1 Qué te van a pedir: el patrón repetido

Microarquitectura de microprocesadores (Tema 5) · 8 transparencias · ~20 min · para el examen del 17/11/2026

De **11 exámenes (dic-2021 → jun-2026)**, **6 incluyen un problema de microoperaciones / microprogramación**, y desde 2023 aparece uno cada año.

Examen · pregunta	Qué pide exactamente	Puntos
jul-2023 · P1	Procesador de **propósito específico** (multiplicar por sumas sucesivas, registros BR/AR/PR): diseñar unidad operativa + **microprograma de transferencia de registros** + UC microprogramada	3,0
jul-2023 · P3a	Diseñar una CPU: **unidad de control y microprograma simbólico** para una UO con espacio de 256 posiciones	2,5
jun-2023 · P3	Unidad de Procesado von Neumann: **microoperaciones** de 7 instrucciones (LDAC, CMAC, ADD, STAC, SUBR, RTN, STOP) + UC **multiplexor–contador–memoria**	3,5
jun-2024 · P3	Igual + separar **ciclo de carga/decodificación** del de **ejecución** + UC + **codificar en binario las 6 primeras microinstrucciones**	3,5
jul-2024 · P2	Igual que jun-2024 (misma figura): microoperaciones + separación de ciclos + UC	4,0
jun-2025 · P4	Igual: microoperaciones (LDAC <dato>, SUB $dir, NAND, JUMP C, JUMP, STAC, STOP) + UC	2,5

Idea clave: el enunciado cambia los nombres de instrucciones y las tablas de señales, pero la **estructura del problema es siempre la misma**: una figura de ruta de datos con bus interior + una tabla de instrucciones + escribir microsecuencias + diseñar la UC. Si dominas un esquema, dominas los seis.

Fuentes: enunciados en `examenes/texto/` (EX_SDII_Jul_23.txt P1 y P3a · EX_SDII_Jun_23.txt P3 · EX_SDII_jun_24.txt P3 · EX_SDII_jul_24.txt P2 · EX_SDII_Jun_25.txt P4). Jun-2022, Sept-2022, Oct-2023, Dic-2021, Jul-2025 y Jun-2026 no traen este tipo de problema (traen mapa de memoria, picoBlaze o ASM+VHDL). Que «desde 2023 aparece cada año» es análisis del material, no texto del profesor.

2 Qué mirar primero: los 6 datos que fija el enunciado

Antes de escribir una sola microoperación, localiza estos seis puntos en el enunciado y en la figura. Ahí está el 90 % de los puntos.

