1 El examen jul-2023: dos problemas, la misma familia

jul-2023 P1 y P3a · 8 transparencias · para dominar el problema que la primera presentación (slides-tema5-microops) solo rozaba en su nº 6

Problema	Qué pide (literal del enunciado)	Baremo
P1 · procesador de **propósito específico**: multiplicar por sumas sucesivas dos números sin signo de 4 bits («para multiplicar 5 por 4… 5 + 5 + 5 + 5 = 20»); multiplicando en **BR**, multiplicador en **AR**, producto en **PR**	«Diseñar una unidad operativa que tenga capacidad para realizar las operaciones necesarias» + «Escribir un microprograma de transferencia de registros para la unidad de control» + «Diseñar una unidad de control microprogramada»	3,0 p = UO **1,5** + µprograma RT **0,75** + UC **0,75**
P3a · CPU microprogramada: una UO ya dibujada en la figura, unida a un «espacio de direccionamiento de 256 posiciones», con ISA de 5 instrucciones (LDA, INC, MUL2, SUB, STOP)	«diseñar una CPU (unidad de control y microprograma simbólico) que implemente el juego de instrucciones de la tabla»	2,5 p (el P3 completo son 4,0; el 3b de mapa de memoria va en su guía)

La diferencia con la familia von Neumann (deck 1): allí la ruta de datos te la da la figura y tú escribes microoperaciones. Aquí en **P1 la ruta de datos la diseñas tú** (ahí están 1,5 de los 3 puntos: anchos, señales y constantes que tú declaras), y en **P3a** vuelves al modo «usa solo lo dibujado», con una trampa de anchos que luego veremos.

Del propio enunciado (apartado «Importante»): «El planteamiento de cada ejercicio es clave y por tanto debe contestarse de forma correcta. No se puntuarán ejercicios en los que la respuesta a dicha cuestión no sea la adecuada.» → Escribe 2–3 frases de planteamiento y declara tus supuestos *antes* de desarrollar.

Fuentes: enunciado literal en `examenes/texto/EX_SDII_Jul_23.txt` (examen del 7/07/23, P1 y P3a). Baremos y citas entrecomilladas, de ese mismo fichero. Encaje con el deck 1: `web/slides-tema5-microops/index.html` transparencia 6.

2 P1 · Planteamiento: los números que deciden todo el diseño

Antes de dibujar una caja, saca en el papel estos cálculos — puntúan y determinan cada ancho de la UO.

Dimensionado (del enunciado y de la aritmética)

- Entradas de **4 bits** sin signo → BR(4) y AR(4).
- Peor caso **15 × 15 = 225 < 256 = 2^8 → PR de 8 bits**. Con 4 bits desborda.
  regla práctica propia: los anchos se suman, 4+4=8
- Máximo 15 sumas → **AR es el contador del bucle** (no hay contador aparte: ahorro de hardware).
  elección de diseño propia
- La ALU suma parciales de 8 bits → **L, R, C y bus interno de 8 bits**; BR y AR entran al bus con extensión a cero y cargan solo los 4 bits bajos.
- El «1» para hacer `AR ← AR − 1` con L−R exige un **conductor de constante 1** al bus: si no lo prevés, el microprograma es imposible.

Supuestos declarados (escríbelos)

- La UO la diseñas tú: cada registro con `EN_x` (conductor tri-state al bus) y `LD_x` (carga en flanco); BR/AR admiten además **carga directa desde los pines** (`LD_BR*, LD_AR*`) — es el «recibir desde el exterior» del enunciado.
- Señales del sistema: `clk, START, B_ENT(4), A_ENT(4), DONE` + salida del producto (el enunciado no fija el puerto de salida: decláralo tú).
- La guía reutiliza como funciones de la ALU la tabla impresa del examen (la de la figura del P3) — en P1 es **supuesto tuyo**, cualquier permutación de C10–C12 vale si la declaras: solo cambian los cables del decodificador.

```
000→0 · 001→R · 010→L+R · 011→L−R
100→L+1 · 101→ASL(L) · 110→L OR R · 111→NOT L
```

De esa tabla usas solo 4 funciones: **0** (PR←0), **R** (test AR sin tocarlo), **L+R** (acumular), **L−R** (descontar AR).

