

Unidad de Aprendizaje: Laboratorio de Informática **Alumno:** _____

Titulación: Grado en Ingeniería Informática **IDUCAM:** _____

Docente: Dra. en C. Ing. Angélica Guzmán Ponce

Instrucciones:

- Este **simulacro de ejercicios** está compuesto por **5 ejercicios de desarrollo**, referentes a los Temas 1, 2, 3, 4 y 5, del mismo tipo que los de la parte de desarrollo del examen parcial. La puntuación total es de 10 puntos; la de cada ejercicio se indica entre corchetes al inicio de su enunciado.
- No está permitido el uso de dispositivos electrónicos (teléfonos, relojes inteligentes, audífonos, etc.) ni de cualquier tipo de material o apoyo adicional (textos, manuales, etc.).

1) [2 Puntos] La fuente de alimentación

Una agencia de diseño encarga un equipo para editar fotografía y vídeo, en caja torre. Estos son sus componentes, con el consumo máximo de cada uno:

Componente	Consumo máximo
Procesador	120 W
Tarjeta gráfica dedicada	220 W
Placa base y memoria	40 W
Dos unidades SSD	5 W cada una
Cinco ventiladores	2 W cada uno

a) ¿Cuál es el consumo máximo del equipo?

Solución. Todos los componentes se alimentan de la misma fuente a la vez, así que el consumo del equipo es la **suma** de los consumos de todos ellos. Los componentes repetidos se multiplican por el número de unidades, porque la tabla da el consumo de *cada una*:

Componente	Cuenta	Consumo
Procesador		120 W
Tarjeta gráfica dedicada		220 W
Placa base y memoria		40 W
Dos unidades SSD	2×5	10 W
Cinco ventiladores	5×2	10 W
Total	$120 + 220 + 40 + 10 + 10$	400 W

Se usa el consumo *máximo* de cada componente, y no el habitual, porque es el peor caso: todos trabajando al máximo a la vez. La fuente tiene que poder entregar esa potencia; si no, el equipo podría apagarse o reiniciarse en los momentos de más carga.

b) ¿Qué potencia de fuente recomendada necesita el equipo? Entre fuentes de 500, 850 y 1200 W, ¿cuál eliges? ¿Qué problema tendría cada una de las otras dos?

Solución. Potencia recomendada. La regla práctica es elegir una fuente de más o menos el **doble del consumo máximo**:

$$2 \times 400 = 800 \text{ W.}$$

Así, incluso con el equipo al máximo, la fuente trabaja en torno al 50 % de su potencia nominal, que es la zona en la que mejor rinde (donde su eficiencia es mayor). Además, queda margen para picos de consumo y para futuras ampliaciones, y la fuente trabaja más fría y silenciosa y dura más.

Comparación de las tres fuentes. Para cada una se calcula a qué porcentaje de su potencia nominal trabajaría con el equipo al máximo (400 W):

Fuente	Carga con el equipo al máximo	Porcentaje	Valoración
500 W	$400/500 = 0,80$	80 %	va muy justa
850 W	$400/850 \approx 0,47$	47 %	adecuada: cerca del 50 %
1200 W	$400/1200 \approx 0,33$	33 %	sobredimensionada

Elección: 850 W. No hay ninguna de 800 W entre las opciones, así que se elige la más próxima *por encima*, para no quedarse corto. Trabajaría al 47 %, muy cerca del 50 % recomendado.

Problema de la de 500 W: con el equipo al máximo trabajaría al 80 %. Iría muy justa: sin margen para picos de consumo ni para futuras ampliaciones, más caliente y ruidosa, y envejecería antes.

Problema de la de 1200 W: trabajaría al 33 % en el peor caso, y muy por debajo en el uso normal. Está sobredimensionada: es más cara y, con poca carga, la eficiencia de una fuente baja, así que desperdiciaría más energía.

- c) Dudas entre dos fuentes de 850 W. Con el equipo consumiendo 360 W, la fuente A toma de la red 450 W, y la B, 400 W. ¿Qué eficiencia tiene cada una? ¿Qué pasa con la energía que toman de la red y no llega a los componentes?

