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. (3.6 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 4 GBytes. Este DIMM tiene 8 chips de memoria SDRAM (Synchronous DRAM) de 1 byte de ancho cada uno. El DIMM esta configurado para leer/escribir ráfagas de 32 bytes (justo el tamaño de bloque de las caches). La latencia de fila es de 3 ciclos, la latencia de columna de 2 ciclos y la latencia de precarga de 1 ciclo.

En los siguientes cronogramas, indica la ocupación de los distintos recursos de la memoria SDRAM: 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 32 bytes de la SDRAM.

CLK																					
Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Com																						
@																						
Datos																						

En ocasiones, es posible que el conjunto **CPU**+**$I**+**$D** solicite múltiples bloques a la SDRAM (por ejemplo porque se produzca un fallo simultáneamente en **$I** y en **$D**). El controlador de memoria envía los comandos necesarios a la SDRAM de forma que ambos bloques sean transferidos lo más rápidamente posible y se maximice el ancho de banda. Rellena los siguientes cronogramas, con el objetivo de minimizar el tiempo total, para la lectura de dos bloques de 32 bytes en función de la ubicación de los dos bloques involucrados.

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

CLK																					
Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Com																						
@																						
Datos																						

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

CLK																					
Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Com																						
@																						
Datos																						

d) Ambos bloques están ubicados en bancos distintos.

CLK																					
Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Com																						
@																						
Datos																						

En esta CPU ejecutamos el siguiente bucle:

```
loop: movb (%ebx), %al
      addl $16, %ebx
      cmpb $46, %al
      jne loop
```

El código ya se encuentra almacenado en **$I**, y **$D** está inicialmente vacía. El registro %ebx vale 0x00080000 y sabemos que el bucle realizará muchas iteraciones. La cache **$D** es de mapeo directo y tiene un tamaño de bloque de 32 bytes. Cada uno de los 8 chips del DIMM está organizado en 32 bancos de 4096 filas x 4096 columnas y las direcciones se mapean al DIMM de la siguiente forma: $\mathbf{b_4b_3b_2b_1b_0f_{11}f_{10}f_9f_8f_7f_6f_5f_4f_3f_2f_1f_0c_{11}c_{10}c_9c_8c_7c_6c_5c_4c_3c_2c_1c_0x_2x_1x_0}$
Los bits de menos peso de la dirección (2-0) seleccionan el chip ($x_2 : x_0$), los bits (14-3) seleccionan la columna ($c_{11} : c_0$), los bits (26-15) seleccionan la fila ($f_{11} : f_0$) y finalmente los bits de mas peso (31-27) seleccionan el banco ($b_4 : b_0$).

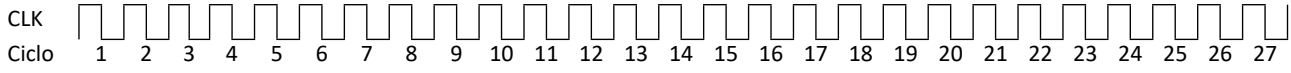
e) **Rellena** la siguiente tabla indicando para cada iteración si la instrucción **movb** producirá acierto (A) o fallo (F) en cache y en caso que se acceda el DIMM indica el banco, la fila y la columna:

Iteración	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**2046**	**2047**	**2048**	**2049**
Cache												
Banco												
Fila												
Columna												

Cada instrucción del código se ejecuta en un solo ciclo a menos que haya un fallo en **$D**. En caso de fallo, el acceso a la SDRAM se inicia en el ciclo siguiente y los datos provenientes de la SDRAM se cargan en **$D** en el mismo ciclo el que aparecen en el bus de datos. La cache tiene continuación anticipada. El controlador de memoria envía los comandos necesarios a la SDRAM de forma que los bloques sean transferidos lo más rápidamente posible y se maximice el ancho de banda con el objetivo de minimizar el tiempo total. El controlador mantiene una página DRAM abierta por un máximo de 8 accesos en la misma página, y la cierra cuando se accede a otra página.