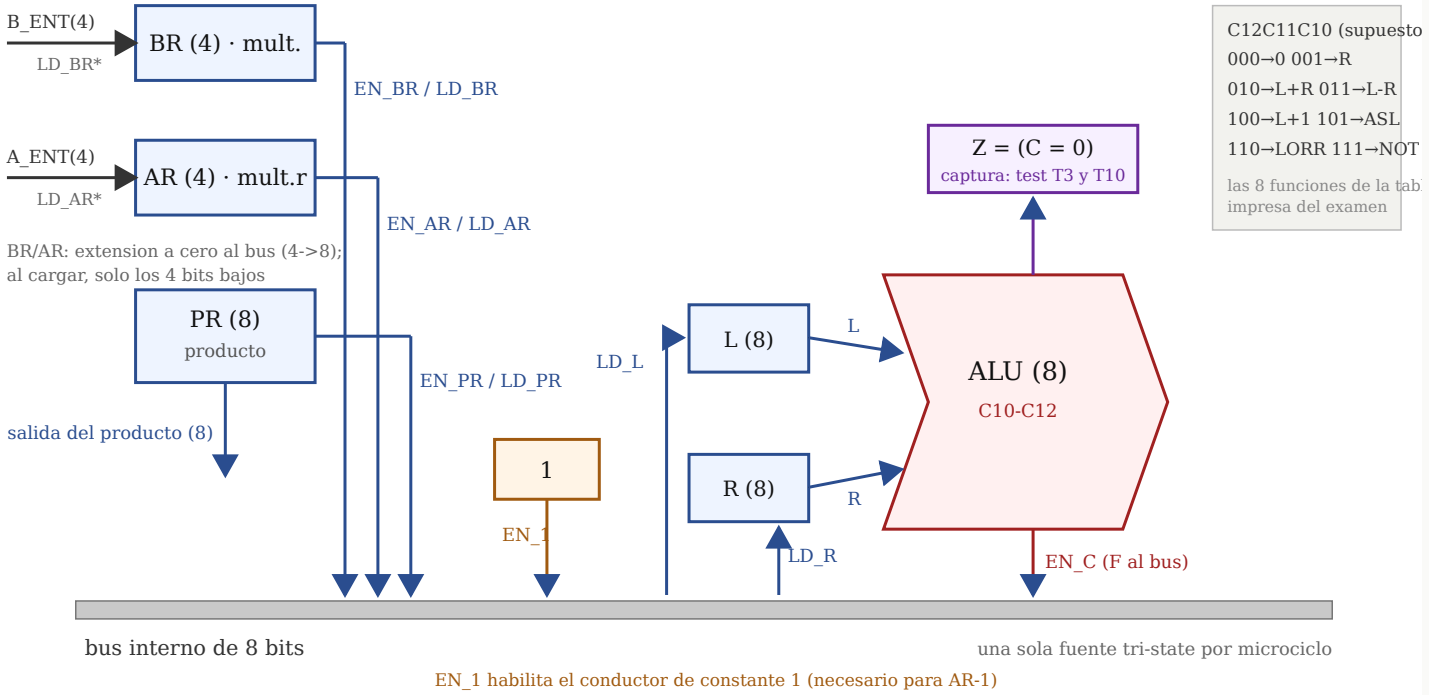
Coste del µprograma (medido con el modelo Python de la guía): `6 + 7·AR µciclos si AR>0; 6 si AR=0` — 5×4 en 34 µc, 1×1 en 13, 13×11 en 83, 15×15 en 111. Y para la UC: **13 microórdenes (T0–T12) → memoria de control de 16 posiciones (MPC de 4 bits)**. Recuento y fórmula: análisis propio verificado por simulación, no texto del profesor.

Fuentes: anchos de entrada y método (5+5+5+5): EX_SDII_Jul_23.txt P1. Cálculos, tabla y conductor de 1: `web/GUIA-julio-2023/index.html` §1.1–1.2 (supuestos declarados de la guía). Medición de µciclos: modelo Python verificado de esa guía. Tabla de ALU impresa: EX_SDII_Jul_23.txt (pertenece a la figura del P3; usarla en P1 es supuesto declarado).

