COGNOMS (en majúscules): . . . . . .

NOM (en majúscules): . . . . . .

Duració: 3 hores

Problema 1. (3,2 puntos)

Un procesador con direcciones físicas de 32 bits dispone de una cache de datos (D-Cache) de primer nivel de 16KB y asociatividad igual a 4 (4 vías). El tamaño del bloque de cache es de 16bytes.

a) **Calcula** el número de bits requeridos para las etiquetas (TAGs) de la D-Cache

Para implementar la memoria de etiquetas y datos se utiliza una tecnología SRAM donde para cada bit (de datos o etiquetas) se requieren en total 6 transistores.

b) **Calcula** el número total de transistores que hacen falta para implementar la memoria de etiquetas de la D-cache. No hace falta considerar los transistores del decodificador/multiplexor.

Al analizar la traza de un programa se observa que las direcciones 0x1000, 0x2000, 0x3000, 0x4000, 0x5000 son accedidas en este orden y de forma repetitiva. La secuencia de accesos es 0x1000, 0x2000, 0x3000, 0x4000, 0x5000, 0x1000, 0x2000, 0x3000, 0x4000, 0x5000, 0x1000, 0x2000, 0x3000, 0x4000, 0x5000, …

c) **Calcula** el número de fallos de cache para los primeros 100 accesos de la secuencia descrita considerando una política de reemplazo LRU.

d) **Razona** si, para la traza anterior y con LRU, podríamos reducir el número de fallos aumentando el tamaño de la cache.

e) **Razona** si emplear una política de reemplazo aleatoria mejoraría o empeoraría los resultados de LRU para el caso anterior.

En este procesador se ha decidido incorporar soporte para memoria virtual con un tamaño de página de 4Kbytes. Para minimizar el impacto en el rendimiento se incluye un TLB. El TLB que se ha introducido es completamente asociativo, dispone de 4 entradas e incorpora una política de reemplazo LRU.

f) **Indica,** para los 100 primeros accesos de la traza anterior, que páginas se acceden y cuántas veces se accede a cada una de ellas.

g) **Calcula** el número de fallos de TLB para los primeros 100 accesos

h) **Razona** si aumentar el tamaño (número de entradas) del TLB puede mejorar la tasa de aciertos para la traza descrita.

COGNOMS (en majúscules): . . . . . .

NOM (en majúscules): . . . . . .

Duració: 3 hores

Problema 2. (3,2 puntos)

El siguiente código en C realiza el producto matriz por vector. El compilador (derecha), realiza los accesos a b en el bucle externo (for1), la misma optimización que hicimos manualmente en la practica 8. El bucle interno (for2), delimitado por el rectángulo, tiene 6 instrucciones. En este ejercicio **estudiaremos exclusivamente los accesos a datos** del código generado por el compilador (resaltados en negrita).

```
#define N 1024
int a[N], b[N], w[N][N];

int i, j;
for (i=0;i<N; i++)
 for (j=0;j<N; j++)
   b[i] = b[i] + a[j]*w[i][j];
```

```
for1:   ...
        movl   b(%esi), %ecx
for2:   movl   a(%eax), %edx
        imull  w(%ebx,%eax), %edx
        addl   $4, %eax
        addl   %edx, %ecx
        cmpl   $4096, %eax
        jne    for2
        movl   %ecx, b(%esi)
        ...
        jne    for1
```

Este bucle se ejecuta en un procesador superescalar que funciona a una frecuencia de 2 GHz.

a) **Calcula** el ancho de banda necesario (en Gbytes/s) con la memoria de datos para que el bucle interno se pueda ejecutar con un IPC (instrucciones por ciclo) de 1,5.

El procesador tiene direcciones físicas de 32 bits con las que accede a una cache de datos 4-asociativa con algoritmo de reemplazo LRU, tamaño de bloque 64 bytes y política de escritura *Copy Back + Write Allocate*. Las etiquetas (TAGS) de la cache son de 20 bits.

b) **Calcula** el número de bloques (líneas) de la cache.

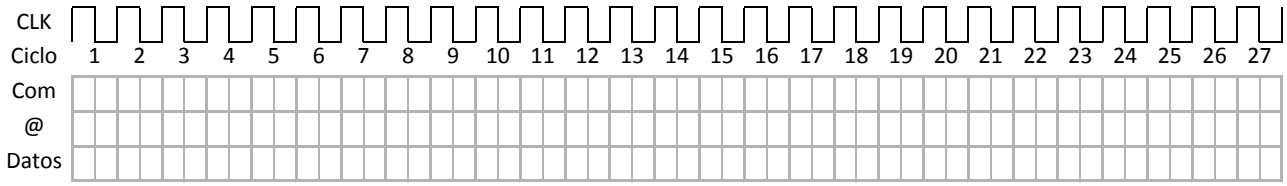
c) **Calcula** el número de fallos que se producirán en la cache de datos para cada una de las estructuras a, b, w.

d) **Calcula** el ancho de banda necesario (en Gbytes/s) entre la cache de datos y la memoria principal para que el bucle interno se pueda ejecutar con un IPC (instrucciones por ciclo) de 1,5

La memoria principal está formada por un único módulo DIMM estándar de 4 GBytes. Este DIMM tiene 8 chips de memoria **DDR**-SDRAM (**Double Data Rate** Synchronous DRAM). El DIMM está configurado para leer/escribir ráfagas de 64 bytes (justo el tamaño de bloque de las caches). La latencia de fila es de 5 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.
Abreviaturas comandos: PR, AC, RD, WR.
Abreviaturas direcciones: F (fila), C (columna), B (banco), FB (fila+banco), CB (columna+banco).