Solución. Eficiencia. Indica qué parte de la potencia que la fuente toma de la red llega de verdad a los componentes:

$$\text{eficiencia} = \frac{\text{potencia entregada a los componentes}}{\text{potencia tomada de la red}} \times 100.$$

Las dos fuentes entregan los mismos 360 W, pero toman de la red potencias distintas:

$$\text{Fuente A: } \frac{360}{450} \times 100 = 0,80 \times 100 = \mathbf{80 \%}$$

$$\text{Fuente B: } \frac{360}{400} \times 100 = 0,90 \times 100 = \mathbf{90 \%}$$

La B es más eficiente: para entregar los mismos 360 W, toma menos de la red.

Energía perdida. Lo que la fuente toma de la red y no entrega a los componentes se pierde dentro de la propia fuente:

$$\text{Fuente A: } 450 - 360 = 90 \text{ W}$$

$$\text{Fuente B: } 400 - 360 = 40 \text{ W}$$

Esa energía se pierde al convertir la corriente alterna de la red en la corriente continua que usan los componentes, y se transforma en **calor**, que expulsa el ventilador de la fuente. Por eso importa: se paga en la factura eléctrica sin aprovecharse, y calienta el equipo y la sala. Con la B se pierden $90 - 40 = 50 \text{ W}$ menos.

2) [2 Puntos] Cambios de base

Resuelve los siguientes cambios de base, escribiendo el procedimiento que sigues en cada caso.

- a) Pasa 1011100110_2 a octal, a hexadecimal y a base 5. En cada caso, elige el camino más corto y explica por qué. Después, comprueba que los tres resultados valen lo mismo en decimal.

Solución. Octal: agrupar los bits de 3 en 3. Como $8 = 2^3$, cada cifra octal equivale exactamente a 3 bits, así que no hace falta pasar por decimal. Los bits se agrupan de 3 en 3 *desde la derecha*, porque la cifra de la derecha es la de menor peso, y el grupo de la izquierda se completa con ceros, que no cambian el valor:

$$1 \ 011 \ 100 \ 110 \rightarrow 001 \ 011 \ 100 \ 110$$

Cada grupo se convierte en una cifra con los pesos 4, 2 y 1:

Grupo de 3 bits	001	011	100	110
Valor (pesos 4, 2, 1)	$0 + 0 + 1 = 1$	$0 + 2 + 1 = 3$	$4 + 0 + 0 = 4$	$4 + 2 + 0 = 6$

Resultado: 1346_8 .

Hexadecimal: agrupar los bits de 4 en 4. Como $16 = 2^4$, cada cifra hexadecimal equivale a 4 bits. También se agrupa desde la derecha y se completa con ceros a la izquierda:

$$10 \ 1110 \ 0110 \rightarrow 0010 \ 1110 \ 0110$$

Cada grupo se convierte en una cifra con los pesos 8, 4, 2 y 1. Los valores de 10 a 15 se escriben con las letras A a F:

Grupo de 4 bits	0010	1110	0110
Valor (pesos 8, 4, 2, 1)	$0 + 0 + 2 + 0 = 2$	$8 + 4 + 2 + 0 = 14$	$0 + 4 + 2 + 0 = 6$
Cifra hexadecimal	2	E	6

Resultado: **2E6**₁₆.

Base 5: hay que pasar por decimal. Como 5 no es una potencia de 2, no hay un número fijo de bits que equivalga a cada cifra, así que no se puede agrupar. El camino es: binario $\rightarrow$ decimal (teorema fundamental de la numeración) $\rightarrow$ base 5 (divisiones sucesivas).

Paso 1: a decimal. Es más corto partir del hexadecimal ya obtenido, que tiene tres cifras, que de los diez bits. Cada cifra se multiplica por el peso de su posición, que son las potencias de 16 empezando por $16^0 = 1$ a la derecha:

$$\begin{aligned} 2E6_{16} &= 2 \times 16^2 + 14 \times 16^1 + 6 \times 16^0 \\ &= 2 \times 256 + 14 \times 16 + 6 \times 1 \\ &= 512 + 224 + 6 = 742. \end{aligned}$$

Paso 2: de decimal a base 5, por divisiones sucesivas. Se divide entre 5 una y otra vez hasta que el cociente es 0. Cada resto es una cifra del resultado: el primer resto es la cifra de las unidades (la de menor peso) y el último, la cifra de mayor peso.