Rellena los siguientes cronogramas con la ejecución de las primeras iteraciones del bucle (las que necesites hasta el ciclo 27) indicando con una cruz (X) el ciclo en que se ejecuta cada una de las instrucciones, o bien con una secuencia de cruces en el caso que se prolongue durante varios ciclos. En la fila cache indicaremos si la instrucción **movb** produce fallo (F) o acierto (A) en **$D**. Finalmente, en las filas Com y Datos indicaremos la ocupación del bus de comandos y de datos de la SDRAM de la misma forma que en los cronogramas anteriores (por simplicidad obviaremos el bus de direcciones).

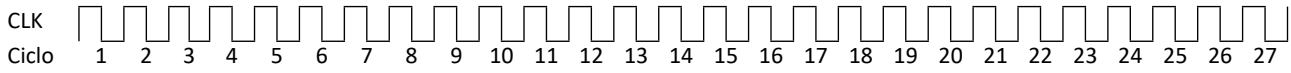
f) **Rellena** el siguiente cronograma teniendo en cuenta que la cache tiene **continuación anticipada**.



Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
movb																											
addl																											
cmpb																											
jne																											
Cache																											
Com																											
Datos																											

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).

g) **Rellena** el siguiente cronograma teniendo en cuenta que la cache tiene **continuación anticipada** + **prefetch**.



Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
movb																											
addl																											
cmpb																											
jne																											
Cache																											
Com																											
Datos																											

COGNOMS:

NOM: DNI/NIE:

Problema 2. (3,2 puntos)

Dado el siguiente código en ensamblador:

```
      movl $0, %esi
for0: movl A(,%esi,4), %eax
      addl $100, %eax
      movl %eax, A(,%esi,4)
      incl %esi
      cmp $200000000,%esi
fi0:  jne for0
      movl $0, %esi
for1: movl A(,%esi,4), %eax
      movl B(,%esi,4), %ebx
      addl %ebx, %eax
      movl %eax, B(,%esi,4)
      incl %esi
      cmp $200000000,%esi
fi1:  jne for1
```

Tenemos un procesador con memoria perfecta (tiempo de acceso de un ciclo), un IPC (Instrucciones Por Ciclo) de 0,4 y una frecuencia de 1GHz. Todos los bucles que aparecen en este problema se ejecutan con este IPC.

a) **Calcula** cuanto tiempo tarda en ejecutarse el primer bucle (for0).

Al procesador del apartado a) le acoplamos un sistema de memoria real con una cache de datos totalmente asociativa que para el primer bucle (for0) tiene una tasa de fallos del 3,125%. Suponemos que la cache de instrucciones siempre acierta. Medimos de nuevo el tiempo de ejecución del primer bucle (for0) y obtenemos 3,9s.

b) **Calcula** la penalización en ciclos por fallo de la cache que hemos incorporado. Si no se puede calcular contestad "INDEFINIDO".

c) **Calcula** las siguientes características de la cache a partir de la tasa de fallos para el primer bucle (for0). Si alguna característica no puede averiguarse escribid "INDEFINIDO".

Tamaño Línea:

Tamaño Cache:

d) **Calcula** el tamaño mínimo de la cache (recuerda que es totalmente asociativa) para que el acceso al vector A del segundo bucle (for1) no tenga ningún fallo de cache en toda la ejecución del segundo bucle (for1).

Cálculo:

Asume una cache de datos de 32KBytes totalmente asociativa.

e) **Calcula** cuanto tiempo tarda en ejecutarse el segundo bucle (for1). Para ello calcula la tasa de fallos de los accesos de este bucle,

Para intentar mejorar el rendimiento se decide utilizar la técnica de "loop fussion". Para ello se modifica el código de la siguiente forma:

```
      movl $0, %esi
for:  movl A(,%esi,4), %eax
      movl B(,%esi,4), %ebx
      addl $100, %eax
      movl %eax, A(,%esi,4)
      addl %eax, %ebx
      movl %ebx, B(,%esi,4)
      incl %esi
      cmp $200000000,%esi
fin:  jne for
```

f) **Calcula** cual es el nuevo tiempo de ejecución del nuevo bucle (for).

g) **Calcula** la mejora entre la primera versión de código y la versión con "loop fussion". Asume que el tiempo de ejecución de la primera versión es la suma del tiempo de los dos bucles (for0 y for1).