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



La cache implementa un mecanismo de *prefetch on miss*, de forma que cuando se produce un fallo se lee el bloque que ha provocado el fallo y se hace prefectch del bloque siguiente. 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.

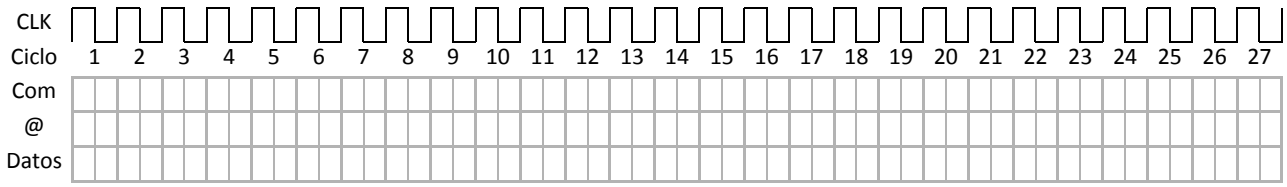
Se quiere estudiar diferentes formas de mapear las direcciones física al DIMM. Cada uno de los 8 chips del DIMM está organizado en 32 bancos de 4096 filas x 4096 columnas. A continuación se muestra un ejemplo de mapeo de direcciones.

$b_4b_3b_2b_1b_0f_{11}f_{10}f_9f_8f_7f_6f_5f_4f_3f_2f_1f_0c_{11}c_{10}c_9c_8c_7c_6c_5c_4c_3c_2c_1c_0x_2x_1x_0$

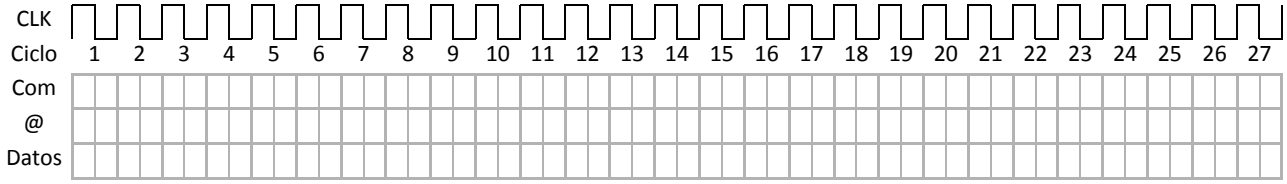
Las direcciones físicas del procesador tienen los bits de menos peso a la derecha y los de más peso a la izquierda. En este ejemplo los bits de menos peso de la dirección (2-0) se usan para seleccionar el chip ($x_2x_1x_0$), los bits (14-3) seleccionan la columna ($c_{11}c_{10}c_9c_8c_7c_6c_5c_4c_3c_2c_1c_0$), los bits (26-15) seleccionan la fila ($f_{11}f_{10}f_9f_8f_7f_6f_5f_4f_3f_2f_1f_0$), y finalmente los bits (31-27) seleccionan el banco ($b_4b_3b_2b_1b_0$). El subíndice indica el peso del bit correspondiente dentro del campo seleccionado. Por ejemplo el bit 31 de la dirección se corresponde al bit 4 del banco y el bit 27 se corresponde al bit 0 del banco.

Rellena los siguientes cronogramas para la lectura de dos bloques de 64 bytes que se encuentran en las direcciones físicas 0 y 64 respectivamente, en función de como se mapean las direcciones al DIMM. Recuerda que el objetivo es generar un secuencia de comandos de forma que se minimize el tiempo total de leer los dos bloques.

