

COGNOMS: NOM:

3er Control Arquitectura de Computadors

Curs 2016-2017 Q1

- **Temps: 13:15 a 15:15**
- **Poseu clarament amb LLETRES MAJÚSCULES a cada full els cognoms i el nom**

Problema 1. (3 puntos)

Se quiere diseñar una memoria cache de datos con políticas de escritura *write through* y *write NO allocate*:

Se han obtenido por simulación las siguientes medidas para un determinado programa:

- porcentaje de escrituras (sobre el total de accesos): 20%
- tasa de aciertos: 0,9

La memoria cache es de mapeo directo y se leen etiquetas y datos en paralelo. En caso de fallo de lectura, el bloque de MP se escribe en la MC y posteriormente el dato se envía a la CPU desde la MC. El tiempo de acceso (T_{sa}) a memoria cache (MC) es de 10 ns tanto para lectura como escritura. El tiempo de acceso a memoria principal (MP) para escribir una palabra es de 90 ns. Para leer o escribir un bloque en la MP se emplean 130 ns.

a) **Calcula** el tiempo empleado en realizar 1000 accesos consecutivos

Dado el siguiente código escrito en ensamblador del x86:

```

movl $0, %ebx
movl $0, %esi
for:
    cmpl $1024*1000, %esi
    jge end

    (a) movl (%ebx, %esi, 4), %eax
    (b) addl 2*8*1024(%ebx, %esi, 4), %eax
    (c) movl %eax, 3*8*1024(%ebx, %esi, 4)

    addl $1, %esi
    jmp for
end:

```

El código se ejecuta en un sistema con memoria cache y memoria virtual. Queremos estudiar el comportamiento de los accesos a datos en este sistema. La memoria cache de datos es *Write Through + Write No Allocate*, 2-asociativa con reemplazo LRU, tamaño 8 KB y 16 bytes por bloque. Responde a las siguientes preguntas:

- b) **Calcula**, para cada uno de los accesos etiquetados como (a, b, c), el conjunto de la memoria cache al que se accede en cada una de las 13 primeras iteraciones del bucle

iteración	0	1	2	3	4	5	6	7	8	9	10	11	12
a													
b													
c													

Calcula la cantidad de aciertos y de fallos de cache, en todo el código.

La memoria virtual utiliza páginas de tamaño 8KB y disponemos de un TLB de 4 entradas y reemplazo LRU.

- c) **Indica**, para cada uno de los accesos indicados (etiquetas a, b, c), a qué página de la memoria virtual se accede en cada una de las siguientes iteraciones del bucle (recuerda que los accesos son a 4 bytes).

iteración	0	1*1024	2*1024	3*1024	4*1024	5*1024	6*1024	7*1024	8*1024	9*1024
a										
b										
c										

Calcula la cantidad de aciertos y de fallos de TLB, en todo el código.

COGNOMS: NOM:

3er Control Arquitectura de Computadors

Curs 2016-2017 Q1

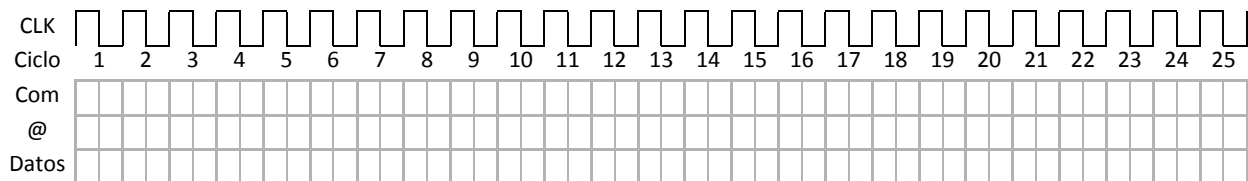
- Temps: 13:15 a 15:15
- Poseu clarament amb LLETRES MAJÚSCULES a cada full els cognoms i el nom

Problema 2. (4 puntos)

Una **CPU** està connectada a una cache de instruccions (**\$I**) i una cache de dats (**\$D**). El conjunt format per **CPU+\$I+\$D** està connectat a una memòria principal formada per un únic mòdul DIMM estàndar de 4 GBytes. Este DIMM té 8 chips de memòria **DDR-SDRAM (Double Data Rate Synchronous DRAM)**. El DIMM està configurat per llegir/escriure ràfegues de 64 bytes (justo el tamaño de bloque de las caches). La latència de fila és de 4 cicles, la latència de columna de 3 cicles i la latència de precarga de 1 cicle.

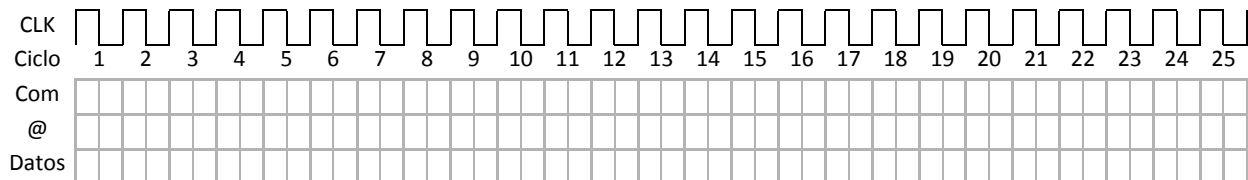
En los siguientes cronogramas, indica la ocupación de los distintos recursos de la memoria DDR: bus de datos, bus de direcciones y bus de comandos. En todos los cronogramas supondremos que no hay ninguna página de DRAM abierta.

