COGNOMS:

NOM: DNI:

IMPORTANTE leer atentamente antes de empezar el examen: Escriba los apellidos, el nombre y el DNI/NIE antes de empezar el examen. Escriba un solo carácter por recuadro, en mayúsculas y lo más claramente posible. Es importante que no haya tachones ni borrones y que cada carácter quede enmarcado dentro de su recuadro sin llegar a tocar los bordes. Use un único cuadro en blanco para separar los apellidos y nombres compuestos si es el caso. No escriba fuera de los recuadros, todo lo que haya fuera de ellos es ignorado. La identificación del alumno se realiza de forma automática, no seguir correctamente estas instrucciones puede comportar no tener nota.

Problema 1. (3 puntos)

Un procesador RISC tiene instrucciones de tamaño fijo de 32bits. Este procesador incorpora una cache de instrucciones y una cache de datos de 16KB cada una. Ambas caches tienen 4 vías y un tamaño de bloque de cache de 32bytes para la de instrucciones y de 16bytes para la de datos. La frecuencia del procesador es 1GHz. En dicho procesador el CPI ideal es de 1 ciclo/instr cuando todos los accesos son acierto de cache.

Se ejecuta un código C que ocupa 40000 bytes (10000 instrucciones) y es completamente secuencial (no incluye instrucciones de saltos ni llamadas a subrutinas). Al analizar el código observamos que el 30% de las instrucciones realizan un acceso a datos y que todos los accesos son lecturas de tamaño 4 bytes. El tiempo de acceso a la cache es de 1 ciclo y el tiempo de penalización por fallo Tpf son 10 ciclos, tanto para datos como para instrucciones.

a) **Calcula** el número de conjuntos de la cache de instrucciones y de datos.

b) **Calcula** la tasa de fallos para las instrucciones del código C.

Los accesos a datos son siempre a datos no accedidos previamente, por lo que no hay localidad temporal. Ademas, los datos accedidos están distribuidos de forma que tampoco es posible aprovechar ningún tipo de localidad espacial. Dado que no hay ningún tipo de reuso, todos los accesos a datos serán fallo.

c) **Calcula** el CPI al ejecutar el código C.

Al medir la energía de conmutación consumida por los accesos a memoria de nuestro procesador obtenemos 1nJ por cada acceso a la cache (datos o instrucciones) y 10nJ extra por cada fallo de cache.

d) **Calcula** la energía de conmutación consumida por la jerarquía de memoria al ejecutar el código C.

e) **Calcula** la potencia media de conmutación de la jerarquía de memoria .

Hemos recompilado el código C activando una optimización que reordena los datos para mejorar la localidad. Para el caso del código C los ha podido reorganizar de forma que se explote el máximo de localidad espacial. Es decir siempre que se lleve un bloque a cache se usarán todos los datos del bloque antes de ser expulsado de la misma.

f) **Calcula** el speed-up (ganancia de tiempo de ejecución) que obtendremos con esta optimización.

COGNOMS:

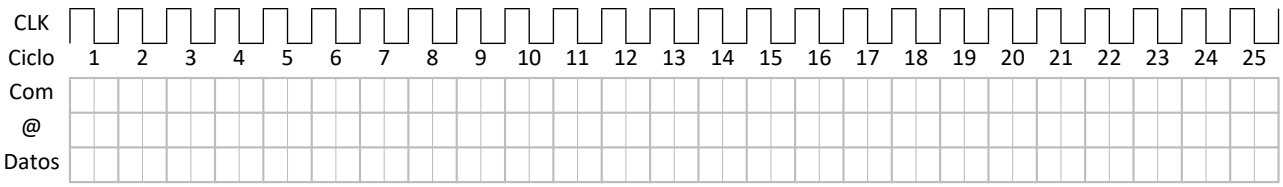
NOM: DNI:

Problema 2. (3,6 puntos)

Una **CPU** está conectada a una cache de instrucciones (**$I**) y una cache de datos (**$D**). El conjunto formado por **CPU**+**$I**+**$D** esta conectado a una memoria principal formada por un único módulo DIMM estándar de 8 GBytes. Este DIMM tiene 8 chips de memoria **DDR**-SDRAM (**Double Data Rate** Synchronous DRAM) de 1 byte de ancho cada uno. El DIMM esta configurado para leer/escribir ráfagas de 64 bytes (justo el tamaño de bloque de las caches). La latencia de fila es de 4 ciclos, la latencia de columna de 3 ciclos y la latencia de precarga de 1 ciclo.