3 P1 · La unidad operativa (1,5 p): dibujarla bien es la mitad del ejercicio

Diagrama redrawn del de la guía §1.2 (misma UO; dibujo simplificado). Todo paso interno = «una fuente al bus → un destino lo carga».

UO del P1 (jul-23): bus interno de 8 bits, una sola fuente al bus por microciclo



- **Dos señales por registro:** `EN_x` habilita el conductor tri-state al bus (si no, alta impedancia); `LD_x` manda la carga desde el bus en el flanco.
- **Z sale de la ALU** (Z = 1 cuando C = 0), capturada al cargar AR y en el test de la guarda — no «mirando AR» en el aire.

Fuentes: UO, señales y diagrama: `web/GUIA-julio-2023/index.html` §1.2 (diagrama de bloques verificado; aquí redrawn en SVG con la misma estructura). Tabla de funciones impresa: EX_SDII_Jul_23.txt. Convención EN/C (alta impedancia): misma que en los enunciados de la familia (p. ej. EX_SDII_jun_24.txt P3).

4 P1 · El microprograma de transferencia de registros (0,75 p)

Notación RT, 13 microórdenes. Fíjate en la guarda (T2–T4) y en que AR se usa como contador: son los dos detalles que distinguen una solución buena.

```
T0 :  BR <- B_ENT ; AR <- A_ENT        (cargas paralelas desde los pines)
T1 :  PR <- 0                         (ALU=0 [000] ; LD_PR)
   -- guarda del multiplicador 0 (evita que 7x0 entre en bucle) --
T2 :  bus <- AR ; R <- bus             (EN_AR ; LD_R)
T3 :  ALU <- R [001] ; Z <- (C=0)      (test sin alterar AR)
T4 :  si Z=1 -> T12 (DONE)             [salto condicionado]
   -- cuerpo del bucle: una suma --
T5 :  bus <- PR ; L <- bus             (EN_PR ; LD_L)
T6 :  bus <- BR ; R <- bus ; ALU <- L+R [010]
T7 :  bus <- C ; PR <- bus             (EN_C ; LD_PR)   PR := PR + BR
   -- decremento de AR (contador) --
T8 :  bus <- AR ; L <- bus             (EN_AR ; LD_L)
T9 :  bus <- 1  ; R <- bus ; ALU <- L-R [011]
T10:  bus <- C ; AR <- bus(4 bajos) ; Z <- (C=0)   AR := AR - 1
T11:  si Z=0 -> T5                     [con Z=1 cae en T12]
T12:  DONE <- 1 ; esperar nuevo START
```

La guarda vale puntos de verdad: sin ella, con multiplicador 0 el `AR−1` envuelve 0→15 y el bucle da 16 vueltas: **7×0 daría 112, no 0** — resultado incorrecto, no solo lento (comprobado por simulación: 115 µc). El test usa la función **R** para mirar AR sin tocarlo.

Traza 5×4 (ejemplo del enunciado)

