GUÍA DE EJERCICIOS RESUELTOS DE TECNOLOGÍA Y ORGANIZACIÓN DE COMPUTADORES (TOC)

Para Estudiantes de 1º Doble Grado Matemáticas y Física

10 de octubre de 2025

Índice

Tema 1: Representación de Datos Numéricos

Ejercicio 1: Representación de Enteros en Complemento a 2 (C2)

Enunciado

Suponiendo un computador que opere con una longitud de palabra $n = 8$ bits y que utiliza la representación en **complemento a 2**, ¿cómo se representarían internamente los siguientes números enteros: $+65$ y -37? [1].

Solución

La representación en complemento a 2 (C2) es un método común en la tecnología de computadores [2]. Para $n = 8$ bits:

A) Representación de $+65$ Para números positivos, la representación es simplemente su binario natural [3].

- $65_{10} = (01000001)_2$.
- En C2 (8 bits): **01000001**. (El bit más significativo es 0, indicando signo positivo).

B) Representación de -37 Para números negativos, se utiliza la técnica de Complemento a 2:

1. Obtener el binario de $|-37| = +37$: $37_{10} = (00100101)_2$.
2. Obtener el Complemento a 1 (C1), invirtiendo los bits: $(11011010)_2$.
3. Obtener el Complemento a 2 (C2), sumando 1 a C1:

$$11011010 + 1 = \mathbf{11011011}$$

4. En C2 (8 bits): **11011011**. (El bit más significativo es 1, indicando signo negativo) [1].

Ejercicio 2: Interpretación de Binarios en Diferentes Representaciones

Enunciado

¿Cuál sería el número decimal entero correspondiente al número binario de 8 bits: 1000 1000, suponiendo la representación en **complemento a 2** (C2)? [4].

Solución

El número binario dado es $N = 10001000$ (8 bits). En representación C2, el bit más significativo (MSB) es 1, lo que indica que el número es **negativo** [1].

Para encontrar su valor decimal, calculamos el C2 de N:

1. Binario: 10001000.

2. Complemento a 1 (C1): 01110111.

3. Complemento a 2 (C2): $01110111 + 1 = 01111000$.

4. Conversión a decimal:

$$0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$

$$64 + 32 + 16 + 8 = 120$$

Dado que el número original era negativo, el valor decimal es $\mathbf{-120}$ [4].

Ejercicio 3: Desbordamiento (Overflow) en Complemento a 2

Enunciado

En la representación de **complemento a 2 con signo** (C2), ¿cuándo ocurre un desbordamiento al realizar una suma? [5].

Solución

En la representación en complemento a 2 (C2) con signo, el desbordamiento (overflow) ocurre cuando el resultado de la operación **no es representable** dentro de la longitud de palabra definida [5].

Específicamente, hay desbordamiento en C2 si:

1. Tras sumar dos números **positivos** (ambos con bit de signo 0), el resultado sale **negativo** (bit de signo 1) [5].

2. Tras sumar dos números **negativos** (ambos con bit de signo 1), el resultado sale **positivo** (bit de signo 0) [5].

No ocurre desbordamiento cuando se suman números de diferente signo [5].

Tema 2: Unidades Funcionales de un Computador

Ejercicio 4: Estructura de Buses y Memoria

Enunciado

Un procesador dispone de un registro de dirección de memoria (AR) de 16 bits y un registro de memoria (DR) de 8 bits. Indicar el número de bits de los buses de datos y de direcciones, y el tamaño máximo de la memoria principal [6].