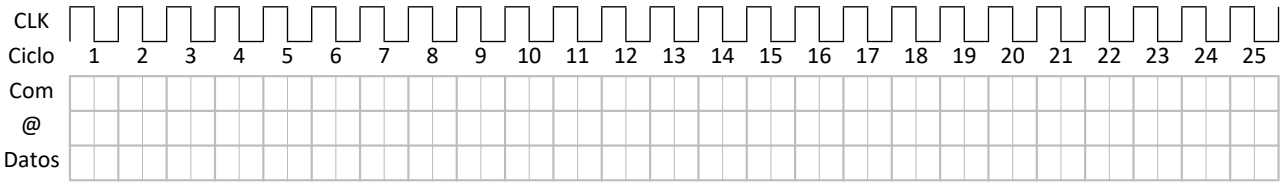
En los siguientes cronogramas, indica la ocupación de los distintos recursos de la memoria DDR: bus de datos, bus de direcciones y bus de comandos. En todos los cronogramas supondremos que no hay ninguna página de DRAM abierta.

a) **Rellena** el siguiente cronograma para una lectura de un bloque de 64 bytes de la DDR.

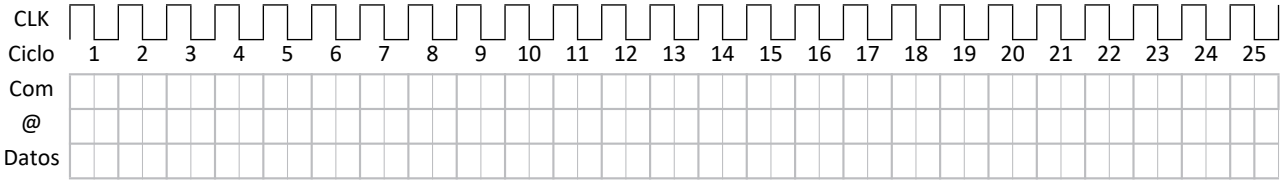


En ocasiones, es posible que el conjunto **CPU**+**$I**+**$D** solicite múltiples bloques a la DDR (por ejemplo porque se produzca un fallo simultáneamente en **$I** y en **$D**). El controlador de memoria envía los comandos necesarios a la DDR-SDRAM de forma que ambos bloques sean transferidos lo más rápidamente posible y se maximice el ancho de banda. Rellena los siguientes cronogramas para la lectura de dos bloques de 64 bytes en función de la ubicación de los dos bloques involucrados. El objetivo es minimizar el tiempo total.

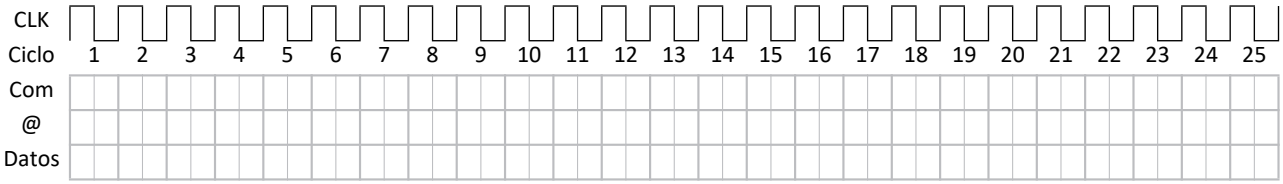
b) Ambos bloques están ubicados en el mismo banco y en la misma página .



c) Ambos bloques están ubicados en el mismo banco pero en páginas distintas.



d) Ambos bloques están ubicados en bancos distintos.



El conjunto **CPU**+**$I**+**$D** funciona a una frecuencia interna mayor que la memoria SDRAM. Un ciclo de la SDRAM corresponde a múltiples ciclos de CPU por lo que los ciclos de los apartados siguientes no se corresponden a los cronogramas anteriores.

Un programa P realiza $5x10^9$ accesos a datos, todos de 8 bytes. Sabemos que **$D** tiene bloques de 64 bytes y políticas de escritura **copy back** + **write allocate**. Hemos medido que, durante la ejecución de P, **$D** tiene una tasa de fallos del 10% y que el 25% de los bloques reemplazados tenían el *dirty bit* a 1.

e) **Calcula** cuantos bytes lee **$D** desde la **DDR** y cuantos bytes escribe **$D** en la **DDR**.

Dado el siguiente fragmento de código:

```
for (i=0; i<N; i++)
  suma = suma + v[i]; // v[i] es un vector de doubles (8 bytes)
```

El código está almacenado en **$I**, las variables i, N y suma están en registros y **$D** está inicialmente vacía. Los elementos del vector v son de 8 bytes y los bloques de **$D** son de 64 bytes. La capacidad de **$D** es de 8 Kbytes.