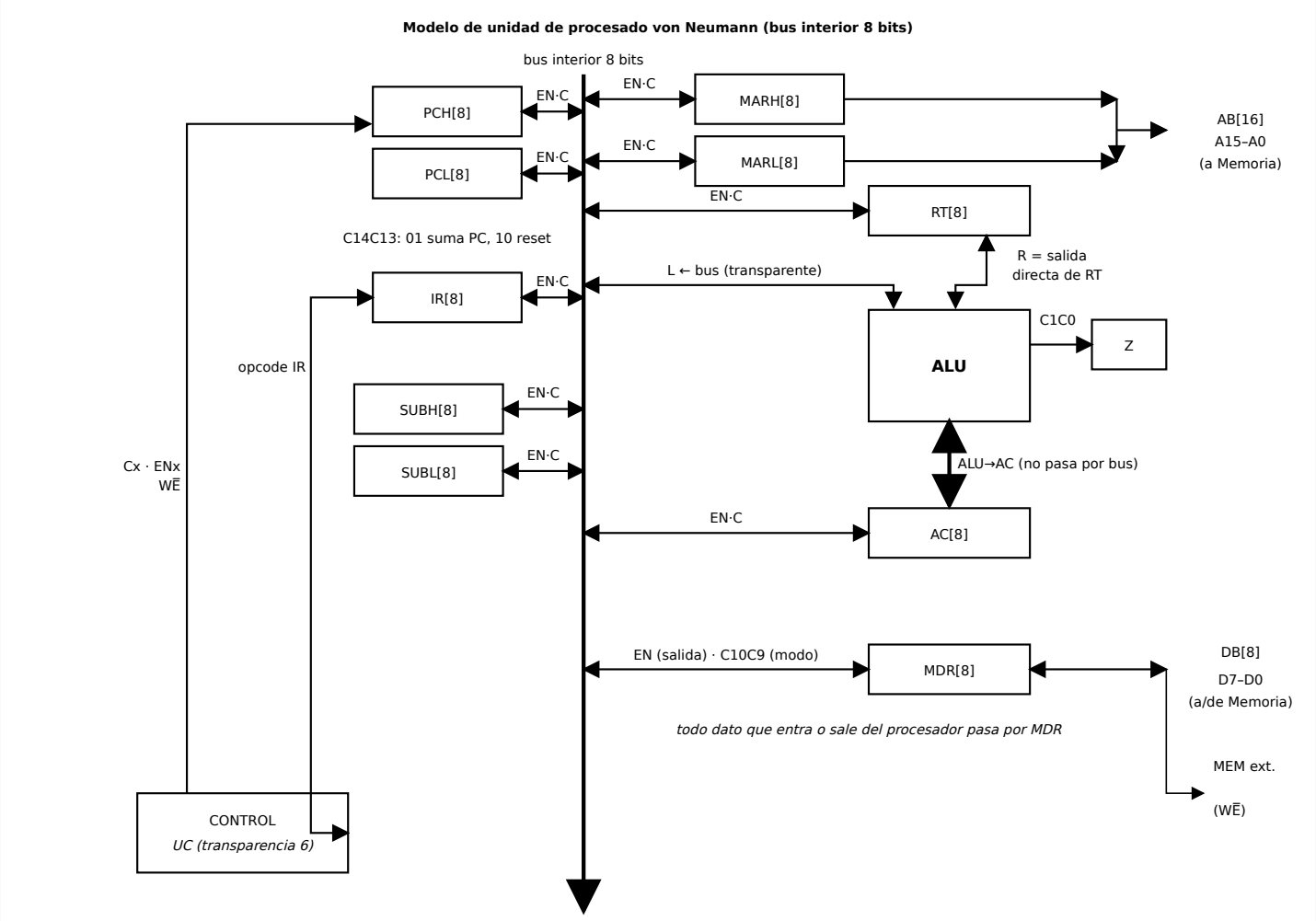
#	Dato	Por qué decide tu solución	Ejemplo citado
1	Anchos de bus: datos 8 (D7–D0), direcciones 16 (A15–A0)	Una dirección de 16 bits **nunca viaja de una vez** por el bus de 8	jun-2024 P3, jul-2024 P2, jun-2025 P4
2	Registros **visibles** (PC = PCH+PCL, AC) y **ocultos** (MAR = MARH+MARL, MDR, IR, RT, SUB)	Solo puedes escribir transferencias entre registros que existen en la figura	jun-2024 P3: «Todo dato que entre o salga del procesador deberá ser almacenado previamente en MDR»
3	Convención de señales: `ENx` habilita la salida al bus («si no está habilitado, alta impedancia»), `Cx` manda la carga	Es la columna «señales de control» de tu tabla de microoperaciones	jun-2024 P3 / jul-2024 P2: parejas EN2–EN11 / C2–C14; jun-2025 P4 repite la regla pero la numeración la fija SU figura
4	Tabla de la ALU (qué hace C1C0) y del PC (C14C13)	**Cambia de un examen a otro:** hay que leerla cada vez, no memorizarla	jun-24/jul-24: 00 L+R · 01 L−R · 10 L · 11 **R+1** | jun-25: 00 **NOT L** · 01 L−R · 10 L · 11 **L AND R**
5	Códigos de MDR (C10C9): 01 = entra de memoria, 10 = sale a memoria	Escribir en memoria exige modo salida + pulso $W\bar{E}$, no vale «MDR ← AC» a secas	Tabla C10C9 idéntica en jun-24 P3, jul-24 P2 y jun-25 P4
6	Formato de instrucción: 1 byte opcode + dato de 8 **o** dirección de 16 en **dos bytes, parte alta primero**	Decide cuántos pasos de fetch/lectura necesitas y el recuento de PC+1	jun-2024 P3: «una dirección de 16 bits dividida en dos partes de 8 bits (primero la parte alta y luego la parte baja)»; igual en jul-2024 P2 y jun-2025 P4

Regla de oro: si la figura y el enunciado no fijan un detalle (una señal, un orden), **decláralo como supuesto en dos líneas** antes de resolver — el profesor no puntúa mal un ejercicio bien planteado con supuesto declarado.