Solución

Los parámetros del procesador definen la arquitectura del sistema:

1. **Número de bits del bus de direcciones (Ancho del bus de direcciones):** Determinado por el tamaño del Registro de Dirección de Memoria (AR).

$$\text{AR} = 16 \text{ bits} \implies \text{Bus de Direcciones} = \mathbf{16} \text{ bits (hilos)[6].}$$

2. **Número de bits del bus de datos (Ancho del bus de datos):** Determinado por el tamaño del Registro de Memoria (DR).

$$\text{DR} = 8 \text{ bits} \implies \text{Bus de Datos} = \mathbf{8} \text{ bits (hilos)[6].}$$

3. **Tamaño en bytes de la memoria principal:** Determinado por la capacidad de direccionamiento (2^n, donde n es el número de bits de dirección) y el tamaño de la palabra de datos (dado por DR).

$$\text{Capacidad} = 2^{16} \text{ direcciones} = 65,536 \text{ posiciones.}$$

Dado que el bus de datos es de 8 bits (1 Byte) [6]:

$$\text{Tamaño de Memoria} = 2^{16} \cdot 1 \text{ Byte} = \mathbf{64} \text{ KB.}$$

Ejercicio 5: Prestaciones del Procesador (MIPS)

Enunciado

Considere tres procesadores (A, B, C) y un programa de prueba (benchmark) que contiene 5 millones de instrucciones. Utilizando los datos de la tabla, indique las prestaciones en MIPS (Millones de Instrucciones Por Segundo) de cada procesador [7, 8].

Procesador	Frecuencia reloj (GHz)	Número de ciclos de reloj por instrucción	Tiempo ejecución I
A	1,5	4	
B	2,0	8	
C	3,0	10	

Solución

El tiempo de ciclo (T_{ciclo}) es el inverso de la frecuencia (f): $T_{\text{ciclo}} = 1/f$ [7]. Las prestaciones en MIPS (Millones de Instrucciones Por Segundo) se calculan como:

$$\text{MIPS} = \frac{\text{Frecuencia del reloj (MHz)}}{\text{Ciclos por instrucción media (CPI)}}$$

Procesador A:

- Frecuencia (f_A) = 1,5 GHz = 1500 MHz.
- CPI (Ciclos por Instrucción) = 4 [8].
- $\text{MIPS}_A = \frac{1500 \text{ MHz}}{4} = \mathbf{375}$ MIPS [8].

Procesador B:

- Frecuencia (f_B) = 2,0 GHz = 2000 MHz.
- CPI = 8 [8].
- $\text{MIPS}_B = \frac{2000 \text{ MHz}}{8} = \mathbf{250}$ MIPS [8].

Procesador C:

- Frecuencia (f_C) = 3,0 GHz = 3000 MHz.
- CPI = 10 [8].
- $\text{MIPS}_C = \frac{3000 \text{ MHz}}{10} = \mathbf{300}$ MIPS [8].

Nota sobre el tiempo de ejecución (aunque no se pide directamente): El tiempo de ejecución (T) se calcula como: $T = \text{Número de Instrucciones} \times \text{CPI} \times T_{\text{ciclo}}$ [7].

$$T_A = 5 \times 10^6 \text{ inst.} \times 4\frac{\text{ciclos}}{\text{inst.}} \times \frac{1}{1{,}5 \times 10^9}\frac{\text{s}}{\text{ciclo}} \approx 0{,}01333 \text{ s[8].}$$

$$T_B = 5 \times 10^6 \text{ inst.} \times 8\frac{\text{ciclos}}{\text{inst.}} \times \frac{1}{2{,}0 \times 10^9}\frac{\text{s}}{\text{ciclo}} \approx 0{,}02000 \text{ s[8].}$$

$$T_C = 5 \times 10^6 \text{ inst.} \times 10\frac{\text{ciclos}}{\text{inst.}} \times \frac{1}{3{,}0 \times 10^9}\frac{\text{s}}{\text{ciclo}} \approx 0{,}01667 \text{ s[8].}$$

Tema 3: Sistemas Combinacionales

Ejercicio 6: Diseño de Sistemas Combinacionales Sencillos

Enunciado

Diseñar un circuito lógico combinacional con tres entradas (a, b, c) y una salida (y) de forma que dicha salida es 1 si y sólo si las señales a y b son 1 **ó** la señal c es 0 [9].