COGNOMS:

NOM: DNI/NIE:

Problema 3. (3,2 puntos)

Tenemos que estimar el rendimiento y la fiabilidad de un cluster de computadores orientado al almacenamiento de archivos. El cluster está formado por un cierto número racks. Cada rack tiene 40 nodos computadores, conectados mediante un switch Ethernet con 48 puertos. Asumiremos los siguientes componentes para cada **nodo**:

- 1 Fuente de alimentación y un ventilador.
- 1 procesador x86 que opera a una frecuencia de 1GHz con un rendimiento de 1000MIPS
- 1 DIMM de 512MB formado por 8 chips DDR266 de DRAM
- 4 discos de 500GB cada uno que se pueden acceder de forma independiente. Cada disco posee un tiempo de posicionamiento promedio del cabezal de 8.5ms (time seek), gira a una velocidad de 7200rpm (lo que da una latencia rotacional promedio (rotational delay) de 4.2ms), y posee una velocidad de transferencia de 50MB/s.
- 1 enlace con el switch Ethernet del rack, con un ancho de banda de 1Gbit/s

Al tratarse de un cluster orientado a almacenamiento de datos, utilizaremos como métrica del rendimiento el número de operaciones de entrada-salida por segundo (**iops**), y usaremos como tamaño promedio de una entrada-salida un bloque de archivo de 50KB. Asumiremos lo siguiente:

- que todo archivo está uniformemente distribuido entre los discos y podemos acceder a todos los discos en paralelo.
- que cada entrada-salida al disco requiere un time seek y también un rotational delay.
- que en cada entrada-salida intervienen el sistema operativo y el protocolo de red, ejecutando 50000 y 100000 instrucciones del procesador, respectivamente.

Supondremos también que todos los dispositivos que intervienen en la cadena de acceso al bloque de datos pueden usarse al 100% de su capacidad. En esta cadena distinguimos: el procesador, la memoria, los cuatro discos del nodo, y la red Ethernet que conecta al nodo con el exterior del cluster. Como el rendimiento está limitado por el eslabón más débil de la cadena, veremos cuál es el máximo rendimiento del sistema calculando, en número de iops, el máximo rendimiento alcanzable por cada uno de estos componentes. Asumiremos que:

- el rendimiento del procesador está determinado por su rendimiento en MIPS y por el número de instrucciones necesarias para realizar una entrada-salida y enviar los datos a través de la red.
- el rendimiento máximo de la memoria es de 42560 iops.
- el rendimiento de un disco está determinado por el tiempo promedio dedicado a realizar una entrada-salida al disco.
- el rendimiento de un enlace de la red Ethernet está determinado por su ancho de banda.

a) **Calcula** cuál es el componente que limita el rendimiento del nodo, y del rack. Calcula para ello el número máximo de iops alcanzable por cada componente.

Para realizar el estudio de fiabilidad del sistema, asumiremos que los tiempos entre fallos se aproximan a una distribución exponencial, y los siguientes valores de MTTF (tiempo medio hasta fallo):

- 1000000 horas para el conjunto procesador - memoria
- 125000 horas para el disco
- 500000 horas para el switch Ethernet
- 200000 horas para la fuente de alimentación
- 200000 horas para el ventilador

b) **Calcula** el MTTF de todo el rack. Expresa el resultado en dias.

Suponed que el sistema está optimizado para capacidad y rendimiento (medido en iops), y no para tolerancia a fallos.

c) **Indica** cuál es el nivel RAID más adecuado para conseguir el mejor rendimiento y capacidad. Argumenta, en términos de rendimiento, porqué es mejor el tamaño de bloque de datos promedio de 50KB y no uno mayor o menor.

Un nodo completo disipa una potencia de 80W (la potencia disipada por el switch es despreciable), y que tenemos un límite de disipación de 40KW para todo el cluster.

d) **Calcula** si es posible llegar al PetaByte de almacenamiento usando racks completos.

Queremos rediseñar el sistema para que sea tolerante a fallos de disco. Un diseñador experto afirma que, si con los discos de cada nodo se crea una organización en array RAID 0+1 con 2 grupos de 2 discos cada uno, el resultado:

1) no afectará al rendimiento
2) no afectará a la capacidad efectiva de nuestro sistema
3) aumentará su fiabilidad

e) **Indica** cuáles de las afirmaciones del experto son ciertas o falsas y porqué.