División	Cociente	Resto	
742 : 5	148	2	$\leftarrow$ primera cifra por la derecha (unidades)
148 : 5	29	3	
29 : 5	5	4	
5 : 5	1	0	
1 : 5	0	1	$\leftarrow$ cifra de mayor peso

Los restos se leen *de abajo arriba*: 1, 0, 4, 3, 2. Resultado: **10432**₅.

Comprobación: los cuatro números valen lo mismo en decimal. En binario, se suman los pesos (potencias de 2) de las posiciones que tienen un 1:

Bit	1	0	1	1	1	0	0	1	1	0
Peso	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valor del peso	512	256	128	64	32	16	8	4	2	1

$$1011100110_2 = 512 + 128 + 64 + 32 + 4 + 2 = 742.$$

En octal y en base 5, cada cifra se multiplica por el peso de su posición (potencias de la base):

$$1346_8 = 1 \times 8^3 + 3 \times 8^2 + 4 \times 8^1 + 6 \times 8^0 = 512 + 192 + 32 + 6 = 742$$

$$10432_5 = 1 \times 5^4 + 0 \times 5^3 + 4 \times 5^2 + 3 \times 5^1 + 2 \times 5^0 = 625 + 0 + 100 + 15 + 2 = 742$$

$$2E6_{16} = 742 \quad (\text{calculado en el paso 1})$$

Los cuatro valen **742**, así que las conversiones son correctas.

- b) Pasa $2C7_{16}$ a decimal con el teorema fundamental de la numeración, y comprueba el resultado pasando a decimal 1307_8 .

Solución. Teorema fundamental de la numeración. El valor de un número es la suma de cada cifra multiplicada por la base elevada a la posición de esa cifra, contando las posiciones desde 0 por la derecha. En base 16, los pesos son $16^0 = 1$, $16^1 = 16$, $16^2 = 256$, etc. La cifra C vale 12: la base 16 necesita 16 símbolos y solo hay diez dígitos, así que se usan las letras A a F para los valores 10 a 15.

Cifra	Valor	Peso de la posición	Valor $\times$ peso
2	2	$16^2 = 256$	$2 \times 256 = 512$
C	12	$16^1 = 16$	$12 \times 16 = 192$
7	7	$16^0 = 1$	$7 \times 1 = 7$

$$2C7_{16} = 512 + 192 + 7 = \mathbf{711}.$$

Comprobación con 1307_8 . Mismo procedimiento, con potencias de 8:

Cifra	Peso de la posición	Cifra $\times$ peso
1	$8^3 = 512$	$1 \times 512 = 512$
3	$8^2 = 64$	$3 \times 64 = 192$
0	$8^1 = 8$	$0 \times 8 = 0$
7	$8^0 = 1$	$7 \times 1 = 7$

$$1307_{(8)} = 512 + 192 + 0 + 7 = \mathbf{711}.$$

Los dos dan 711: son el mismo número escrito en dos bases.

También se puede comprobar sin pasar por decimal, a través del binario, porque 16 y 8 son potencias de 2: cada cifra hexadecimal se escribe con 4 bits y después los mismos bits se reagrupan de 3 en 3 desde la derecha:

Cifra hexadecimal	2	C (12)	7	Grupos de 3 bits	001	011	000	111
En 4 bits	0010	1100	0111	Cifra octal	1	3	0	7

Salen las cifras 1, 3, 0 y 7: $1307_{(8)}$.

- c) Pasa 425 a hexadecimal por divisiones sucesivas. A partir del resultado, escríbelo en binario y en octal sin volver a dividir.

Solución. De decimal a hexadecimal: divisiones sucesivas entre 16. Al dividir un número entre la base, el resto es su última cifra en esa base, y el cociente es el número que queda sin esa cifra. Se repite con cada cociente hasta llegar a 0:

División	Cociente	Resto	
425 : 16	26	9	$\leftarrow$ primera cifra por la derecha (unidades)
26 : 16	1	10 (A)	
1 : 16	0	1	$\leftarrow$ cifra de mayor peso

Los restos se leen de abajo arriba: **1A9**₍₁₆₎. Comprobación: $1 \times 256 + 10 \times 16 + 9 \times 1 = 256 + 160 + 9 = 425$.