Obtenga la tabla de verdad, la expresión analítica mínima y la implementación del circuito en forma AND/OR/INVERSORES [10, 11].

Solución

Un sistema combinacional es aquel donde las salidas en cualquier instante dependen solo de los valores de las entradas en ese mismo instante [12]. El proceso de diseño incluye obtener la tabla de verdad y minimizar la expresión booleana [10].

1. Tabla de Verdad [13] La condición es $y = 1$ si ($a = 1$ AND $b = 1$) OR ($c = 0$).

Dec	a	b	c	y
0	0	0	0	1 (c=0)
1	0	0	1	0
2	0	1	0	1 (c=0)
3	0	1	1	0
4	1	0	0	1 (c=0)
5	1	0	1	0
6	1	1	0	1 (c=0 o a=1,b=1)
7	1	1	1	1 (a=1,b=1)

2. Expresión Analítica y Minimización (Suma de Productos) [13] La función se define por los minitérminos $m(0, 2, 4, 6, 7)$:

$$y(a, b, c) = \sum m(0, 2, 4, 6, 7)$$

Usando un Mapa de Karnaugh (simplificado en la fuente) [13]:

$c \setminus ab$	00	01	11	10
0	1	1	1	1
1	0	0	1	0

La minimización produce dos grupos: un grupo que cubre todos los 1s en $c = 0$, que es $\bar{c}$, y otro grupo que cubre el m_7, que junto a m_6 ya cubierto por $\bar{c}$ se simplifica a $a \cdot b$ [13].

$$\mathbf{y(a, b, c) = \bar{c} + a \cdot b}[13]$$

3. Implementación AND/OR/INVERSORES [11] La expresión mínima $\mathbf{y = \bar{c} + a \cdot b}$ se implementa con una puerta AND (para $a \cdot b$), un inversor (para $\bar{c}$) y una puerta OR (para la suma de productos) [9, 11].

Ejercicio 7: Sumador Completo de 1 bit

Enunciado

Defina las ecuaciones lógicas para la salida de suma (S) y el acarreo de salida (C_{out}) de un sumador completo de 1 bit, cuyas entradas son A, B y el acarreo de entrada C_{in} [14].

Solución

Un sumador completo (Full Adder) tiene tres entradas (A, B, C_{in}) y dos salidas (S, C_{out}) [14].

Tabla de Verdad (Resumida)

C_{in}	A	B	C_{out}	S
0	0	0	0	0
...	...	...	...	...
1	1	1	1	1

Las funciones de suma (S) y acarreo de salida (C_{out}) son:

Función de Suma (S): La suma S es 1 para los minitérminos $m(1, 2, 4, 7)$ [15].

$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$$

La forma minimizada es la función XOR de las tres entradas:

$$\mathbf{S = A \oplus B \oplus C_{in}}[15]$$

Función de Acarreo de Salida (C_{out}): El acarreo C_{out} es 1 para los minitérminos $m(3, 5, 6, 7)$ [15].

$$C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in}$$

La forma minimizada es:

$$\mathbf{C_{out} = AB + AC_{in} + BC_{in}}[15]$$

Tema 4: Sistemas Secuenciales

Ejercicio 8: Biestable D (Flip-Flop D)

Enunciado

Muestre la tabla de transición (o tabla inversa) para un biestable tipo D y defina cómo se relaciona la entrada D con el estado siguiente Q^+ [16, 17].

Solución

Un biestable (flip-flop) tipo D es un elemento secuencial fundamental [18]. Los sistemas secuenciales necesitan elementos de memoria (como los biestables) para memorizar el estado del sistema [19].

Tabla de Estados Abreviada y de Transición para el Biestable D El biestable tipo D es muy sencillo: el estado siguiente (Q^+) es igual al valor de la entrada D en el flanco activo del reloj [16].

$$\mathbf{Q^+ = D}[16]$$

Tabla Abreviada (Estado Siguiente vs. Entrada D) [16, 17]

D	Q^+ (Estado Siguiente)
0	0
1	1

Tabla Inversa (Tabla de Transición) [16, 17] Muestra el valor de entrada D necesario para producir una transición de Q a Q^+.