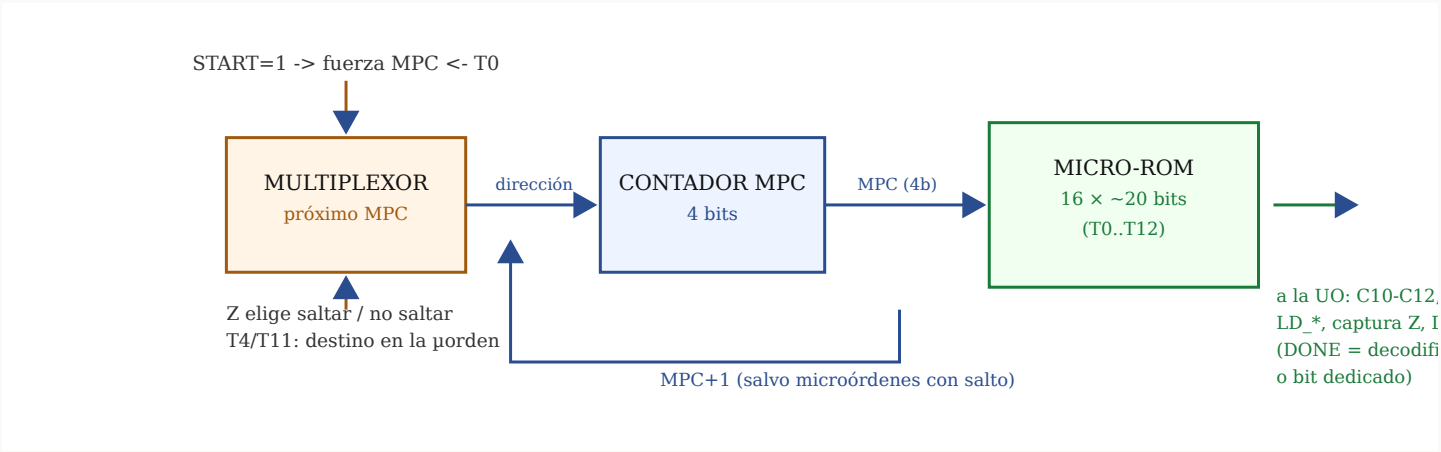
tras…	PR	AR	Z
T1	0	4	0
T7 (suma 1ª/2ª/3ª/4ª)	5 / 10 / 15 / 20	4	0
T10 (1ª/2ª/3ª/4ª vuelta)	5 / 10 / 15 / 20	3 / 2 / 1 / **0**	0/0/0/**1**

Con Z=1 en T10, el salto de T11 no salta → T12. **PR = 20 en 34 µciclos** (medido con el modelo Python de la guía).

Fuentes: microprograma y traza: `web/GUIA-julio-2023/index.html` §1.3 (ejecutados sobre un modelo Python de la UO: 5×4=20 en 34 µc; 7×0=0 con guarda, 112 sin ella — 115 µc). Ejemplo 5×4: literal del enunciado (EX_SDII_Jul_23.txt P1). Teoría: Tema 5, pág. 7–8 del PDF «Tema 5 SD-II_20230215.pdf» (propósito específico: algoritmo y µprograma RT).

5 P1 · La UC microprogramada (0,75 p): dimensionar, no adivinar

Arquitectura multiplexor–contador–memoria (la de la asignatura). El dimensionado sale del microprograma anterior: se cuenta y se redondea a potencia de 2.



La palabra de microorden (~20 bits)

Campo	Bits
Selección ALU (C10–C12)	3
Habilitaciones de bus: EN_BR, EN_AR, EN_PR, EN_1, EN_C	5
Cargas: LD_L, LD_R, LD_PR, LD_AR (+ cargas de pines en T0)	5
Captura de Z	1
Dirección de salto destino	4
Control de salto: no saltar / si Z / si ¬Z	2