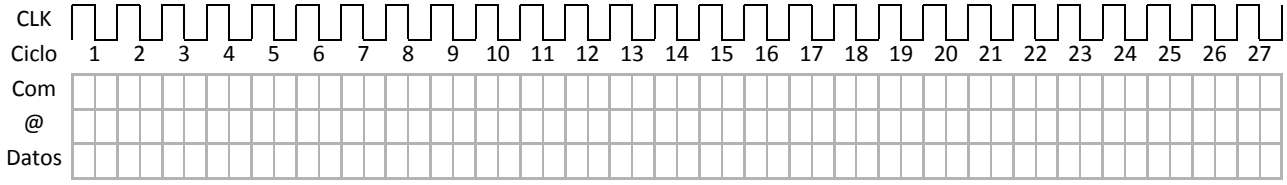
f) **Rellena** el cronograma para la lectura de dos bloques en las direcciones 0 y 64 con el siguiente mapeo:
$b_4b_3b_2b_1b_0f_{11}f_{10}f_9f_8f_7f_6f_5f_4f_3f_2f_1f_0c_{11}c_{10}c_9c_8c_7c_6c_5c_4c_3c_2c_1c_0x_2x_1x_0$



g) **Rellena** el cronograma para la lectura de dos bloques en las direcciones 0 y 64 con el siguiente mapeo:
$f_{11}f_{10}f_9f_8f_7f_6f_5f_4f_3f_2f_1f_0c_{11}c_{10}c_9c_8c_7c_6c_5c_4c_3b_4b_3b_2b_1b_0c_2c_1c_0x_2x_1x_0$



h) **Rellena** el cronograma para la lectura de dos bloques en las direcciones 0 y 64 con el siguiente mapeo:
$b_4b_3b_2b_1b_0c_{11}c_{10}c_9c_8c_7c_6c_5c_4c_3f_{11}f_{10}f_9f_8f_7f_6f_5f_4f_3f_2f_1f_0c_2c_1c_0x_2x_1x_0$



COGNOMS (en majúscules): . . . . . .

NOM (en majúscules): . . . . . .

Duració: 3 hores

Problema 3. (3,6 puntos)

Se ha ejecutado un programa P en un sistema con un solo procesador y un solo disco D (un PC de sobremesa que denominaremos PC1) y se ha visto que su tiempo de ejecución es de T horas. Para poder estimar el rendimiento del programa P hemos medido (en el PC1) que el programa se ejecuta en tres fases bien diferenciadas:

Fase 1: Código SECUENCIAL que no puede paralelizarse, ocupa el 2% del tiempo de la ejecución de P en el PC1.

Fase 2: Código PARALELIZABLE, ocupa el 80% del tiempo de la ejecución de P en el PC1. Se puede paralelizar perfectamente sin overhead de comunicación/sincronización.

Fase 3: Código de E/S que pasa todo su tiempo accediendo en el disco D, ocupa el 18% del tiempo de la ejecución de P en el PC1.

Con el objeto de reducir el tiempo de ejecución del programa, se baraja la opción de substituir el procesador del PC1 por un sistema multiprocesador de 16 procesadores idénticos al del PC1, manteniendo igual el resto del sistema. A este sistema multiprocesador le llamaremos PC2.

a) **Calcula** el máximo speed-up que podría conseguirse al ejecutar el programa P con el PC2.

Sabemos que el programa P tiene 10^5 instrucciones estáticas y ejecuta 300×10^{12} instrucciones dinámicas. En la Fase 2, la única que realiza cálculos en punto flotante, se ejecutan 260×10^{12} instrucciones, de las cuales 150×10^{12} son instrucciones de coma flotante que realizan un total de 100×10^{12} operaciones de coma flotante en simple precisión.

b) **Calcula** los MIPS y MFLOPS al ejecutar el programa P en el PC1 si la ejecución tarda 50×10^3 segundos.

c) **Calcula** los MIPS y MFLOPS al ejecutar el programa P en el PC2 en vez de en PC1.

Otra opción que se ha barajado para mejorar el rendimiento del sistema PC1 es añadir un RAID de discos en lugar del disco duro D. El RAID nos permite paralelizar la Fase 3, ya que en esta fase hay suficientes accesos como para saturar el ancho de banda de todos los discos. El disco D tiene una capacidad de 1 Terabyte y un ancho de banda de 400 MBytes/s.

El RAID de discos del que disponemos tiene 6 discos como el D y puede configurarse como **RAID 5** o **RAID 51**.

d) **Describe** las principales características de cada uno de estos sistemas RAID, **dibujando** un esquema de cómo se distribuyen los datos y especificando el tipo de entrelazado, Terabytes de información redundante, número mínimo de discos que han de fallar para que el sistema deje de ser operativo, ancho de banda **máximo** de las lecturas en acceso secuencial y ancho de banda **máximo** de las escrituras en acceso secuencial.

Decidimos configurar el sistema de discos como RAID 5. Al sistema multiprocesador con 16 procesadores y RAID 5 de 6 discos le llamaremos PC3

e) **Calcula** el speed-up máximo, **sobre el PC2**, que podemos esperar al ejecutar el programa P en el **PC3**, asumiendo que todos los accesos a disco son lecturas secuenciales.

Sabemos que la Fase 3 tarda 9000 segundos con un solo disco D (PC1 y PC2) y que 2/3 del tiempo se realizan lecturas secuenciales y 1/3 del tiempo se realizan escrituras aleatorias.

f) **Calcula** el tiempo de la Fase 3 en el **PC3** y el ancho de banda efectivo del RAID 5 durante la Fase 3