COGNOMS:

NOM: DNI/NIE:

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. (3,4 puntos)

Tenemos una CPU (C1) que tiene un tiempo de ciclo (Tc) de 1 ns. Al ejecutar un programa P (que ejecuta $40x10^9$ instrucciones) en un simulador de C1 donde todos los accesos a memoria hacen *hit* en la Cache de Instrucciones (I$) y Datos (D$) tarda $60x10^9$ ciclos.

a) **Calcula** el CPI ideal (CPIideal) y el tiempo de ejecución en segundos (Texec) del programa P en este sistema de memoria ideal.

Medimos el número medio de referencias por instrucción (nr) y vemos 1.5refs/instrucción repartidas de la siguiente manera (1) 1.00 referencia por instrucción a instrucciones y (2) 0.5 referencias por instrucción a datos.

Con una I$ i una D$ reales tenemos un *miss rate* en D$ del 11% y en I$ del 5%.

En caso de acierto en la I$ y en la D$ el tiempo de servicio es de 1 ciclo. En caso de *miss* en la I$ o en la D$ el tiempo de penalización por acceder a la memoria principal es de 100 ciclos.

La D$ tiene una política de escritura *Copy Back* y *Write Allocate*, aunque en el programa P el número de bloques modificados es negligible.

b) **Calcula** el tiempo medio de acceso a memoria en ciclos para los accesos a instrucciones (TmaI)

c) **Calcula** el tiempo medio de acceso a memoria en ciclos para los accesos a datos (TmaD)

d) **Calcula** el tiempo medio de acceso a memoria en ciclos para todos los accesos (Tma)

e) **Calcula** el tiempo de ejecución del programa P en la CPU C1 con caches I$ i D$ reales (TexeR1)

Para mejorar el rendimiento del programa P diseñamos una nueva CPU (C2) a partir de la CPU C1 descrita anteriormente añadiéndole una cache Unificada de segundo nivel (L2$). El tiempo de ejecución del programa P en C2 es de 218 s y el *miss rate* local de la L2$ para el programa P es del 30%. El tamaño de bloque (línea) de todas las caches es de 64 bytes. Los accesos a la I$ son siempre de 4 bytes (el tamaño de las instrucciones), los accesos a la D$ son siempre de 8 bytes.

f) **Calcula** el número de accesos, el número de bytes que se leen y el ancho de banda (en MBytes/segundo) en todos los elementos de la jerarquía: I$, D$, L2$ y Memoria Principal (MP), que realiza el programa P en la CPU C2. Justifica las respuestas.

	Accesos	**Bytes leídos**	**Ancho de banda**
I$			
D$			
L2$			
MP			

Un acceso a I$ requiere una energía dinámica (de conmutación) de 1 nJ, a D$ de 1,5 nJ, a L2$ de 5 nJ y a MP de 50 nJ.

g) **Calcula** la energia total y la potencia dinámica media consumida por la jerarquía de memoria de C2 durante la ejecución de P.

La cache L2$ tiene un tiempo de servicio en caso de acierto de 10 ciclos y una penalización en caso de *miss* de 100 ciclos.

h) **Calcula** el mínimo *hit rate* local (h) que debería tener la L2$ para que un programa se ejecute más rápidamente en la CPU C2 que en la C1

COGNOMS:

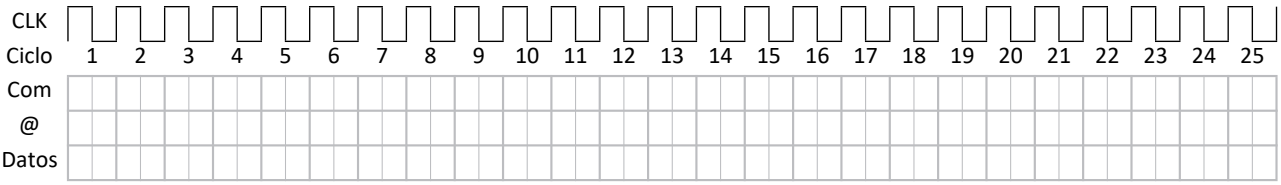
NOM: DNI/NIE:

Problema 2. (3,4 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 16 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.

