exceso a 16. Esto implica que el valor del exponente se conseguirá a partir del resto entre el número codificado con exceso y este exceso. En este ejercicio, el exponente es $10100_{(2)} = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 20_{(10)}$. Sabiendo que hay una codificación con exceso de 16, el exponente será igual a $20_{(10)} - 16_{(10)} = 4_{(10)}$.

Ahora se puede unir el signo, el exponente y la mantisa para obtener el número representado:

$$1,11001100_{(2)} \cdot 2^4$$



Haremos un cambio de base para obtener el valor decimal:

$$\begin{aligned}
 &1,11001100_{(2)} \cdot 2^4 \\
 &= (1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 0 \cdot 2^{-4} + 1 \cdot 2^{-5} + 1 \cdot 2^{-6} + 0 \cdot 2^{-7} + 0 \cdot 2^{-8}) \cdot 2^4 \\
 &= 1,79764344_{(10)} \cdot 2^4 \\
 &= 28,762295_{(10)}
 \end{aligned}$$

b) [10 %] Representad el número $42,625_{(10)}$ en este formato.

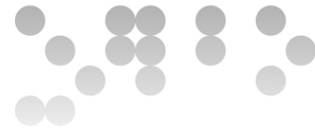
Primero, tenemos que convertir $42,625_{(10)}$ en base 2. Para ello, convertimos primero la parte entera $42_{(10)}$:

DIVISIÓN	COCIENTE	RESTO	
$42 \div 2$	21	0	 Leemos los restos de abajo hacia arriba para obtener el número en base 2
$21 \div 2$	10	1	
$10 \div 2$	5	0	
$5 \div 2$	2	1	
$2 \div 2$	1	0	
$1 \div 2$	0	1	

$42_{(10)}$ en base 2 es $101010_{(2)}$.

Ahora, convertimos la parte decimal $0,625_{(10)}$ a base 2 multiplicando sucesivamente por 2:

$$\begin{aligned}
 0,625 \cdot 2 &= 1,25 \text{ (parte entera 1)} \\
 0,25 \cdot 2 &= 0,5 \text{ (parte entera 0)} \\
 0,5 \cdot 2 &= 1,0 \text{ (parte entera 1)}
 \end{aligned}$$



Leemos la parte entera de arriba hacia abajo para obtener la parte decimal en base 2, la cual es $101_{(2)}$.

Por lo tanto, $42,625_{(10)}$ en base 2 es $101010,101_{(2)}$.

Ahora que tenemos el número en base 2, normalizamos esta representación en la forma $1, M \cdot 2^e$:

$$101010,101_{(2)} = 1,01010101_{(2)} \cdot 2^5$$

Donde:

- $e = 5$, ya que tenemos que mover el punto decimal 5 lugares a la izquierda hasta que la parte entera sea 1.
- $M = 01010101$ representa la mantisa sin bit implícito.

A continuación, codificamos el exponente en exceso a 16, lo cual se calcula así:

$$e_{\text{codificado}} = e_{\text{real}} + 16_{(10)} = 5_{(10)} + 16_{(10)} = 21_{(10)}$$

$$21_{(10)} = 10101_{(2)}$$

Por último, nos falta determinar el bit del signo. Como $42,625_{(10)}$ es positivo, el bit de signo es 0.

Hasta este punto, sabemos lo siguiente:

- **Bit de signo (bit 13):** 0
- **Exponente (bits 12 a 8):** 10101
- **Mantisa (bits 7 a 0):** 01010101 (truncada a 8 bits)

Como resultado, obtenemos que el número $42,625_{(10)}$ en coma flotante con las especificaciones determinadas en el enunciado es $01010101010101_{(2)}$.

- c) [5 %] ¿Se ha producido algún error en la representación del apartado anterior? En caso afirmativo, calculad el mencionado error en formato decimal.

Para comprobarlo, primero vamos a convertir la mantisa a decimal:

$$1,01010101_{(2)} = 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} + 0 \cdot 2^{-5} + 1 \cdot 2^{-6} + 0 \cdot 2^{-7} + 1 \cdot 2^{-8}$$

$$= 1 + 0 + 0,25 + 0 + 0,0625 + 0 + 0,015625 + 0 + 0,00390625$$

$$= 1,33203125_{(10)}$$

Ahora que tenemos la mantisa, multiplicamos por la base con el exponente (2^5):



$$1,33203125_{(10)} \cdot 2^5 = 42,625_{(10)}$$

El número original era $42,625_{(10)}$, y el número representado también es $42,625_{(10)}$.

Por lo tanto, podemos concluir que **no se ha producido ningún error en la representación del apartado anterior.**