

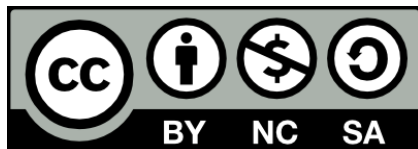
# Tema 3: Jerarquía de Memorias

- Conceptos Avanzados de Memoria Principal

Departament d'Arquitectura de Computadors

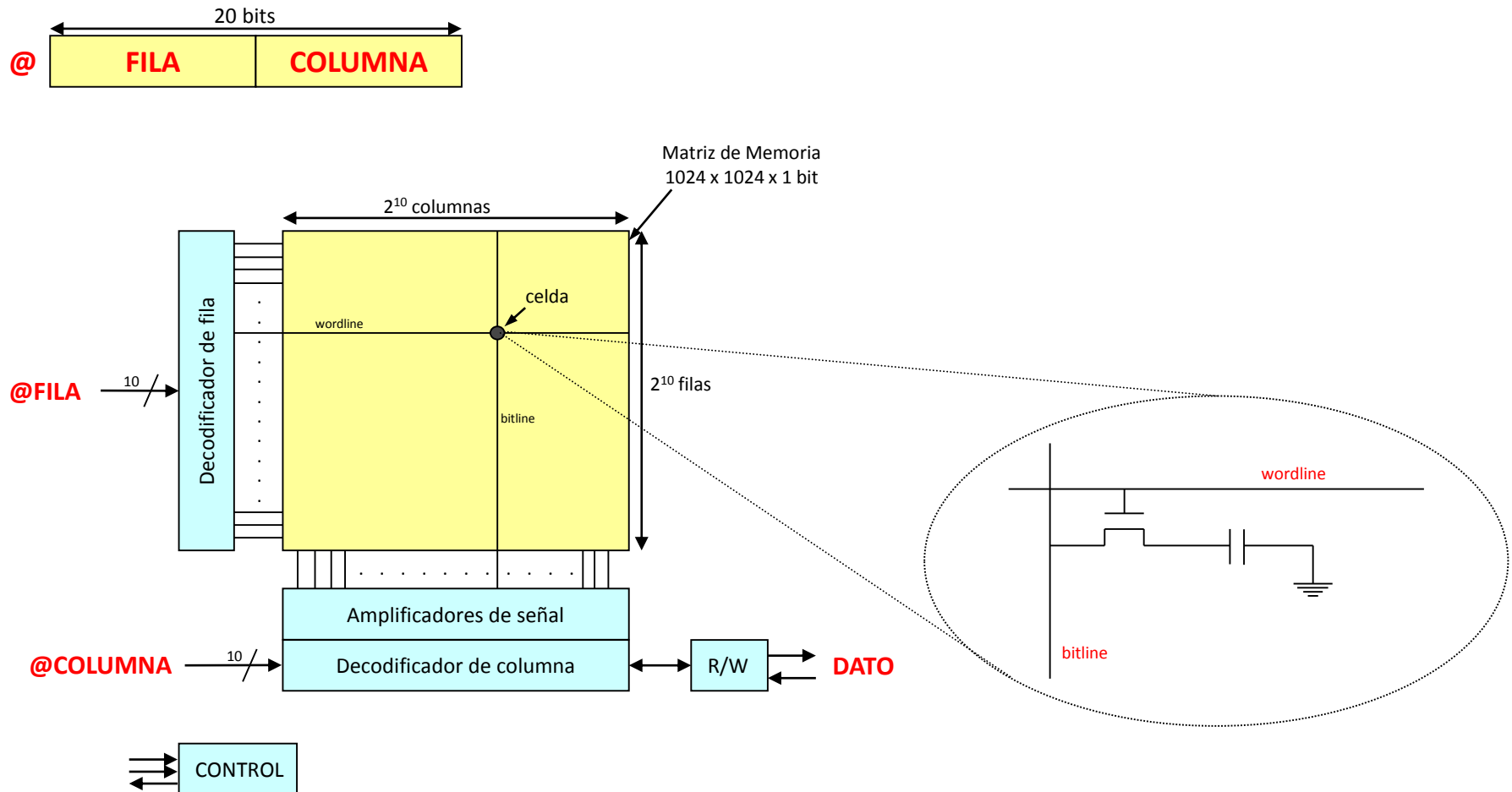
Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya



