✅ Resumen para el tipo test (preguntas 1–40)

Te agrupo los temas clave que aparecen en el test, con explicaciones breves para que puedas repasarlos rápidamente:

🧩 Arquitectura y jerarquía del computador

- **Von Neumann**: CPU, memoria única para datos e instrucciones, entrada/salida.
- **Jerarquía de transformación**: Aplicaciones → compiladores → ISA → microarquitectura → lógica digital → circuitos.
- **ISA (Instruction Set Architecture)**: conjunto de instrucciones que entiende el hardware.

💾 Memoria y almacenamiento

- **DRAM**: memoria principal, más lenta que la caché.
- **SRAM**: usada en caché, más rápida y cara.
- **Caché**: explota la localidad temporal y espacial.
- **Memoria virtual**: da la ilusión de más memoria, gestionada por el sistema operativo.

🧮 Codificación y representación

- **Binario y desplazamientos**: desplazar a la izquierda = multiplicar por 2.
- **Máscaras AND**: permiten aislar bits.
- **Codificación de texto**: ASCII, ISO 8859-1/15, Unicode.
- **RGB**: modelo aditivo (rojo, verde, azul).
- **Mapas de bits**: representación por píxeles.

🧠 Procesadores y ejecución

- **Contador de programa (PC)**: apunta a la siguiente instrucción.
- **Segmentación (pipelining)**: varias instrucciones en paralelo en distintas etapas.
- **Tendencias modernas**: más núcleos, no más frecuencia.

🧰 Sistemas operativos

- **Servicios**: gestión de procesos, E/S, sistema de archivos.
- **Llamadas al sistema**: síncronas (ej. `ecall`).
- **Interrupciones**: asíncronas (ej. pulsación de tecla).
- **Dispositivos como ficheros** en UNIX/Linux.

👩‍💻 Lenguajes y ejecución

- **Python**: interpretado, más lento que C, pero más rápido de programar.
- **Bytecode y PVM**: Python se compila a bytecode, luego interpretado.
- **ISA virtual**: capa intermedia para facilitar portabilidad.

📚 Enlazado

- **Estático**: copia las bibliotecas en el ejecutable (más pesado).
- **Dinámico**: comparte bibliotecas en memoria (más eficiente).

🧠 Cómo resolver los problemas (P1–P4)

🔢 P1: Representación numérica compacta

- Usa **notación exponencial**: bits para exponente y mantisa.
- Elige el número de bits del exponente según el rango (usa $\log_2$).
- Usa **notación sesgada** para representar exponentes negativos.
- Calcula el valor como:
  $\text{Valor} = (1 + \text{mantisa}) \cdot 2^{\text{exponente real}}$

🧮 P2: Análisis de código en RISC-V

- Identifica qué registros se usan para variables (`x10`, `x11`, etc.).
- Sigue el flujo del programa con el **PC** (contador de programa).
- Analiza instrucciones como `lw`, `addi`, `bge`, `j`.
- Para codificar instrucciones, usa el formato binario de RISC-V:
  - `addi`: 12 bits inmediato + 5 fuente + 3 funct + 5 destino + 7 opcode.

🧠 P3: Acceso a memoria y caché

- **Alternativa 1**: acceso secuencial → buena localidad espacial.
- **Alternativa 2**: acceso disperso → más fallos de caché.
- Usa bits de dirección para:
  - **Offset** (dentro del bloque): $\log_2$(tamaño bloque).
  - **Índice** (posición en caché): $\log_2$(nº bloques).
  - **Etiqueta (tag)**: el resto.
- Determina si hay acierto o fallo en caché según el tag y el índice.

💻 P4: Teoría breve

- **E/S**: gestionada por controladores (hardware) y drivers (software).
- **Llamada al sistema**: instrucción explícita del programa.
- **Interrupción**: evento externo que interrumpe la ejecución.