DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (I/2026)

Ayudantía 1

Ayudantes: Alberto Maturana (alberto.maturana@uc.cl), Mario Rojas (mario.denzel@estudiante.uc.cl)

Pregunta 1: Representación de Números

(a) Convierta los siguientes números decimales a binario:

1. 97_{10}
2. 388_{10}

Solución:

Para pasar los números a binario usaremos la siguiente tabla de potencias de 2.

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
512	256	128	64	32	16	8	4	2	1	0.5	0.25	0.125	0.0625

1. 97_{10}:

$$97 - 2^6 = 97 - 64 = 33$$
$$33 - 2^5 = 33 - 32 = 1$$
$$1 - 2^0 = 1 - 1 = 0$$

Entonces,

$$97 = 2^6 + 2^5 + 2^0$$
$$97_{10} = 1100001_2$$

2. 205_{10}:

$$388 - 2^8 = 388 - 256 = 132$$

$$132 - 2^7 = 132 - 128 = 4$$

$$4 - 2^2 = 4 - 4 = 0$$

Entonces,

$$388 = 2^8 + 2^7 + 2^2$$

$$388_{10} = 110000100_2$$

(b) Reescriba los siguientes números según su representación complemento de 2. Después sume el segundo con el tercero. (ocupe grupo de 4 bits).

1. -7_{10}
2. $FE_{16} = 0xFE$
3. -123_{10}
4. -1_{10}
5. 0_n

Solución:

1. -7_{10}:

   Pasamos el número sin signo a binario,

   $$7_{10} = 111_2$$

   Completamos el grupo de 4 bits con 0's.

   $$111_2 = 0111_2$$

   Para complemento de 2 debemos invertir los bits y después sumar 1.

   $$0111 \rightarrow 1000 \rightarrow 1001$$

   $$-7_{10} = 1001_2 \quad \text{(complemento de 2)}$$

2. FE_{16}:

   Pasamos el número a binario,
   Recomendamos separar cada caracter en 4 bits y luego transformarlos individualmente.

   $$FE_{16} = 1111 \;\; 1110_2$$

   Como el bit más significativo es 1 y el número es positivo, debemos agregar un cero a la izquierda para mantener el signo correcto. Sin embargo, como se nos exige hacerlo con grupo de 4 bits debemos agregar un grupo entero de 4 bits.

   $$1111 \;\; 1110_2 = 0000 \; 1111 \; 1110_2$$

Para representar el número en complemento de 2 no debemos hacer nada más porque es un número positivo.

$$FE_{16} = 0000\ 1111\ 1110_2 \quad \text{(complemento de 2)}$$

3. -123_{10}:

   Pasamos el número sin signo a binario,

   $$123_{10} = 111\ 1011_2$$

   Completamos el grupo de 4 bits con 0's a la izquierda.

   $$1111011_2 = 0111\ 1011_2$$

   Para complemento de 2 debemos invertir los bits y después sumar 1.

   $$01111011 \rightarrow 10000100 \rightarrow 10000101$$

   $$-123_{10} = 10000101_2 \quad \text{(complemento de 2)}$$

4. -1_{10}:

   Pasamos el número sin signo a binario,

   $$1_{10} = 1_2$$

   Completamos el grupo de bits con 0's.

   $$1_2 = 00000001_2$$

   Para complemento de 2 debemos invertir los bits y después sumar 1.

   $$00000001 \rightarrow 11111110 \rightarrow 11111111$$

   $$-1_{10} = 11111111_2 \quad \text{(complemento de 2)}$$

5. 0_n:

   Pasamos el número sin signo a binario,

   $$0_n = 0_2$$

   Completamos (8 bits) con 0's.

   $$0_2 = 00000000_2$$

   En complemento de 2 solo hay una representación.

   $$0_{10} = 00000000_2 \quad \text{(complemento de 2)}$$

6. $FE_{16} + (-7_{10})$: Ya sabemos que (asumimos que -7_{10} ya está en 16 bits:

$$FE_{16} = 0000\ 0000\ 1111\ 1110_2$$

$$-7_{10} = 1111\ 1111\ 1111\ 1001_2$$

Sumamos:

$$0000\ 0000\ 1111\ 1110_2 + 1111\ 1111\ 1111\ 1001_2$$

Resultado (despreciando el acarreo final):

$$0000\ 0000\ 1111\ 0111_2$$

(c) ¿Qué ocurre si sumamos 52_{10} con 19_{10} usando solo 7 bits y representándolos en complemento 2?

Solución:

$$52_{10} = 0110100_2$$

$$19_{10} = 0010011_2$$

$$0110100_2 + 0010011_2 = 1000111_2$$

$$1000111_2 = -57_{10}$$

El resultado real debería ser 71_{10}, pero no cabe en 7 bits con complemento de 2, por lo que ocurre **overflow**.