- Memoria Principal
- Conceptos Básicos Memoria Cache
- Memoria Virtual
- Conceptos Avanzados Memoria Cache
- Memoria Principal
- **Conceptos Avanzados Memoria Principal**
  - Introducción
  - Estructura interna de una DRAM actual
  - Accesos múltiples
  - Organización de la Memoria Principal
  - Refresco en una DRAM
  - Detección y Corrección de Errores
- **Juntándolo todo: Visión General de la Jerarquía**

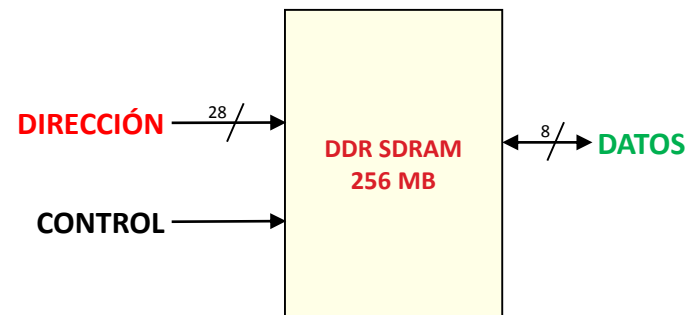
# Estructura básica de una DRAM



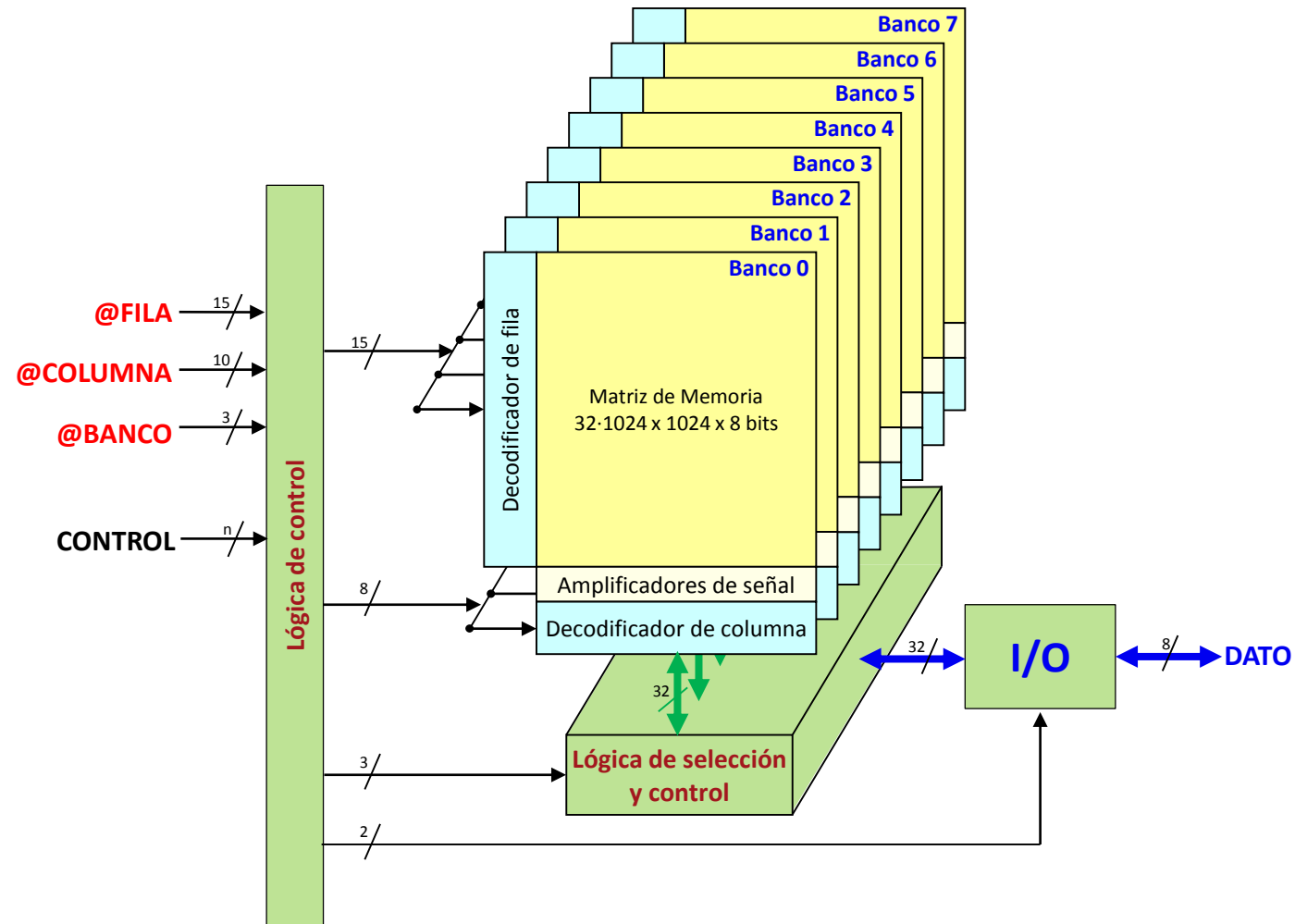