De hexadecimal a binario: cada cifra, 4 bits. Como $16 = 2^4$, cada cifra hexadecimal se sustituye directamente por sus 4 bits, sin dividir:

Cifra hexadecimal	1	A (10)	9
En 4 bits (pesos 8, 4, 2, 1)	0001	1010	1001

Se juntan los grupos y se quitan los ceros de la izquierda, que no cambian el valor: 0001 1010 1001 $\rightarrow$ **110101001**₍₂₎.

De binario a octal: grupos de 3 bits. Como $8 = 2^3$, se reagrupan esos mismos bits de 3 en 3 desde la derecha:

Grupo de 3 bits	110	101	001
Valor (pesos 4, 2, 1)	$4 + 2 + 0 = 6$	$4 + 0 + 1 = 5$	$0 + 0 + 1 = 1$

Resultado: **651**₍₈₎. Comprobación: $6 \times 64 + 5 \times 8 + 1 \times 1 = 384 + 40 + 1 = 425$.

3) [2 Puntos] Unidades de información y códigos

- a) Una carpeta de vídeos ocupa 16 GB. Expresa su tamaño en MB y en Mb, como potencias de 2. ¿Por qué salen números distintos?

Solución. Las unidades, como potencias de 2. En informática, cada unidad es $2^{10} = 1024$ veces la anterior, porque 1024 es la potencia de 2 más cercana a 1000:

1 KB	$= 2^{10}$ bytes	$= 1024$ bytes
1 MB	$= 2^{10}$ KB $= 2^{20}$ bytes	$= 1024$ KB
1 GB	$= 2^{10}$ MB $= 2^{30}$ bytes	$= 1024$ MB

Trabajar con potencias de 2 simplifica las cuentas: para multiplicar potencias de la misma base se *suman* los exponentes, y para dividir se *restan*.

En MB. El 16 también es una potencia de 2: $16 = 2^4$. Para pasar de GB a MB se multiplica por $1024 = 2^{10}$:

$$16 \text{ GB} = 2^4 \text{ GB} = 2^4 \times 2^{10} \text{ MB} = 2^{4+10} \text{ MB} = \mathbf{2^{14} \text{ MB}}$$

$$2^{14} \text{ MB} = 16 \times 1024 \text{ MB} = 16.384 \text{ MB}$$

En Mb. Un byte son 8 bits, y $8 = 2^3$. Para pasar de megabytes a megabits se multiplica por 8:

$$2^{14} \text{ MB} = 2^{14} \times 2^3 \text{ Mb} = 2^{14+3} \text{ Mb} = \mathbf{2^{17} \text{ Mb}}$$

$$2^{17} \text{ Mb} = 16.384 \times 8 \text{ Mb} = 131.072 \text{ Mb}$$

Por qué salen números distintos. Porque son unidades distintas: **MB** (con B mayúscula) son *megabytes* y **Mb** (con b minúscula) son *megabits*. Un byte son 8 bits, así que el mismo tamaño expresado en megabits es un número 8 veces mayor. Por eso conviene fijarse en la unidad: las velocidades de conexión a Internet, por ejemplo, se dan en megabits por segundo (Mb/s), no en megabytes.

b) Si cada vídeo de esa carpeta ocupa 256 MB, ¿cuántos vídeos hay en ella?

Solución. Si todos los vídeos ocupan lo mismo, el número de vídeos es el tamaño total dividido entre lo que ocupa cada uno, con los dos tamaños en la misma unidad (MB). El tamaño de cada vídeo también es una potencia de 2: $256 = 2^8$. Para dividir potencias de la misma base se restan los exponentes:

$$\frac{2^{14} \text{ MB}}{2^8 \text{ MB}} = 2^{14-8} = 2^6 = \mathbf{64 \text{ vídeos.}}$$

Comprobación en decimal: $16.384/256 = 64$, y al revés, $64 \times 256 = 16.384$.

c) Un código de longitud fija tiene que representar las 27 letras del alfabeto español en mayúsculas, los diez dígitos, el espacio en blanco y cinco signos de puntuación (punto, coma, punto y coma, dos puntos y guion). ¿Cuántos bits por carácter necesita como mínimo? ¿Cuánto ocupa con ese código un texto de 8192 caracteres, en bytes y en KB? ¿Y con ASCII extendido, de 8 bits? ¿Por qué los ordenadores no usan un código propio como este, sino ASCII o Unicode?