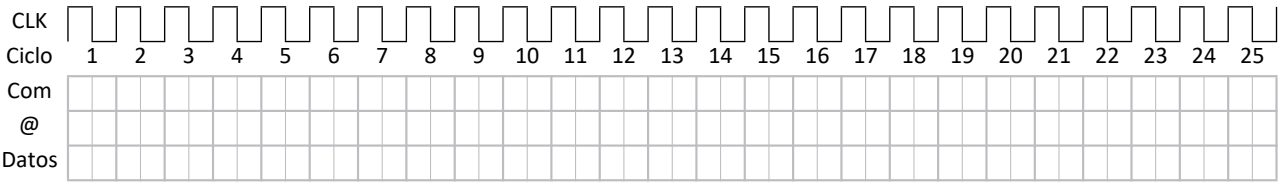
En los siguientes cronogramas, indica la ocupación de los distintos recursos de la memoria DDR: bus de datos, bus de direcciones y bus de comandos. En todos los cronogramas supondremos que no hay ninguna página de DRAM abierta.

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

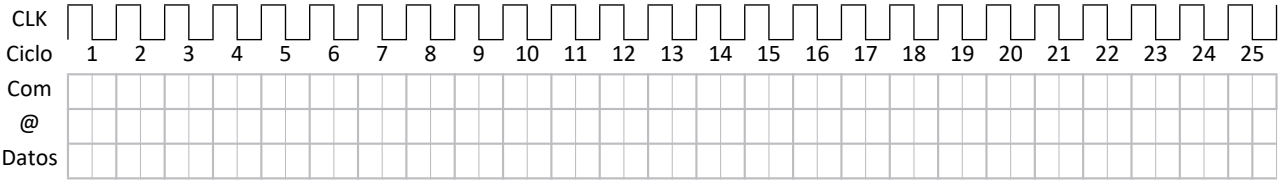


En ocasiones, es posible que el conjunto **CPU**+**$I**+**$D** solicite múltiples bloques a la DDR (por ejemplo porque se produzca un fallo simultáneamente en **$I** y en **$D**). El controlador de memoria envía los comandos necesarios a la DDR-SDRAM de forma que ambos bloques sean transferidos lo más 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.

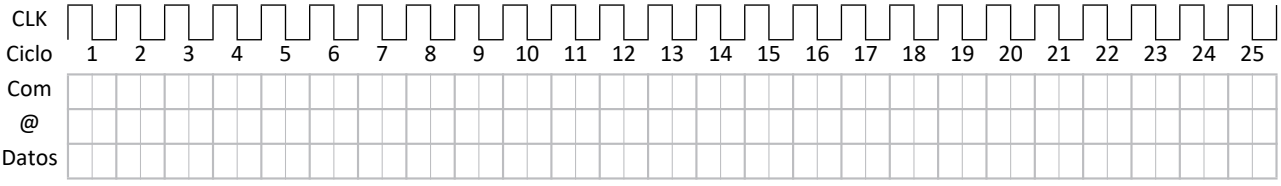
b) Ambos bloques están ubicados en bancos distintos.



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



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



El conjunto **CPU**+**$I**+**$D** funciona a una frecuencia interna mayor que la de la memoria SDRAM. Un ciclo de la SDRAM corresponde a múltiples ciclos del conjunto **CPU**+**$I**+**$D** 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 4 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 floats (4 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 4 bytes y los bloques de **$D** son de 64 bytes. La capacidad de **$D** es de 8 Kbytes.