Fuentes: citas literales de los enunciados en `examenes/texto/`: EX_SDII_jun_24.txt P3, EX_SDII_jul_24.txt P2, EX_SDII_Jun_25.txt P4 (tablas ALU y C10C9 incluidas en esos mismos ficheros).

3 El modelo de ruta de datos (el mismo en jun-24 = jul-24)

Una sola idea: **todo paso interno es «una fuente al bus → un registro lo carga»**. La ALU coge L del bus y R de su registro, y solo escribe en AC. Memoria siempre a través de MAR/MDR.



Cómo escribir una transferencia: `MARH ← PCH` = «pon PCH en el bus (su EN) y que MARH la capture (su C)». Un paso = una microoperación, **una sola fuente interna** (dos ENx a la vez = colisión). Lectura/escritura externa y PC+1 sí pueden acompañar a un paso interno: no usan el bus interior.

Fuente del diagrama: versión genérica del diagrama verificado de la guía de jun-2024 P3 (figura idéntica byte a byte a la de jul-2024 P2). La numeración exacta de EN/C la fija **la figura de tu examen**.

4 El ciclo de carga y decodificación: siempre los mismos 4 pasos

Los enunciados de jun-2024, jul-2024 y jun-2025 piden explícitamente «indicar cuáles corresponden al ciclo de carga y decodificación y cuáles al de ejecución». Escríbelos aparte y puntúas gratis.

```
T0  MARH ← PCH    · (EN, C del MARH)
T1  MARL ← PCL    · MAR ya apunta a la instrucción
T2  MDR ← M[MAR] y PC ← PC+1 (externo: C10C9=01; C14C13=01)
T3  IR ← MDR    · opcode al decodificador → salto al microprograma de esa instrucción
```

- **T2 es doble y legal:** la lectura externa no usa el bus interior, así que MDR←M[MAR] y PC+1 van en el mismo paso.
- Después de T3, la **decodificación** dirige el MPC a la secuencia de ejecución de ese opcode (en la UC: salto condicionado por el vector del IR — transparencia 6).
- Contabilidad de PC: **1 byte → +1 en T2; 2 bytes → +2 (T2 y lectura del operando); 3 bytes → +3**. El último incremento va al leer el último byte, no al final.
- Al terminar cada instrucción se **vuelve al inicio del ciclo de carga** (salto `goto FETCH`), salvo STOP.

Y la ejecución arranca leyendo los operandos de la instrucción

El enunciado fija que el operando vive en memoria: «opcode, después ningún operando, o un dato de 8 bits, o una dirección de 16 bits (primero la parte alta)». Toda instrucción con operando empieza apuntando el PC al operando y leyéndolo con MDR — la misma plantilla una y otra vez.

> **Atajo:** si tu ejecución empieza «MARH←PCH · MARL←PCL · leer con MDR», vas bien en todos los exámenes de la familia. Cambia lo que haces con lo leído (transparencia 5), no la lectura.

Fuentes: T0–T3 y recuento de incrementos, verificados paso a paso (modelo Python) en `guias/GUIA-examen-junio-2024.md` P3a; exigencia literal de separar carga/decodificación vs ejecución: jun-2024 P3, jul-2024 P2, jun-2025 P4; cita del operando: enunciado de jun-2024 P3 (repetida en jul-2024 P2 y jun-2025 P4).

5 Ejecución: 6 recetas que cubren todas las instrucciones vistas

Las ISA de los exámenes cambian de nombres, pero las instrucciones caen siempre en estos seis moldes. Ejemplos tomados de las soluciones verificadas de las guías.

Molde	Instrucciones que lo usan (examen)	Micro-receta
A. Cargar de memoria (AC ← M[$dir])	LDAC $dir (jun24, jul24)	Leer los 2 bytes de $dir (guarda ALTA en RT, baja en MDR) → `MARH←RT · MARL←MDR` → `MDR←M[MAR] · AC←MDR`
B. Sumar/restar con operando	ADD (dato) (jun24/jul24), SUB $dir (jun25), ADD <dir1><dir2> (jun23)	Leer el operando → **RT ← MDR** (RT es la entrada R de la ALU) → `AC ← L+R` o `L−R` con el código C1C0 de SU tabla, y `Z ← (resultado=0)`
C. Guardar en memoria (M[$dir] ← AC)	STAC $dir (jun23, jun24, jul24, jun25)	Montar MAR como en A → **dos cosas y en paso distinto**: AC al bus con EN del AC + modo MDR «salida» (C10C9=10), y el pulso `WE̅` . «MDR←AC» a secas no basta
D. Salto	JUMP Z / JUMP C / JUMP (jun24, jul24, jun25)	Montar MAR con $dir como en A → `PC ← M[MAR]` (o PCH/PCL). La decisión Z/C la toma la **UC** (salto condicionado); no uses la ALU antes de decidir o destruyes Z
E. Subrutina	SUBR / RTN-RETS (jun23, jun24, jul24)	Antes de seguir leyendo bytes: `SUBH←PCH · SUBL←PCL` (con el PC ya apuntando a la **siguiente instrucción**) → salto como en D. Al volver: `PC ← SUB`
F. Operación repetida con la ALU	MUL3 (jun24, jul24), CMAC (jun23)	La ALU no multiplica: `×3 = (AC+AC)+AC` . Deja un operando fijo en RT (`RT←AC`) y encadena sumas `AC←L+R` . Con una sola suma dando L=R=AC obtendrías 2×AC