Solución. Bits por carácter. Primero se cuentan los caracteres que hay que representar:

$$27 \text{ letras} + 10 \text{ dígitos} + 1 \text{ espacio} + 5 \text{ signos} = 43 \text{ caracteres.}$$

Cada carácter necesita una combinación de bits distinta, y con n bits hay 2^n combinaciones. Se busca el menor n tal que $2^n \geq 43$:

Bits (n)	Combinaciones (2^n)	¿Llegan para 43 caracteres?
4	$2^4 = 16$	no
5	$2^5 = 32$	no
6	$2^6 = 64$	sí, y sobran $64 - 43 = 21$

Hacen falta como mínimo **6 bits por carácter**. Al ser un código de longitud fija, todos los caracteres ocupan esos 6 bits.

Tamaño del texto con el código de 6 bits.

$$\text{en bits} = 8192 \text{ caracteres} \times 6 \text{ bits} = 49.152 \text{ bits}$$

$$\text{en bytes} = 49.152/8 = \mathbf{6144 \text{ bytes}} \quad (1 \text{ byte} = 8 \text{ bits})$$

$$\text{en KB} = 6144/1024 = \mathbf{6 \text{ KB}} \quad (1 \text{ KB} = 1024 \text{ bytes})$$

Con potencias de 2 sale lo mismo: $8192 = 2^{13}$, así que el texto ocupa $6 \times 2^{13} \text{ bits} = 6 \times 2^{13}/2^3 = 6 \times 2^{10} \text{ bytes} = 6 \text{ KB}$.

Con ASCII extendido (8 bits). Cada carácter ocupa 8 bits, es decir, 1 byte:

$$8192 \times 1 \text{ byte} = 8192 \text{ bytes} = 2^{13} \text{ bytes} = 2^{13-10} \text{ KB} = 2^3 \text{ KB} = \mathbf{8 \text{ KB.}}$$

El código propio ahorra $8 - 6 = 2 \text{ KB}$, un 25 % ($2/8 = 0,25$), porque usa 6 bits por carácter en lugar de 8.

Por qué no se usa un código propio.

- No es un estándar: cualquier otro ordenador o programa interpretaría mal esas combinaciones de bits, y no se podría intercambiar información.
- Es muy limitado: no tiene minúsculas, tildes ni muchos otros símbolos.
- Sus caracteres de 6 bits no encajan con el byte, que es la unidad en que se organiza la memoria.

ASCII y Unicode son estándares que comparten todos los sistemas, y Unicode, además, recoge los caracteres de todos los idiomas. El pequeño ahorro de espacio no compensa la falta de compatibilidad.

4) [2 Puntos] **Direcciones, capacidad y latencia de la memoria**

- a) Un procesador tiene un bus de direcciones de 26 líneas, y la memoria se direcciona byte a byte. ¿Cuántas posiciones puede direccionar? ¿Cuántos MB son? ¿Cuál es la última dirección, en hexadecimal?

Solución. Posiciones. Cada línea del bus de direcciones lleva un bit (0 o 1). Con n líneas se pueden formar 2^n combinaciones distintas, es decir, 2^n direcciones. Con 26 líneas:

$$2^{26} = 2^{20} \times 2^6 = 1.048.576 \times 64 = \mathbf{67.108.864 \text{ posiciones.}}$$

Como la memoria se direcciona byte a byte, cada posición es un byte.

En MB. 1 MB son 2^{20} bytes. Para dividir potencias de la misma base se restan los exponentes:

$$\frac{2^{26} \text{ bytes}}{2^{20} \text{ bytes/MB}} = 2^{26-20} = 2^6 = \mathbf{64 \text{ MB.}}$$

Comprobación: $67.108.864/1.048.576 = 64$.

