PEC1 - Primera prueba de evaluación continua

Presentación

Esta PEC se focaliza en los sistemas básicos de codificación de la información. Es muy importante que se conozca cómo se representa la información dentro de un computador antes de introducir los circuitos combinacionales y secuenciales. La PEC contiene un conjunto de problemas relacionados con los contenidos del Módulo 2.

Competencias

- Saber cómo se representa la información y, en particular, los números de forma digital: números naturales y enteros, tanto en signo y magnitud como en complemento a 2.
- Entender los mecanismos de cambios de base en la representación de números.

Objetivos

- Saber representar un mismo valor numérico en bases diferentes (2,10,16).
- Comprender los conceptos de rango y precisión de los formatos de codificación de la información numérica en un computador, y también los conceptos de desbordamiento y de error de representación.
- Saber representar y operar números naturales en binario.
- Saber representar y operar números enteros en signo y magnitud en base 2.
- Saber representar y operar números enteros en complemento a 2.
- Saber representar y operar números fraccionarios en coma fija.
- Conocer otros tipos de representaciones para almacenar información en un computador.

Recursos

Los recursos que se recomienda utilizar para esta PEC son los siguientes:

- **Básicos**: El módulo 2 de los materiales. En cada pregunta, se indica qué sección de los materiales se puede encontrar la información para resolverlos. Notad que también existe en los materiales una cantidad muy extensa de ejercicios para ver cómo se pueden resolver
- **PEC antiguas:** En el aula de CANVAS, en recursos adicionales podéis encontrar PECs anteriores resueltas
- **Complementarios:** No utilicéis la calculadora para resolver los problemas ya que en el examen no la podréis utilizar. En este momento, os recomendamos que la utilicéis para ver si vuestros resultados son correctos.

Criterios de valoración

- **Razonad la respuesta en todos los ejercicios. Las respuestas sin justificación no recibirán puntuación.**
- **Los ejercicios realizados con IA generativa no recibirán puntuación ya que el objetivo de esta actividad es que entendáis cómo representan la información los computadores. Son ejercicios sencillos que podéis realizar con lápiz y papel sin soportes adicionales.**
- **Los ejercicios que se detecten como plagio mediante la herramienta de plagio de la universidad no recibirán puntuación.**
- **La valoración está indicada en cada uno de los subapartados**

Uso de herramientas de IA

En esta actividad no está permitido el uso de herramientas de inteligencia artificial.

En el plan docente y en el web sobre integridad académica y plagio de la UOC encontraréis información sobre qué se considera conducta irregular en la evaluación y las consecuencias que puede tener.

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).

Enunciado de la PEC

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 calcular la representación decimal del número $10101010_{(2}$ es suficiente con aplicar el TFN (multiplicar cada dígito por el peso de su posición y hacer la suma). Así pues, obtenemos que:

$$10101010_{(2} = 1\cdot2^7 + 0\cdot2^6 + 1\cdot2^5 + 0\cdot2^4 + 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 0\cdot2^0$$

$$= 128 + 32 + 8 + 2 = 170_{(10}$$

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)

Aplicando el TFN obtenemos:

Parte entera: $1\cdot2^4 + 0\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = 16 + 4 + 0 + 1 = 21_{(10}$.
Parte fraccionaria: $0\cdot2^{-1} + 1\cdot2^{-2} + 0\cdot2^{-3} = 0 + 0{,}25 + 0 = 0{,}25_{(10}$.

Así pues, $10101{,}010_{(2} = 21{,}25_{(10}$.

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)

La precisión es la distancia entre dos números representables consecutivos. En este caso diferirán en el bit de menor peso de la representación:

$00000{,}001_{(2} = 2^{-3} = 0{,}125_{(10}$

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)

Utilizamos el método de la división entera por 2:

307	=	153·2	+	**1**
153	=	76·2	+	**1**
76	=	38·2	+	**0**
38	=	19·2	+	**0**
19	=	9·2	+	**1**
9	=	4·2	+	**1**
4	=	2·2	+	**0**
2	=	1·2	+	**0**
1	=	0·2	+	**1**

$$307_{(10} = 100110011_{(2}$$

$307_{(10}$ en binario es $100110011_{(2}$. Como el número es positivo, la representación en Ca2 es $0100110011_{(Ca2}$.

El número mínimo de bits necesario para representarlo es de 10 bits. Podemos comprobarlo teniendo en cuenta que el rango de representación en Complemento a 2 con n bits es desde -2^{n-1} hasta $2^{n-1}-1$:

$$[-2^{n-1}, 2^{n-1}-1] = [-2^{10-1}, 2^{10-1}-1] = [-512, 511]$$

Con 9 bits no sería suficiente ya que $[-2^{9-1}, 2^{9-1}-1] = [-256, 255]$ y $307_{(10}$ no se encuentra dentro de este rango.

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)

Para expresar $-307_{(10}$ en Complemento a 2, partimos de la representación del valor positivo, $307_{(10}$, que hemos calculado en el apartado anterior: $0100110011_{(Ca2}$.

Cambiamos el signo del número complementándolo bit a bit y sumando 1 al resultado:

$$307_{(10} = 0100110011_{(2} \rightarrow 1011001100_{(2} + 1_{(2}:$$

$$\begin{array}{r} 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0 \\ +\qquad\qquad\qquad\qquad\qquad\quad 1 \\ \hline 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \end{array}$$

$$-307_{(10} = 1011001101_{(Ca2}$$

Para representar el valor negativo también necesitaremos 10 bits:

$$-307_{(10} \in [-512, 511]$$

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

En este caso el signo se codifica con el primer bit, que será 1, y la magnitud contiene la representación binaria. En el apartado *a*) la hemos calculado y era $307_{(10} = 100110011_{(2}$.