Trampa recurrente (jun24/jul24): no existe camino ALU→bus ni registro R cargable: **R es la salida directa de RT y la ALU solo escribe AC**. Todo operando que quieras sumar, descárgalo antes en RT. Si la figura de tu examen cambia esto, la figura manda.

Fuentes: microsecuencias verificadas con modelo Python en `guias/GUIA-examen-junio-2024.md` P3a (recetas D/E, resumidas de sus secciones homólogas); ISA citadas de EX_SDII_Jun_23.txt P3, EX_SDII_jun_24.txt P3, EX_SDII_jul_24.txt P2, EX_SDII_Jun_25.txt P4.

6 El microprograma y la UC multiplexor–contador–memoria

Todos los problemas de la familia piden «diseñar la unidad de control (arquitectura multiplexor–memoria–contador) y conectarla», y jun-2024 añade «codificar en binario las primeras 6 microinstrucciones» (0,5 p).

Paso 1 · microprograma simbólico

Escribe cada microoperación en una posición numerada de la memoria de control (MC), con etiquetas y saltos:

```
0  FETCH   MARH ← PCH
1          MARL ← PCL
2          MDR ← M[MAR], PC ← PC+1
3          IR ← MDR,  goto DEC
...
k  LDAC    ...  goto FETCH
```

Así lo presenta el temario (diapositivas 7–8, «Diseño de la unidad de control» I y II) y así lo exige el modelo de las guías.

Paso 2 · dimensionar la MC y la palabra

- **Nº de posiciones:** la menor potencia de 2 ≥ nº de µinstrucciones (temario diapositiva 9: 11 posiciones → 16 = 2^4).
- **MPC** (contador de microprograma) y **campo de direcciones de salto**: esos 4 bits.
- **Selección de condiciones** = ⌈log2(nº condiciones + 2)⌉: 000 no saltar, 111 salto incondicional, el resto elige qué señal mira el MUX (tabla diapositiva 6: aquí Z, carry, bits del IR…).
- **Señales de control** = los bits ENx/Cx/W$\bar{E}$/C1C0/C10C9… de la figura (en jun-2024 la palabra salió de 24 bits).

Paso 3 · codificar = traducir, no inventar

Cada palabra es literal: *qué condición compruebo* | *a dónde salto si se cumple* | *qué EN/C activo*. Los pasos de transferencia pura llevan condición «no saltar». Cuidado con **saltar a la etiqueta equivocada** (cazado en jul-2023 al verificar con intérprete): el resultado puede salir bien y solo el recuento de µciclos delata el fallo — comprueba el nº de µciclos que promete tu propio microprograma.

Fuentes: arquitectura MPC/MUX/MC/CMBR, tabla de condiciones y dimensionamiento: Tema 5 (`temarios/tema-5.txt`, diaps. 4–10; su ejemplo de sumas sucesivas = jul-2023 P1). Exigencia de la UC: jun-2023 P3, jun-2024 P3, jul-2024 P2, jun-2025 P4 (enunciados). Palabra de 24 bits y salto a etiqueta equivocada (cazado con modelo Python): `guias/GUIA-examen-junio-2024.md` P3c; recuento de µciclos verificado con intérprete: `guias/GUIA-examen-julio-2023.md` P1.

7 Errores típicos (los que de verdad cuestan puntos)

Lista destilada de las verificaciones automáticas de las guías: cada uno de estos fallos fue detectado y corregido en curso; el examinador los penaliza porque rompe el reloj o el programa.