Última dirección. Las direcciones van de 0 a $2^{26} - 1$: como la primera es la 0, la última es una menos que el número de posiciones (igual que en decimal, con tres cifras se llega hasta $10^3 - 1 = 999$). En binario, $2^{26} - 1$ son 26 bits todos a 1. Para pasarlo a hexadecimal se agrupan de 4 en 4 desde la derecha. Como $26 = 6 \times 4 + 2$, salen seis grupos de 1111 y un grupo de 11 a la izquierda:

Grupo	11	1111	1111	1111	1111	1111	1111
Valor	$2 + 1 = 3$	15	15	15	15	15	15
Cifra hexadecimal	3	F	F	F	F	F	F

Resultado: **3FFFFFFF**₍₁₆₎ (en decimal, 67.108.863).

- b) Una memoria de 32 KB se direcciona byte a byte. ¿Cuántas líneas de dirección necesita como mínimo? ¿Cuál es su última dirección, en decimal y en hexadecimal?

Solución. Líneas de dirección. Cada byte necesita su propia dirección, así que hacen falta tantas direcciones como bytes:

$$32 \text{ KB} = 32 \times 1024 \text{ bytes} = 2^5 \times 2^{10} \text{ bytes} = 2^{15} \text{ bytes} = 32.768 \text{ bytes.}$$

Con n líneas hay 2^n direcciones, y se necesitan 2^{15} : con 15 líneas hay justo $2^{15} = 32.768$ direcciones, y con 14 solo habría $2^{14} = 16.384$, la mitad, que no llegan. Como mínimo, **15 líneas**.

Última dirección. Se empieza a contar en 0, así que la última es una menos que el número de direcciones:

$$2^{15} - 1 = 32.768 - 1 = \mathbf{32.767.}$$

En binario son 15 unos. Se agrupan de 4 en 4 desde la derecha: como $15 = 3 \times 4 + 3$, salen tres grupos de 1111 y un grupo de 111 a la izquierda:

Grupo	111	1111	1111	1111
Valor	$4 + 2 + 1 = 7$	15	15	15
Cifra hexadecimal	7	F	F	F

Resultado: **7FFF**₍₁₆₎.

- c) Otra unidad de memoria tiene 13 líneas de dirección y guarda palabras de 32 bits. ¿Cuántas palabras guarda? ¿Cuál es su capacidad, en bits y en KB? Compárala con la de la memoria del apartado b): ¿por qué necesita menos líneas de dirección?

Solución. Palabras. Con 13 líneas hay 2^{13} direcciones, y aquí cada dirección corresponde a una palabra:

$$2^{13} = 2^{10} \times 2^3 = 1024 \times 8 = \mathbf{8192 \text{ palabras.}}$$

Capacidad. Es el número de palabras multiplicado por el tamaño de cada una:

$$\text{en bits} = 8192 \text{ palabras} \times 32 \text{ bits} = \mathbf{262.144 \text{ bits}}$$

$$\text{en bytes} = 262.144/8 = 32.768 \text{ bytes}$$

$$\text{en KB} = 32.768/1024 = \mathbf{32 \text{ KB}}$$

Con potencias de 2 sale lo mismo: $2^{13} \times 2^5 = 2^{18}$ bits; $2^{18}/2^3 = 2^{15}$ bytes; $2^{15}/2^{10} = 2^5 = 32 \text{ KB}$.

Comparación con la memoria del apartado b). Las dos tienen la **misma capacidad**, 32 KB, pero esta usa 13 líneas de dirección en lugar de 15. La diferencia está en lo que hay en cada dirección:

- En b), cada dirección corresponde a 1 byte: hacen falta $32.768 = 2^{15}$ direcciones.

- Aquí, cada dirección corresponde a una palabra de 32 bits, es decir, de 4 bytes: hace falta la cuarta parte de direcciones, $32.768/4 = 8192 = 2^{13}$.