Comprobaciones antes de entregar

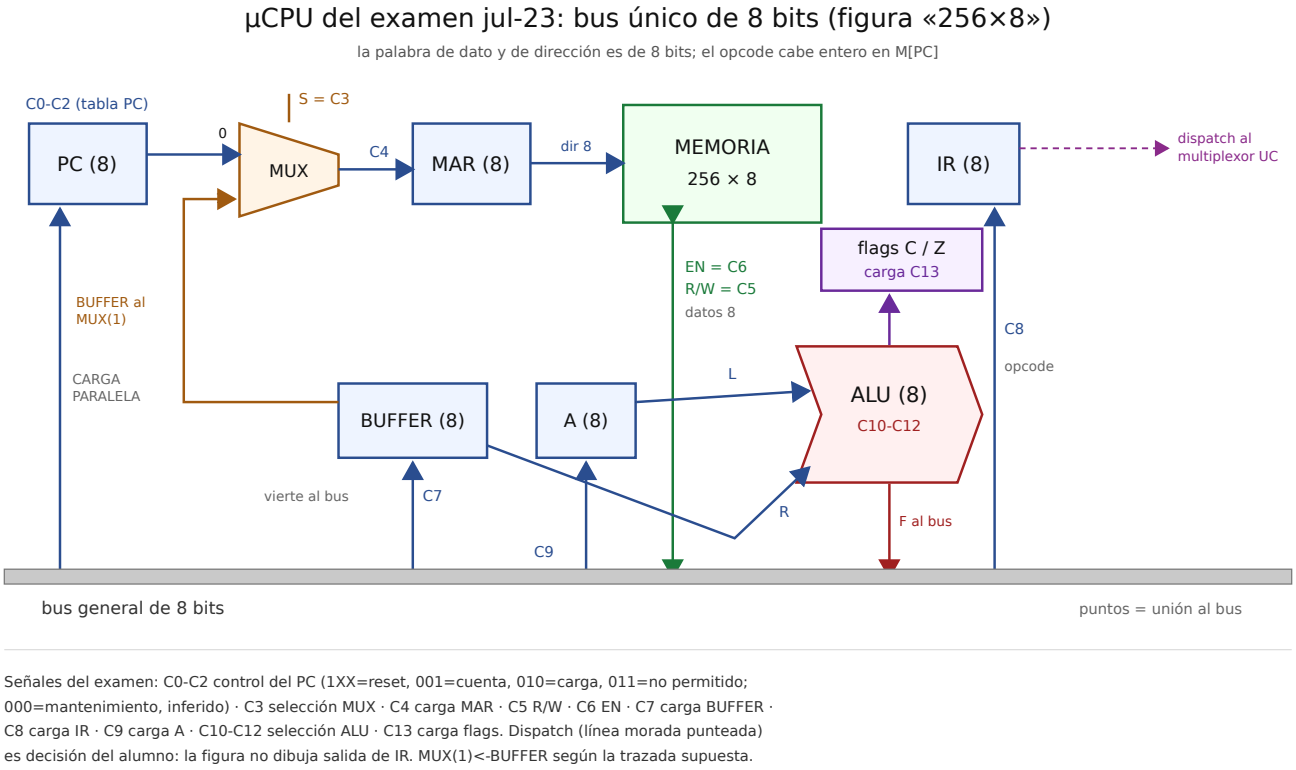
- **Posiciones:** 13 µórdenes (T0–T12) → la menor potencia de 2 es **16 → MPC de 4 bits** (temario: 11 µins → 16 = 2^4; aquí igual con 13).
- **Cada µorden activa una sola fuente EN** (salvo T0, que carga desde pines y no usa el bus) — repasa T2–T11 uno a uno.
- Los saltos **T4→T12 (si Z) y T11→T5 (si ¬Z)** van codificados en la propia palabra; el multiplexor selecciona MPC+1 o el destino según Z.
- START puede implementarse con la entrada RESET/sincronismo del contador.

Errores que rompen el P1: PR de 4 bits (225 desborda) · no prever el «1» para AR−1 · olvidar la guarda (7×0 → 112) · bus de 4 con PR de 8 (incoherencia) · Z «sin decir de dónde sale».

Fuentes: arquitectura, campos y dimensionado: `web/GUIA-julio-2023/index.html` §1.4 y §Errores típicos P1. Regla de la potencia de 2 y estructura MPC/MUX/MC: `temarios/tema-5.txt` diaps. 4–10 y Tema 5, pág. 4–5 y 9–10 del PDF «Tema 5 SD-II_20230215.pdf». Desglose de campos ~20 bits: propuesta de la guía (análisis propio: «~» porque el recuento exacto depende de cómo codifiques las cargas de pines).

6 P3a · La figura manda: qué registros y señales existen (y cuáles no)

Aquí no diseñas la UO: usas **solo** lo dibujado. La trampa está en los anchos — la memoria de la figura está rotulada «256×8».



Señales del examen: C0-C2 control del PC (1XX=reset, 001=cuenta, 010=carga, 011=no permitido; 000=mantenimiento, inferido) · C3 selección MUX · C4 carga MAR · C5 R/W · C6 EN · C7 carga BUFFER · C8 carga IR · C9 carga A · C10-C12 selección ALU · C13 carga flags. Dispatch (línea morada punteada) es decisión del alumno: la figura no dibuja salida de IR. MUX(1)<-BUFFER según la trazada supuesta.