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

La tabla de transición confirma la relación fundamental: la entrada del biestable D es directamente el estado siguiente deseado Q^+ [16].

Ejercicio 9: Análisis de Máquina de Estados Finitos (FSM) Tipo Moore

Enunciado

Para la FSM analizada en el ejemplo de la fuente, definida por las funciones de excitación D_0 y D_1 y la salida Y [20]:

$$D_0 = Q_0 \oplus X$$

$$D_1 = \bar{X} \cdot Q_1 + X \cdot \bar{Q}_1 \cdot Q_0 + X \cdot Q_1 \cdot \bar{Q}_0$$

$$Y = Q_1 \cdot Q_0$$

Indique de qué tipo de máquina de estados se trata (Mealy o Moore) y cuál es el comportamiento que describe cuando $X = 1$ [20].

Solución

Tipo de FSM: Una FSM es de tipo **Moore** si las salidas dependen **solo** del estado actual [21, 22]. Es de tipo **Mealy** si las salidas dependen del estado actual **y de las entradas externas** [21, 23].

Observando la función de salida:

$$Y = Q_1 \cdot Q_0$$

La salida Y depende únicamente de las variables de estado Q_1 y Q_0 [20]. Por lo tanto, se trata de una FSM de **Tipo Moore** [21].

Comportamiento cuando $X = 1$: El ejemplo de análisis de la FSM muestra que el sistema es un contador módulo 4, controlado por la entrada X [20].

- Si $X = 0$, el contador se detiene (mantiene el estado actual) [20].
- Si $X = 1$, el sistema pasa por los cuatro estados secuencialmente: $Q_1Q_0 = 00, 01, 10, 11$ [20].

Cuando $X = 1$, la máquina actúa como un **contador binario síncrono ascendente módulo 4** [20]. La salida Y se activa ($Y = 1$) únicamente cuando el contador alcanza el estado $Q_1Q_0 = 11$ (que es el conteo 3) [20].

Tema 5: Sistemas en el Nivel de Transferencia de Registros (RT)

Ejercicio 10: Microoperaciones en Camino de Datos (Buses Dedicados)

Enunciado

Con un esquema de transferencia basado en **multiplexores con buses dedicados** (Ejercicio 1 de la fuente), y asumiendo 3 registros R_0, R_1, R_2, indique si la operación $R_0 \leftarrow R_1$ y $R_2 \leftarrow R_0$ puede efectuarse en un solo ciclo de reloj, indicando los valores de control $S_0, S_1, S_2, W_0, W_1, W_2$ [24].

Solución

Un sistema de transferencia de registros (RT) basado en buses dedicados utiliza buses separados (controlados por S_i) para seleccionar la fuente de datos, y señales de escritura (W_i o LD_i) para habilitar la carga en el registro destino [24, 25].

La microoperación doble $R_0 \leftarrow R_1$ y $R_2 \leftarrow R_0$ requiere dos transferencias simultáneas:

- Transferencia 1: $R_1 \rightarrow R_0$. R_1 debe ser la fuente para R_0.
- Transferencia 2: $R_0 \rightarrow R_2$. R_0 debe ser la fuente para R_2.

Dado que se trata de buses dedicados (generalmente un bus por par de transferencia o similar, según el esquema del Ejercicio 1, donde R_0, R_1, R_2 tienen sus propios multiplexores de entrada/salida [24]), el ejemplo de la fuente muestra que este sistema **permite múltiples transferencias simultáneas** (como $R_0 \leftarrow R_1, R_1 \leftarrow R_2, R_2 \leftarrow R_0$ en un solo ciclo) [25].

No obstante, la operación específica $R_0 \leftarrow R_1$ y $R_2 \leftarrow R_0$ (donde R_0 es fuente y destino simultáneamente) implica que el bus que alimenta a R_0 toma el dato de R_1, y el bus que alimenta a R_2 toma el dato de R_0.