Como $4 = 2^2$, y cada línea de dirección que se quita divide entre 2 el número de direcciones, bastan 2 líneas menos: $15 - 2 = 13$. A cambio, necesita un bus de datos de 32 líneas en lugar de 8, para transportar la palabra entera de una vez.

- d) Compara dos módulos de memoria: uno DDR4-2400 con una latencia CL de 18 y otro DDR5-6400 con una CL de 32. El número que sigue a DDR4 o DDR5 es su velocidad en MT/s, millones de transferencias por segundo. Pasa las dos latencias a nanosegundos con la fórmula

$$\text{latencia (ns)} = \frac{\text{CL} \times 2000}{\text{velocidad en MT/s}}$$

¿Cuál responde antes? ¿Cuál transfiere más datos por segundo? ¿A qué frecuencia va el reloj de cada uno?

Solución. Latencias. Se sustituyen en la fórmula el CL y la velocidad de cada módulo:

$$\begin{aligned} \text{DDR4-2400: } \frac{18 \times 2000}{2400} &= \frac{36.000}{2400} = \mathbf{15 \text{ ns}} \\ \text{DDR5-6400: } \frac{32 \times 2000}{6400} &= \frac{64.000}{6400} = \mathbf{10 \text{ ns}} \end{aligned}$$

Responde antes el DDR5 (10 ns frente a 15 ns), aunque su CL sea mayor. El CL no es un tiempo, sino un número de ciclos de reloj, y los ciclos del DDR5 son mucho más cortos. Para comparar hay que pasar a tiempo real, que es lo que hace la fórmula.

Transfiere más datos el DDR5: 6400 millones de transferencias por segundo, frente a 2400. Como cada módulo transfiere 64 bits (8 bytes) en cada transferencia:

$$\begin{aligned} \text{DDR4-2400: } 2400 \times 8 &= 19.200 \text{ MB/s} \\ \text{DDR5-6400: } 6400 \times 8 &= 51.200 \text{ MB/s} \end{aligned}$$

Reloj. Las memorias DDR (*Double Data Rate*) hacen dos transferencias en cada ciclo de reloj: una en el flanco de subida y otra en el de bajada. Por eso el reloj va a la mitad de las transferencias por segundo:

$$\begin{aligned} \text{DDR4-2400: } 2400/2 &= \mathbf{1200 \text{ MHz}} \\ \text{DDR5-6400: } 6400/2 &= \mathbf{3200 \text{ MHz}} \end{aligned}$$

De ahí sale la fórmula: un reloj de f MHz hace f millones de ciclos por segundo, así que cada ciclo dura $1000/f$ ns. Como f es la mitad de los MT/s, cada ciclo dura $1000/(MT/s/2) = 2000/MT/s$ ns, y la latencia es CL veces ese tiempo. Por ejemplo, en el DDR5-6400 cada ciclo dura $1000/3200 = 0,3125$ ns, y $32 \times 0,3125 = 10$ ns.

5) [2 Puntos] El reloj del procesador

Un procesador funciona con un reloj base de 200 MHz y un multiplicador de 20. La memoria principal tiene un tiempo de acceso de 15 ns, y la caché L1, de 0,75 ns.

- a) ¿A qué frecuencia trabaja el procesador? ¿Cuánto dura un ciclo de reloj?

Solución. Frecuencia. La placa base da al procesador un reloj base, y el procesador lo multiplica internamente para obtener su frecuencia de trabajo:

$$\text{frecuencia} = \text{reloj base} \times \text{multiplicador} = 200 \text{ MHz} \times 20 = 4000 \text{ MHz} = \mathbf{4 \text{ GHz}}.$$

(1 GHz = 1000 MHz = 10^9 ciclos por segundo.)

Duración de un ciclo. La frecuencia es el número de ciclos por segundo, así que un ciclo dura la inversa de la frecuencia:

$$T = \frac{1}{f} = \frac{1}{4 \times 10^9 \text{ ciclos/s}} = 0,25 \times 10^{-9} \text{ s} = \mathbf{0,25 \text{ ns}}.$$

(1 ns = 10^{-9} s, una milmillonésima de segundo.) Comprobación: $4 \times 10^9 \text{ ciclos} \times 0,25 \times 10^{-9} \text{ s por ciclo} = 1 \text{ s}$.