La trampa de anchos (error nº1 de este problema): la figura —imagen incrustada en el PDF, invisible para quien copia el texto— rotula la memoria **«ESPACIO DIRECCIONABLE 256×8»** y todos los buses llevan barra «8». Consecuencias: el opcode de 8 bits **cabe entero en M[PC]**, cada `<dir>` ocupa **una posición entera** (00H–FFH, sin «página cero»), y la ALU tiene **L (de A) y R (de BUFFER)** fijas: invertirlos en SUB deja el resultado complementado. **No hay** MDR aparte (BUFFER hace de registro de datos), ni latches A/B, ni DIR1/DIR2: si añades algo, dilo como extensión. La fila **000 = mantenimiento del PC no está impresa** — infiérela y declárala (la usa STOP).

Fuentes: figura original recuperada del PDF del examen: `figuras/jul23/recortes/figura-2-unidad-operativa-incrustada.png` ; tabla del PC literal (1XX reset, 001 cuenta, 010 carga paralela, 011 NO PERMITIDO) en EX_SDII_Jul_23.txt (final). Análisis de recursos, señales C0–C13 y supuestos: `web/GUIA-julio-2023/index.html` §3.a. Teoría: Tema 5, pág. 22 del PDF «Tema 5 SD-II_20230215.pdf» (procesador de referencia con RAM 256×8, BUFFER y A).

7 P3a · Microprograma simbólico y dimensionado de la UC

Fetch común en 00H–02H, dispatch por opcode en IR (decisión tuya: hay que declararla), y cada bloque termina saltando a 00H. µprograma en hex.

```
FETCH (00H-02H):
00H: MAR <- PC                  (bus <- PC; MUX S=C3=0; C4 carga MAR)
01H: IR <- M[MAR] ; PC cuenta   (EN=C6 activo, R/W=C5 lectura; C8 carga IR)
02H: proximo <- dispatch(IR)    00000001->03H  00000010->07H
                                 00000011->0CH  00000100->11H  00000101->1EH
LDA (03H-06H):   03H: MAR <- PC ; 04H: BUFFER <- M[MAR], PC cuenta
                 05H: MAR <- BUFFER (S=1) ; 06H: A <- M[MAR] ; -> 00H
INC (07H-0BH):   lectura de <dir> igual que LDA ; 0AH: A <- M[MAR]
                 0BH: M[MAR] <- L+1 [100] ; C13 ; -> 00H
                  (el dato escrito sale por el bus desde F; R/W escritura)
MUL2 (0CH-10H):  igual que INC pero 10H: M[MAR] <- ASL(L) [101] ; C13 ; -> 00H
SUB d1 d2 d3 (11H-1DH):
11H-14H: leen d1 y guardan M[d1] en A        (A conserva M[d1] durante todo)
15H-18H: leen d2 y dejan M[d2] en R (BUFFER)
19H: A <- L-R [011] ; C13                    (resultado en A)
1AH-1CH: leen d3 -> MAR ; 1DH: bus <- A ; M[MAR] <- bus ; -> 00H
STOP (1EH): PC en mantenimiento (C0C1C2=000) ; MPC <- 1EH (bucle auto-puntado)
```

Dimensionado (se cuenta, no se adivina)

- µinstrucciones: fetch 3 + LDA 4 + INC 5 + MUL2 5 + SUB 13 + STOP 1 = **31** (posiciones 00H–1EH).
- 16 < 31 ≤ 64 → **memoria de control de 64 posiciones, MPC de 6 bits**; campo de dirección 6 bits.
- Palabra ~30 bits: próximo-MPC (6) + condición/dispatch (2–3) + **las 14 señales del examen C0–C13**. orientativo — la guía lo presenta como propuesta, el recuento depende de tu codificación
- UC: multiplexor–contador–memoria, con IR gobernando el multiplexor en la posición 02H.

Verificación de la guía (modelo Python de la figura)

Programa test `LDA 0EH · INC 0EH · MUL2 0EH · SUB 0EH 0FH 0CH · STOP` con M[0EH]=6, M[0FH]=4: **43 µciclos**; M[0EH] 6→7→14; M[0CH]=14−4=**10**; PC=0BH al STOP. MUL2(9)=18 exacto; MUL2(128)=0 con C=1; SUB(2−4)=254 (complemento a 2) con C=1.