Así pues, será $-307_{(10} = 1100110011_{(SM2}$.

El número mínimo de bits necesario para representarlo es de 10 bits. Podemos comprobarlo teniendo en cuenta que el rango de representación en Signo y magnitud con *n* bits es desde $-2^{n-1}+1$ hasta $2^{n-1}-1$:

$$[-2^{n-1}+1,\ 2^{n-1}-1] = [-2^{10-1}+1,\ 2^{10-1}-1] = [-511,\ 511]$$

Con 9 bits no sería suficiente ya que $[-2^{9-1}+1,\ 2^{9-1}-1] = [-255,\ 255]$ y $-307_{(10}$ no se encuentra dentro de este rango.

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

En el apartado *a)* se ha visto que la representación en binario del número $307_{(10}$ es $100110011_{(2}$. Para obtener la representación en hexadecimal, hacemos agrupaciones de 4 bits empezando por el bit de menor peso. Como faltan tres bits para hacer la tercera agrupación, añadiremos tres 0 a la izquierda con el fin de completar un grupo de 4 bits:

$$307_{(10} = 100110011_{(2} = 0001\ 0011\ 0011_{(2}$$

Tenemos, pues, tres agrupaciones: 0001 y 0011 y 0011. Ahora traducimos a hexadecimal cada una de estas agrupaciones:

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

Finalmente, obtenemos el número $133_{(16.}$.

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 anterior es incorrecta ya que se produce un error en la conversión del segundo número.

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

La solución correcta en este paso debería ser:

- $011001 \rightarrow 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0$
  $= 16 + 8 + 1 = 25.$

A partir de este paso, todo el desarrollo es incorrecto

De todas formas, tal como se explica en los materiales, la suma de dos números en Complemento a 2 se puede hacer directamente sin pasar a decimal.

Si consideramos que son números en Complemento a 2, podemos hacer la operación de suma directamente:

	1	*1*					<-acarreo
		1	1	0	1	1	0
+		0	1	1	0	0	1
	1	0	0	1	1	1	1

No hay desbordamiento ya que se están sumando dos números de signos diferentes. Por lo tanto, el resultado es correcto.

b) [10 %] Son números enteros en signo y magnitud.
(Sección 2.3.2. Suma y resta en signo y magnitud)

En los números con signo y magnitud el primer dígito codifica el signo. Por este motivo podemos comprobar que los dos números son de signo diferente. Por lo tanto, para realizar la operación de suma se debe eliminar el bit de signo y restar el número de

magnitud más pequeña del otro (en los números dados el número de magnitud mayor es el segundo) y finalmente asignar el signo del número de magnitud mayor al resultado de la resta.

	1	1	0	0	1	
		1	*1*			<-acarreo
-	1	0	1	1	0	
	0	0	0	1	1	
0	0	0	0	1	1	

No hay desbordamiento ya que se han sumado números de signo diferente y por lo tanto el resultado de la operación es siempre correcto.

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

Si consideramos que son números naturales, podemos realizar la suma directamente, pero tendremos que analizar el acarreo ya que puede producir desbordamiento:

1	*1*						<-acarreo
	1	1	0	1	1	0	
+	0	1	1	0	0	1	
1	0	0	1	1	1	1	

Hay desbordamiento ya que hay acarreo en el bit de más peso. Por lo tanto, el resultado de la operación es incorrecto.

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?

El bit de signo es 0 por lo tanto el número es positivo.

La Mantisa es 11001100, que una vez normalizada se obtiene: $1{,}11001100_{(2}$

El exponente es 10100 y por lo tanto corresponde al $10100_{(2} = 20_{(10}$.
Para obtener el número codificado hay que restar el exceso y por lo tanto el número codificado es el $20_{(10} - 16_{(10} = 4_{(10}$.

Por lo tanto, el número decimal a que corresponde es:

$$+ 1{,}11001100_{(2} \cdot 2^4 = + 11100{,}1100_{(2} = 2^4 + 2^3 + 2^2 + 2^{-1} + 2^{-2} =$$
$$16 + 8 + 4 + 0{,}5 + 0{,}25 = + 28{,}75_{(10}$$

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

Para hacerlo, hay que representar la parte entera y posteriormente la parte fraccionaria en base 2:

42	=	21 * 2	+	0
21	=	10 * 2	+	1
10	=	5 * 2	+	0
5	=	2 * 2	+	1
2	=	1 * 2	+	0
1	=	0 * 2	+	1

Por lo tanto $42_{(10} = 101010_{(2}$

0,625	*	2	=	1	,25
0,25	*	2	=	0	,5
0,5	*	2	=	1	,0
0,0	*	2	=	0	,0

Por lo tanto $0{,}625_{(10} = 0{,}101_{(2}$

Combinando las dos partes obtenemos que:

$$42{,}625_{(10} = 101010{,}101_{(2}$$

Con el fin de representar el número en el formato deseado primeramente hay que normalizar el número en el formato utilizado en la representación:

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

La mantisa a codificar es 01010101. Dado que ocupa 8 bits y que el tamaño disponible para su representación es también 8 bits, la mantisa se puede representar correctamente.

El exponente es 5, hay que codificarlo en exceso a 16. Así, el número a codificar es 5 + 16 = 21 y dado que $21_{(10} = 10101_{(2}$ el exponente a incluir en la representación es 10101.

Finalmente, el número $42{,}625_{(10}$ representado en el formato indicado en el enunciado es:

S	Exponente	Mantisa
0	**10101**	**01010101**

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

Al representar el valor anterior, la descomposición en bits del número original se ha realizado de manera exacta. Es decir, no hemos tenido que descartar bits de la parte fraccionaría para construir la mantisa y el exponente se ha podido representar correctamente. Así pues, no se ha producido ningún error.