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

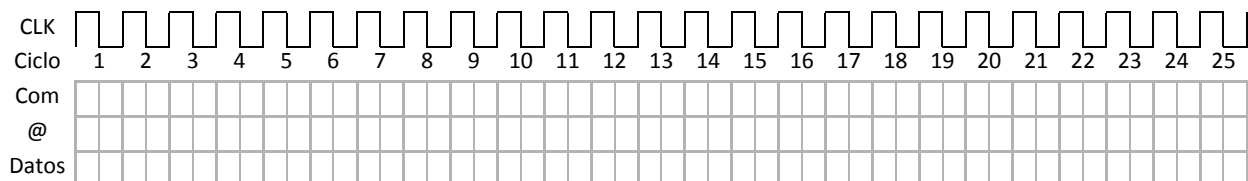


En ocasiones, es posible que el conjunto **CPU+\$I+\$D** solicite múltiples bloques a la DDR (por ejemplo porque se produzca un fallo simultáneamente en **\$I** y en **\$D**). El controlador de memoria envía los comandos necesarios a la DDR-SDRAM de forma que ambos bloques sean transferidos lo más rápidamente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

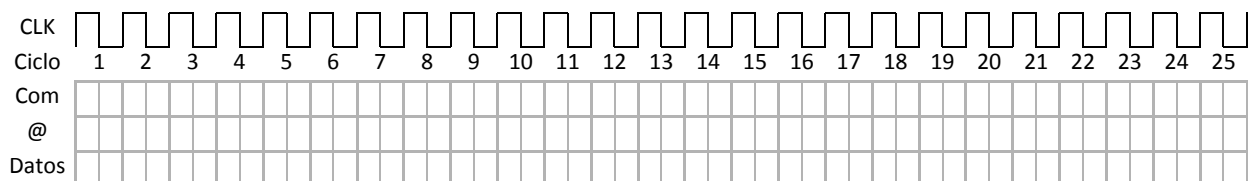
b) Ambos bloques están ubicados en el mismo banco pero en páginas distintas.



c) Ambos bloques están ubicados en la misma página .



d) Ambos bloques están ubicados en bancos distintos.



Un programa P realiza 5×10^9 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 30% 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 estan en registros y **\$D** está inicialmente vacia. Los elementos del vector v son de 8 bytes y los bloques de **\$D** son de 64 bytes. La capacidad de **\$D** es mayor de 8 Kbytes.

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

- En la 1a ejecución el bucle tarda 42.500 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**.

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 *prefetch* previo del bloque (i+1) pendiente de completar (en ambos casos es innecesario hacer prefetch de nuevo). En esta variante de **\$D** ejecutamos una única vez el bucle anterior con N muy grande (mucho mayor que el tamaño de cache).

g) **Calcula** el número máximo de ciclos que puede durar un *prefetch* para que el bucle se ejecute en $30 \cdot N$ ciclos. ¿Es posible ejecutar el bucle en menos de $30 \cdot N$ haciendo el *prefetch* más rápido?

h) **Calcula** los ciclos que tarda en ejecutarse el bucle si un *prefetch* tarda 100 ciclos en traer los datos a la cache. ¿Y si tarda 400 ciclos?

COGNOMS: NOM:

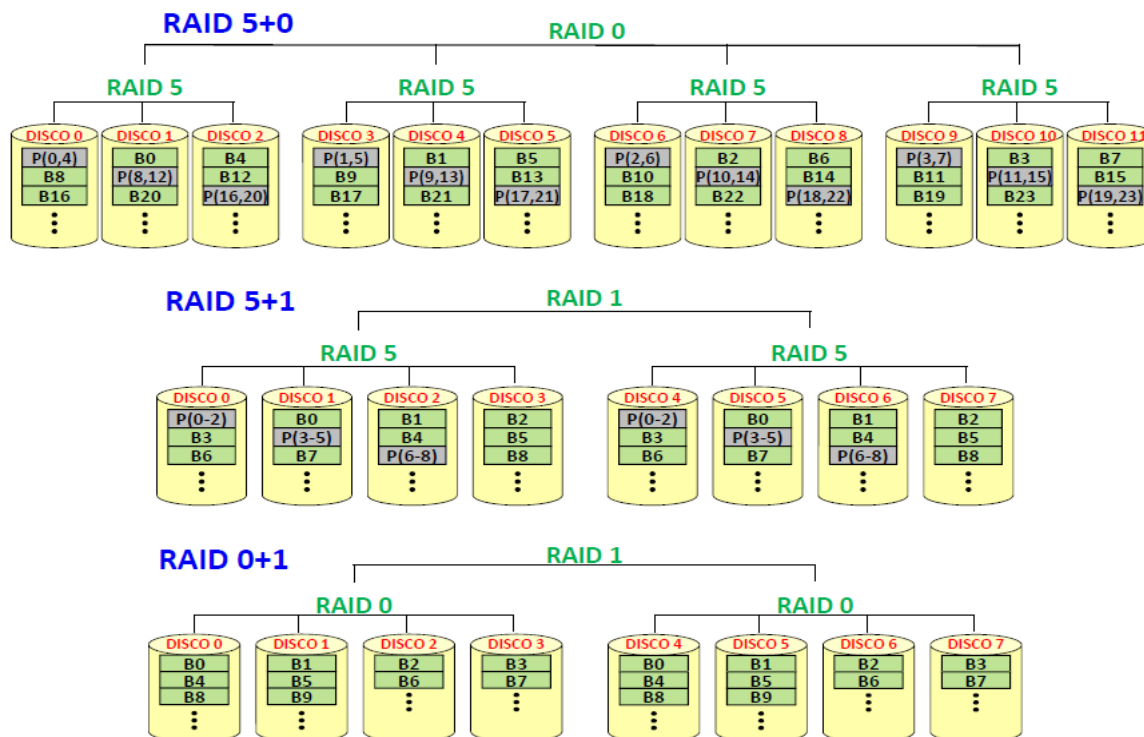
3er Control Arquitectura de Computadors

Curs 2016-2017 Q1

- Temps: 13:15 a 15:15
- Poseu clarament amb LLETRES MAJÚSCULES a cada full els cognoms i el nom

Problema 3. (3 puntos)

Determinadas configuraciones de RAID pueden seguir funcionando aunque algunos discos fallen. En la figura se muestran 3 configuraciones de RAID: RAID 5+0, RAID 5+1 y RAID 0+1.



- a) El RAID 5+0 de la figura puede seguir funcionando aunque fallen 4 discos. Marcad con una cruz en la tabla qué 4 discos podrían fallar y el RAID seguir funcionando.
- b) El mismo RAID 5+0 de la figura puede dejar de funcionar si fallan 2 discos. Marcad con una cruz en la tabla qué 2 discos deberían fallar para que el RAID no funcione.

	RAID 0											
	RAID 5			RAID 5			RAID 5			RAID 5		
	disco 0	disco 1	disco 2	disco 3	disco 4	disco 5	disco 6	disco 7	disco 8	disco 9	disco 10	disco 11
(a)												
(b)												

- c) El RAID 5+1 de la figura puede seguir funcionando aunque fallen 5 discos. Marcad con una cruz en la tabla qué 5 discos podrían fallar y el RAID seguir funcionando.
- d) El mismo RAID 5+1 de la figura puede dejar de funcionar si fallan 4 discos. Marcad con una cruz en la tabla qué 4 discos deberían fallar para que el RAID no funcione.

	RAID 1							
	RAID 5				RAID 5			
	disco 0	disco 1	disco 2	disco 3	disco 4	disco 5	disco 6	disco 7
(c)								
(d)								

- e) El RAID 0+1 de la figura puede seguir funcionando aunque fallen 4 discos. Marcad con una cruz en la tabla qué 4 discos podrían fallar y el RAID seguir funcionando.
- f) El mismo RAID 0+1 de la figura puede dejar de funcionar si fallan 2 discos. Marcad con una cruz en la tabla qué 2 discos deberían fallar para que el RAID no funcione.

	RAID 1							
	RAID 0				RAID 0			
	disco 0	disco 1	disco 2	disco 3	disco 4	disco 5	disco 6	disco 7
(e)								
(f)								

Queremos evaluar el rendimiento de utilizar estos RAIDs, para ello vamos a utilizar discos de 1 TB de capacidad y 500 MB/s de ancho de banda. En la evaluación vamos a utilizar una aplicación con un porcentaje muy elevado de operaciones de E/S con disco. En esta aplicación pueden identificarse 4 fases (en las fases 1, 2 y 4 sólo tendremos en cuenta el coste de la transferencia de información):

- fase 1: En donde se ha de leer de disco los datos de entrada que ocupan 10 GB, distribuidos entre todos los discos.
- fase 2: Cálculo, con un tiempo de ejecución de 3s.
- fase 3: En donde se ha de escribir un bloque de datos de 5 GB, lo que permite realizar **escrituras secuenciales**.
- fase 4, En donde se han de realizar **escrituras aleatorias** de 5 GB de datos, distribuidas uniformemente entre todos los discos.

- g) **Calcula** el tiempo de ejecución de nuestra aplicación si utilizamos un único disco.

- h) **Calcula** el tiempo de ejecución de nuestra aplicación en un RAID 5+1, organizado en 2 grupos de 10 discos cada uno.

- i) **Calcula** el tiempo de ejecución de nuestra aplicación en un RAID 0+1, organizado en 2 grupos de 10 discos cada uno.