Observación de diseño (declárala): SUB reutiliza A como temporal de M[d1] y del resultado — **destruye el acumulador previo**. Con otra temporal se preservaría, pero la figura no dibuja más registros: añadirla es extensión declarada.

Fuentes: microprograma, recuento 31→64×6 y verificación Python: `web/GUIA-julio-2023/index.html` §3.a (los µciclos citados son mediciones del modelo, análisis propio verificado). ISA y códigos: tabla literal de EX_SDII_Jul_23.txt (longitudes de instrucción inferidas de los operandos — el enunciado no las da; la guía lo declara). Teoría: Tema 5, pág. 17–18 (fetch común y dispatch) y 9 (dimensionado) del PDF «Tema 5 SD-II_20230215.pdf».

8 Errores típicos y receta del problema (P1 y P3a)

Destilado de las secciones «Errores típicos» de la guía jul-2023: cada fallo fue detectado al verificar con modelo; el examinador lo penaliza porque rompe el resultado.

P1 (tú diseñas la UO)

- PR de 4 bits → 15×15 desborda (el caso límite del examen).
- No prever el «1» para AR−1 (conductor de constante u otra ruta).
- Sin guarda AR=0: 7×0 → 112 en vez de 0.
- Bus de 4 con PR de 8: incoherencia de anchos — justifica el bus de 8.
- Z «mirando AR» sin decir que sale de la salida C de la ALU.

P3a (usas solo lo dibujado)

- Ignorar el «256×8» de la figura y suponer palabra de 4 bits.
- INC/MUL2 sin microciclo de **escritura** (R/W) — el dato escrito viaja por el bus desde F.
- SUB sin temporal para M[d1] (usa A y advierte que lo destruye).
- Usar C0C1C2=011 (NO PERMITIDO) o la fila 000 sin declararla.
- Olvidar PC+1 tras cada operando consumido (programa descuadrado).
- No activar **EN (C6)** en las lecturas, o no declarar el dispatch (IR no tiene salida dibujada).
- Intervertir L/R de la ALU en SUB: resultado al revés (M[d2]−M[d1]).

1. **Lee el enunciado dos veces** y apunta el baremo: en P1 el 1,5 p de la UO exige dibujar + justificar anchos y señales; en P3a vale «solo lo dibujado».
2. **Planteamiento escrito** (2–3 frases) + supuestos declarados: funciones ALU adoptadas, conductor de 1, fila 000 del PC, dispatch… «No se puntuarán ejercicios en los que el planteamiento no sea adecuado» es literal del examen.
3. **Microprograma primero, dimensionado después:** cuenta µórdenes → potencia de 2 → bits de MPC y campo de salto → campos de control desde la figura (P1: los tuyos; P3a: C0–C13).
4. **Auto-comprueba con un caso pequeño** (5×4 en P1; un LDA y un SUB en P3a): recorre paso a paso y mira PC y µciclos — un salto a la etiqueta equivocada da resultado correcto y recuento falso.

Dónde seguir: guía completa del jul-2023 con P2 (ASM+VHDL del mismo sistema) y P3b (mapa de memoria): `web/GUIA-julio-2023/` · familia von Neumann (microoperaciones con figura dada): presentación `web/slides-tema5-microops/` · Teoría: Tema 5, págs. 4–10 (UC y dimensionado) y 7–8 y 17–18 (microprogramas) del PDF «Tema 5 SD-II_20230215.pdf». Esta presentación se imprime a PDF con Ctrl+P: una transparencia por página.

Fuentes: listas de errores: `web/GUIA-julio-2023/index.html`, secciones «Errores típicos P1» y «Errores típicos P3.a» (fallos detectados al verificar con modelo Python la propia guía). Cita del planteamiento: EX_SDII_Jul_23.txt, apartado «Importante 1». El orden de la receta y el consejo de auto-comprobación son criterio de preparación propio, no texto del profesor.