Hemos ejecutado 2 veces consecutivas el mismo fragmento de código (para **N = 1000**) y hemos medido los ciclos de CPU de ambas ejecuciones:

- En la 1a ejecución el bucle tarda 55.000 ciclos.
- En la 2a ejecución el bucle tarda 30.000 ciclos.

f) **Calcula** el tiempo de penalización medio (en ciclos) en caso de fallo en **$D**.

Deseamos ejecutar una sola copia del mismo fragmento de código para N muy muy grande (el vector recorrido es mucho mayor que el tamaño de cache).

g) **Calcula** en función de N los ciclos que tarda el fragmento de código anterior.

A la cache **$D** le añadimos un mecanismo de *prefetch* hardware. Cuando se accede un bloque (i) se desencadena *prefetch* del bloque siguiente (i+1) siempre que el bloque (i+1) no se encuentre ya en la cache o no haya un *prefecth* previo del bloque (i+1) pendiente de completar (en ambos casos es innecesario hacer prefecth de nuevo).

h) **Calcula** el número máximo de ciclos que puede durar un *prefetch* para que el bucle se ejecute en 40*N ciclos.

i) ¿Es posible ejecutar el bucle en menos de 40*N haciendo el *prefetch* más rápido? (justifica la respuesta)

COGNOMS:

NOM: DNI:

Problema 3. (3,4 puntos)

Disponemos de 8 discos físicos de 1 Tbyte de capacidad por disco, que ofrecen un ancho de banda efectivo de 200 Mbytes/s por disco. Con estos discos deseamos montar un sistema de almacenamiento en RAID 5.

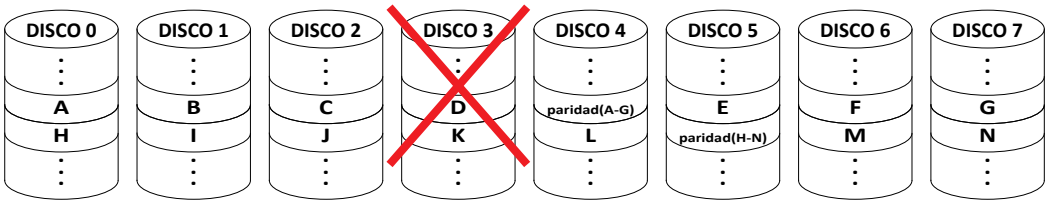
a) **Calcula** la cantidad de información útil (datos) que puede almacenar el sistema RAID5

b) **Dibuja** un esquema y **describe** cómo se hace una escritura aleatoria de 1 bloque de disco en el disco 3 del RAID 5.

El controlador del RAID, distribuye las peticiones aleatorias uniformemente entre todos los discos del sistema.

c) **Calcula** el ancho de banda efectivo cuando realizamos lecturas aleatorias y cuando realizamos escrituras aleatorias.

En la siguiente figura mostramos de forma parcial el RAID5, con el contenido de los bloques A, B, ... G i H, I, ... M, N, así como su paridad, que para este caso estaría almacenada en los discos 4 y 5.



d) Suponiendo que el DISCO 3, ha dejado de funcionar, ¿qué pasos debe hacer el sistema RAID para obtener el bloque D?.

e) Con los 8 discos del RAID 5 se quiere montar un RAID 51**. Dibuja** este esquema e **indica** el ancho de banda efectivo (datos útiles) máximo de lectura del RAID 51.

f) **Indica** el ancho de banda efectivo **máximo** de escrituras aleatorias que puede conseguirse en el esquema RAID 51 del apartado e) y **justifica** tu respuesta.

Cada uno de los RAIDs 5 del apartado e) tiene una fuente de alimentación con un MTTF de 200.000 horas y un MTTR de 15 horas. Suponiendo que los discos nunca fallarán y teniendo en cuenta exclusivamente las fuentes de alimentación.

g) **Calcula** el MTTF del RAID 51.

Disponemos de una aplicación A que tiene dos fases de ejecución (lectura y cálculo) y se ejecuta en un solo procesador P con un único disco duro D en un tiempo T. La fase de lectura supone el 30% del tiempo de ejecución, y el 80% de la fase de cálculo es paralelizable. Ejecutamos la aplicación en un sistema multiprocesador MP que tiene 4 procesadores P y un RAID cuyo ancho de banda efectivo de lectura es 10 veces el del disco duro D. Suponiendo que la paralelización se hace de forma perfecta.

h) **Calcula** el speedup al ejecutar la aplicación A en el sistema MP.