En un programa de prueba hemos ejecutado 2 veces consecutivas el mismo fragmento de código (para **N = 2000**) y hemos medido los ciclos de CPU de ambas ejecuciones:

- En la 1a ejecución el bucle tarda 70.000 ciclos.
- En la 2a ejecución el bucle tarda 40.000 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 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 25*N ciclos.

i) ¿Es posible ejecutar el bucle en menos de 25*N haciendo el *prefetch* más rápido? (justifica la respuesta)

COGNOMS:

NOM: DNI/NIE:

Problema 3. (3,2 puntos)

Disponemos de un disco duro con 8 superficies (4 platos de 2 caras por cada plato) y un radio útil para almacenar información de 2 cm por superficie. El brazo móvil permite una distancia entre pistas (incluyendo el grosor de las mismas) de 50 μm. El total de información "bruta" que puede almacenar una pista es de 94248 bits. Los sectores del disco almacenan 357 bytes de información "bruta", de los cuales 101 bytes corresponden a información de control. La velocidad de rotación (número de vueltas por minuto) es de 7200 r.p.m.

a) **Calcula** cuántos sectores hay por pista. **Calcula** también la capacidad "neta" del disco en bytes. **Escribe** el resultado en MB (potencias de 10).

b) **Calcula** la velocidad máxima de transferencia de información "bruta" del disco. **Calcula** también el tiempo de transferencia de un sector.

Sea una posible configuración RAID 5 con 10 discos. El tiempo medio entre fallos de un disco ($MTTF_d$) es de 60000 horas. El tiempo invertido en cambiar un disco averiado y reconstruir la información (MTTR) es de 30 horas. Recuerda que en una configuración RAID 5 el sistema deja de funcionar cuando falla un segundo disco durante el tiempo de recuperación del disco averiado (MTTR).

c) **Escribe** la expresión general del tiempo medio entre fallos para el RAID 5 anterior ($MTTF_{RAID}$) y **calcula** su valor con los datos proporcionados. Da el resultado en horas.

De acuerdo a un estudio realizado sobre el uso de este dispositivo en un procesador Intel CISC x86 que traduce internamente las instrucciones x86 a microoperaciones (uops), se ha determinado que la rutina de servicio que se ejecuta en cada interrupción de este dispositivo precisa de 3600 uops. Sabemos que cada instrucción dinámica de lenguaje máquina x86 de la rutina se traduce en una uop y que cada 520 instrucciones CISC se necesitan en media 200 uops adicionales. El procesador funciona a una frecuencia de 2 GHz.

d) **Calcula** el CPI y el tiempo de ejecución de la rutina de servicio de interrupciones para un valor de UPC (uops por ciclo) de 1,1.

Disponemos de un programa P que consta de dos fases de ejecución (una de lectura de disco que supone el 30% del tiempo de ejecución y otra de cálculo la cual es totalmente paralelizable). Cuando P se ejecuta en un sistema con un solo procesador y un solo disco (en un PC de sobremesa) tarda un tiempo de 200 horas. Cuando P se ejecuta en un sistema multiprocesador con 13 procesadores y un RAID 5 de 10 discos (cada procesador y cada disco son iguales a los del PC) tarda un tiempo de 40 horas. Sabemos que el programa P realiza 1 operación de coma flotante por cada 3 instrucciones dinámicas de coma flotante, y que el procesador tiene un rendimiento promedio, para este programa, de 1000 MFLOPS cuando se ejecuta en el PC de sobremesa.

e) **Calcula** cuantas instrucciones dinámicas de coma flotante realiza el programa P en el multiprocesador. **Calcula** también a cuantos MFLOPS se ejecuta el programa P en el multiprocesador.

Sabemos que la potencia consumida por cada procesador es de 90W, y la potencia consumida por cada disco es de 30W. Suponer que estos son los únicos elementos con consumo significativo. Durante la fase de E/S se necesita un único procesador activo para controlar la actividad de los discos.

f) **Calcula** los MFLOPS/W del multiprocesador cuando se consigue apagar completamente los elementos (procesadores y/o discos) que no se utilizan en cada fase.