- b) ¿Cuántos ciclos de espera necesita para leer un dato de la memoria principal? ¿Y de la caché L1?

Solución. El procesador lo mide todo en ciclos de reloj, así que hay que ver cuántos ciclos caben en el tiempo de acceso: se divide el tiempo de acceso entre la duración de un ciclo. Si no saliera exacto, se redondearía hacia arriba, porque el procesador solo puede recoger el dato al final de un ciclo completo.

$$\text{Memoria principal: } \frac{15 \text{ ns}}{0,25 \text{ ns por ciclo}} = \mathbf{60 \text{ ciclos}}$$

$$\text{Caché L1: } \frac{0,75 \text{ ns}}{0,25 \text{ ns por ciclo}} = \mathbf{3 \text{ ciclos}}$$

Leer de la caché es $60/3 = 20$ veces más rápido: por eso es tan importante que los datos más usados estén en ella.

- c) Un programa tiene 3000 millones de instrucciones, con un CPI medio (ciclos por instrucción) de 2 que ya tiene en cuenta las esperas a la memoria. ¿Cuánto tarda en ejecutarse?

Solución. El CPI indica cuántos ciclos necesita, de media, cada instrucción. Primero se calculan los ciclos totales y después se pasan a tiempo.

$$\text{ciclos totales} = \text{instrucciones} \times \text{CPI} = 3000 \times 10^6 \times 2 = 6000 \times 10^6 = 6 \times 10^9 \text{ ciclos}$$

$$\text{tiempo} = \text{ciclos} \times \text{duración del ciclo} = 6 \times 10^9 \times 0,25 \times 10^{-9} \text{ s} = 6 \times 0,25 \text{ s} = \mathbf{1,5 \text{ s}}$$

(En el último paso, $10^9 \times 10^{-9} = 1$.) Es lo mismo que dividir los ciclos entre la frecuencia, que son los ciclos por segundo:

$$\frac{6 \times 10^9 \text{ ciclos}}{4 \times 10^9 \text{ ciclos/s}} = 1,5 \text{ s.}$$

- d) El usuario sube el multiplicador a 25. ¿A qué frecuencia trabaja ahora el procesador? Si el CPI no cambiara, ¿cuánto tardaría el programa? ¿Por qué, en la práctica, tarda algo más? ¿Qué inconvenientes tiene hacer esto?

Solución. Nueva frecuencia y nueva duración del ciclo:

$$f = 200 \text{ MHz} \times 25 = 5000 \text{ MHz} = \mathbf{5 \text{ GHz}}$$

$$T = \frac{1}{f} = \frac{1}{5 \times 10^9 \text{ ciclos/s}} = 0,2 \times 10^{-9} \text{ s} = 0,2 \text{ ns}$$

Con el mismo CPI. Los ciclos totales no cambian (6×10^9), pero cada ciclo es más corto:

$$6 \times 10^9 \times 0,2 \times 10^{-9} \text{ s} = \mathbf{1,2 \text{ s}} \quad \left(\text{o bien } \frac{6 \times 10^9}{5 \times 10^9} = 1,2 \text{ s} \right).$$

Por qué en la práctica tarda algo más. Solo se ha acelerado el procesador, no la memoria, que sigue tardando lo mismo. Medidas en ciclos, las esperas ahora son más largas:

$$\text{Memoria principal: } 15/0,2 = 75 \text{ ciclos (antes, 60)}$$

$$\text{Caché L1: } 0,75/0,2 = 3,75 \rightarrow 4 \text{ ciclos, redondeando hacia arriba (antes, 3)}$$

Como el CPI incluía esas esperas, sube por encima de 2, y el tiempo queda por encima de 1,2 s. Es el embotellamiento de Von Neumann: la memoria no sigue el ritmo del procesador.

Inconvenientes (*overclocking*).

- Consume más y se calienta más, así que necesita mejor refrigeración.
- Puede volverse inestable (bloqueos, errores, reinicios), porque sus circuitos tienen que completar cada operación en un ciclo más corto.
- Se acorta su vida útil y se puede perder la garantía.
- No todos los procesadores y placas base permiten cambiar el multiplicador.