Si consideramos la tabla de soluciones para el Ejercicio 1 [24]:

- $R_0 \leftarrow R_1$: $S_0 = 0, W_0 = 1$ [24].
- $R_2 \leftarrow R_0$: $S_2 = 0, W_2 = 1$ [24].

La operación $R_0 \leftarrow R_1, R_2 \leftarrow R_0$ **NO** puede efectuarse en un solo ciclo.

Operación RT	¿En un solo ciclo?	S_0	S_1	S_2	W_0	W_1	W_2
$R_0 \leftarrow R_1, R_2 \leftarrow R_0$	**NO**	–	–	–	–	–	–

(Nota: El ejercicio 1 del fuente solo lista transferencias simples y una triple, no la combinación específica solicitada) [24, 25]. La respuesta se deduce por las limitaciones inherentes: si R_0 se usa como fuente ($R_0 \rightarrow R_2$) y como destino ($R_1 \rightarrow R_0$), se requiere que R_0 retenga el valor original para R_2 antes de que se cargue el nuevo valor. Sin embargo, en un sistema síncrono, la escritura ocurre en el flanco del reloj. Es posible que el valor de R_0 se lea antes del flanco, y ambos valores se escriban simultáneamente, pero sin un diagrama claro y microoperaciones específicas para esta combinación, la imposibilidad (NO) se mantiene como una restricción lógica general en diseños síncronos simples donde la fuente y el destino coinciden.

Ejercicio 11: Programación en Ensamblador CS1 (Nivel RT)

Enunciado

Utilizando las instrucciones del Computador Sencillo CS1 (Tabla P.1, donde los datos y direcciones son de 8 bits y 6 bits respectivamente [26, 27]), realice un programa que sume los valores que hay almacenados en las direcciones de memoria $M(\$3A), M(\$3B)$ y $M(\$3C)$ y almacene el resultado en la posición de memoria $M(\$3E)$. Es decir: $M(\$3E) \leftarrow M(\$3A) + M(\$3B) + M(\$3C)$ [28].

Solución

El CS1 usa un acumulador (AC) para realizar operaciones aritméticas [29]. Primero, el AC debe inicializarse a cero [28]. Se utiliza una dirección auxiliar, por ejemplo $M(\$3D)$, para realizar la operación $AC \leftarrow AC - M(\$3D)$ (SUB) después de haber hecho $M(\$3D) \leftarrow AC$ (STA) para garantizar que el AC quede a cero, independientemente de su valor inicial [30].

Instrucciones del CS1 [27, 29]:

- STA $\$DirDato$: $M(\$DirDato) \leftarrow AC$ (CO: 11)
- SUB $\$DirDato$: $AC \leftarrow AC - M(\$DirDato)$ (CO: 10)
- ADD $\$DirDato$: $AC \leftarrow AC + M(\$DirDato)$ (CO: 01)
- STOP: Fin ejecución (CO: 00)

Tabla del Programa La dirección $3D$ en binario es 111101_2. Las direcciones $3A, 3B, 3C, 3E$ son $111010_2, 111011_2, 111100_2, 111110_2$ respectivamente [30, 31].

Programa Ensamblador	Descripción RT del programa	CO (2 bits)	Dirección dato (6 bits)	Instr
STA $\$3D$	$M(\$3D) \leftarrow AC$	11	111101	
SUB $\$3D$	$AC \leftarrow AC - M(\$3D)$	10	111101	
ADD $\$3A$	$AC \leftarrow AC + M(\$3A)$	01	111010	
ADD $\$3B$	$AC \leftarrow AC + M(\$3B)$	01	111011	
ADD $\$3C$	$AC \leftarrow AC + M(\$3C)$	01	111100	
STA $\$3E$	$M(\$3E) \leftarrow AC$	11	111110	
STOP	Fin ejecución	00	000000	