- **No separar carga/decodificación de ejecución.** Es literalmente lo que piden (2,25 p en jun-2024). Escribe T0–T3 comunes una sola vez.
- **Montar una dirección de 16 bits «en un paso».** La dirección llega en dos bytes por un bus de 8: hacen falta dos lecturas y RT/MDR de puente. `MAR ← PC` a secas es incorrecto con esta figura.
- **Escribir transferencias que la figura no permite** (registro R cargable, ALU→bus, «RT ← resultado de la ALU»): la ALU solo escribe AC y R viene fija de RT.
- **PC+1 de más o de menos.** 2 bytes → +1 en T2 y +1 al leer el operando; 3 bytes → el tercero al leer el byte bajo. Con un +1 de más el programa se descuadra tras la primera instrucción.
- **«MDR ← AC» a secas para escribir en memoria:** falta modo salida C10C9=10 y el pulso $W\bar{E}$ en paso distinto.
- **Dos ENx internos a la vez (colisión de bus):** cada µ-paso admite una sola fuente; sí pueden acompañar lectura/escritura externa y PC+1.
- **Usar la ALU antes de un JUMP Z y destruir Z**, o «decidir en el aire»: la decisión se materializa en el campo de selección de condiciones de la µinstrucción.
- **SUBR: guardar SUB demasiado tarde** (después del último PC+1) → al volver (RETS) caerías en mitad del operando en vez de en la instrucción siguiente.
- **Asumir que la tabla ALU/PC o la numeración EN/C es la del año pasado:** jun-2024 tiene 11=R+1 y jun-2025 tiene 00=NOT L y 11=L AND R. Lee SU tabla antes de poner códigos.

Fuentes: sección «Errores típicos — P3» de `guias/GUIA-examen-junio-2024.md` (los fallos se inyectaron a propósito en el modelo Python y fueron detectados 4/4); tablas ALU citadas de EX_SDII_jun_24.txt P3 y EX_SDII_Jun_25.txt P4; peso del desglose carga/ejecución: baremo literal de jun-2024 P3 (2,25 p).

8 Receta de ejecución en el examen (orden y ejemplo completo)

Orden recomendado para los ~25–35 min del problema, y un esqueleto resuelto (LDAC $dir, jun-2024) con el formato exacto que se espera.

1. **Lee los 6 datos** (transparencia 2) y anótalos en 4 líneas: anchos, registros, señales, tablas ALU/PC/MDR, formato de instrucción.
2. **Escribe el fetch T0–T3** y dilo: «esto es carga/decodificación, común a todas».
3. **Clasifica las instrucciones** en los 6 moldes (transparencia 5) y resuelve un molde por clase antes de repetir.
4. **Auto-comprueba cada paso:** ¿existen fuente y destino en la figura? ¿una sola fuente al bus? ¿PC+1 donde toca? ¿la escritura externa lleva modo + $\overline{WE}$?
5. **UC:** µprograma numerado → dimensiona MC/campos → codifica solo si se pide. Comprueba los µciclos que promete tu propio microprograma.

Esqueleto verificado: LDAC $dir (3 bytes), ejecución completa en jun-2024

```
E0  MARH ← PCH          E5  MDR ← M[MAR] (byte BAJA de $dir), PC ← PC+1
E1  MARL ← PCL          E6  MARH ← RT          (alta ya guardada)
E2  MDR ← M[MAR] (byte ALTO de $dir), PC+1   E7  MARL ← MDR → MAR = $dir
E3  RT  ← MDR           E8  MDR ← M[MAR] (el DATO; sin PC+1)
E4  MARL ← PCL          E9  AC  ← MDR
(EN/C en la guía: T2+E2+E5 = los 3 incrementos del PC)
```

Dónde seguir: guías publicadas con la resolución completa y widgets en el portal (jun-2024 P3 = plantilla maestra; jun-2023 P3, jul-2024 P2, jun-2025 P4, jul-2023 P1/P3) · Teoría: PDF del Tema 5, diapositivas 4–23 (25–42 ya son mapa de memoria, otro problema) · Esta presentación se imprime a PDF con Ctrl+P: una transparencia por página.

Fuentes: esqueleto LDAC transcrito de la solución verificada (modelo Python) en `guias/GUIA-examen-junio-2024.md` P3a; referencias de diapositivas a `temarios/tema-5.txt`. El reparto de tiempos (3–4 min lectura, 25–35 min problema) es criterio de preparación, no viene del profesor.