Pregunta 2: Representaciones Numéricas [I1 - 2025-1]

(1 pto.) Suponga que se suman dos números de 8 bits con signo, A y B, ambos negativos. Como resultado, se genera un acarreo de salida igual a 1. ¿Implica esto necesariamente la ocurrencia de un overflow? Justifique su respuesta.

Solución: No. El *overflow* ocurre cuando el resultado de una operación entre dos números está fuera del rango de su representación. Al usar una representación con signo, el acarreo de salida igual a 1 no implica que exista *overflow*. Esto se puede evidenciar con un contraejemplo en una representación de números enteros con 4 bits:

$$-12 + -22 = 1111 + 1110 = (1)1101 = -32$$

Si bien existe un acarreo de salida, no hay *overflow*. Un acarreo de salida solo evidencia *overflow* en una representación **sin signo**.
Desglose puntaje: Se otorgan 0.5 ptos. por responder correctamente; y 0.5 ptos. por justificación.

Pregunta 3: Representaciones de Números Enteros [I1 - 2025-2]

(3 ptos.) Describa un algoritmo para transformar números naturales codificados en base 8 a base 4 sin hacer uso de la fórmula de transformación de bases $\sum_{k=0}^{n-1} s_k \times b^k$. Es decir, su algoritmo no puede transformar su número a base decimal como paso intermedio. Ilustre su funcionamiento transformando el número 712_8 a base 4.

Solución: A continuación, se describe un algoritmo que cumple con lo solicitado:

1. Transformar cada dígito del número en base 8 en tres bits que lo representen, manteniendo el orden correlativo entre ellos. Por ejemplo, $712_8 = 111|001|010_b$.
2. Si el total de bits no es múltiplo de 2, agregar un 0 a la izquierda. Por ejemplo, $111001010_b = 0111001010_b$.
3. Agrupar los bits en grupos de 2 y transformar cada grupo a su número equivalente en base 4, manteniendo el orden correlativo entre grupos. Por ejemplo, $01|11|00|10|10_b = 1_4|3_4|0_4|2_4|2_4 = 13022_4$.

Este algoritmo funciona ya que tanto la base 8 como la base 4 son **potencias de 2**, por lo que podemos definir un procedimiento similar al de la transformación de base hexadecimal a binaria.

Nota: Se otorgan **2.5 ptos.** por describir un algoritmo que funcione correctamente y **0.5 ptos.** por ilustrarlo. Si describen un algoritmo similar al de la pauta sin el paso 2, solo se descuentan **0.5 ptos.**

Pregunta 4: Desafío Propuesto

Descubra la palabra secreta:

Transforme el siguiente número a su representación binaria y luego a hexadecimal: 51966_{10}. Finalmente, réstele $0xA3E$ al resultado hexadecimal para descubrir la segunda palabra secreta.

Solución: Parte 1. Conversión de 51966_{10} a binario:

Dividimos sucesivamente por 2:

- $51966/2 = 25983$ residuo 0
- $25983/2 = 12991$ residuo 1
- $12991/2 = 6495$ residuo 1
- $6495/2 = 3247$ residuo 1
- $3247/2 = 1623$ residuo 1
- $1623/2 = 811$ residuo 1
- $811/2 = 405$ residuo 1
- $405/2 = 202$ residuo 1
- $202/2 = 101$ residuo 0
- $101/2 = 50$ residuo 1
- $50/2 = 25$ residuo 0
- $25/2 = 12$ residuo 1
- $12/2 = 6$ residuo 0
- $6/2 = 3$ residuo 0
- $3/2 = 1$ residuo 1
- $1/2 = 0$ residuo 1

Leyendo los residuos de abajo hacia arriba:

$$51966_{10} = 1100\ 1010\ 1111\ 1110_2$$

Parte 2. Conversión de binario a hexadecimal:

Agrupamos en bloques de 4 bits:

- $1100_2 = 12_{10} = C_{16}$
- $1010_2 = 10_{10} = A_{16}$
- $1111_2 = 15_{10} = F_{16}$
- $1110_2 = 14_{10} = E_{16}$

La primera palabra secreta es: $\mathbf{CAFE}_{16}$.

3. Resta para hallar la segunda palabra secreta:

Realizamos la operación $CAFE_{16} - A3E_{16}$:

$$\begin{array}{rcccc} & C & A & F & E \\ - & 0 & A & 3 & E \\ \hline & C & 0 & C & 0 \end{array}$$

Resultado: La segunda palabra secreta es **C0C0**.

Feedback ayudantía

Escanee el QR para entregar feedback sobre la ayudantía.