# Estructura actual de una DRAM

- Una característica fundamental de las DRAM actuales es que están divididas en **bancos**.
- Los bancos permiten:
  - Realizar accesos concurrentes a diferentes bancos
  - Ocultar la precarga
  - Ocultar el refresco
  - Tener arrays de memoria más pequeños:
    - ✓ Tiempo de acceso ↓
    - ✓ Consumo de energía ↓

- Por ejemplo, una DDR3 de 256 MB tiene:
  - 8 bancos de 32 MB
  - Cada banco tiene 32 K filas por 1K columnas de 1 byte.

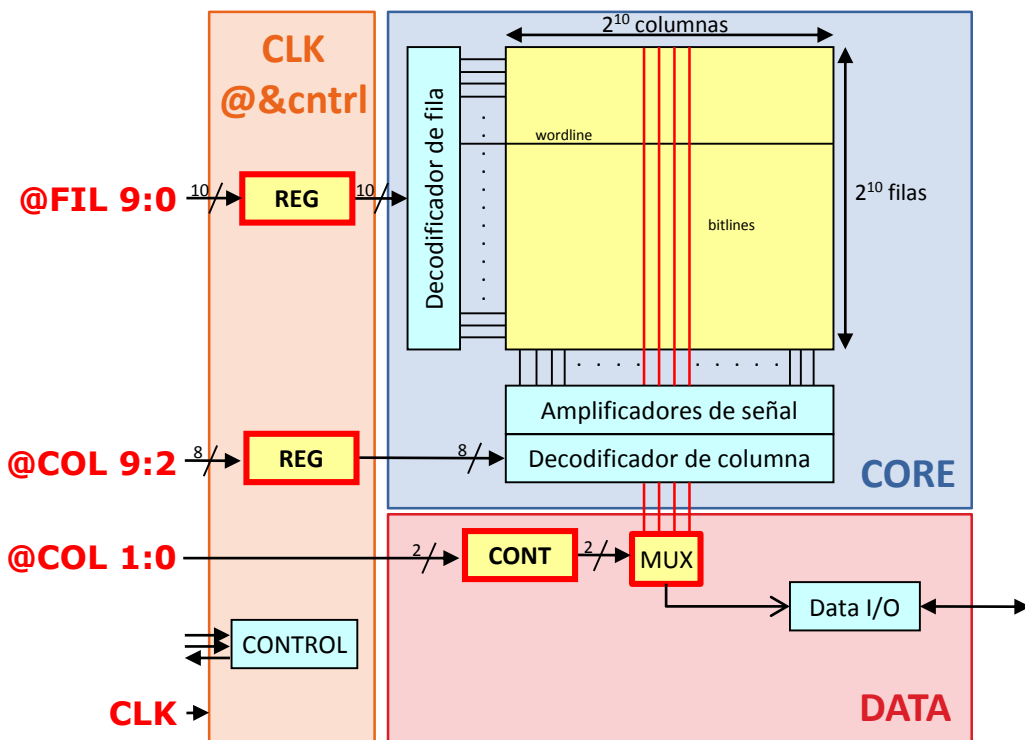


# Estructura actual de una DRAM



# Variantes de SDRAM

## ■ SDR (Single Data Rate), DDR (Double Data Rate), DDR2, DDR3, DDR4

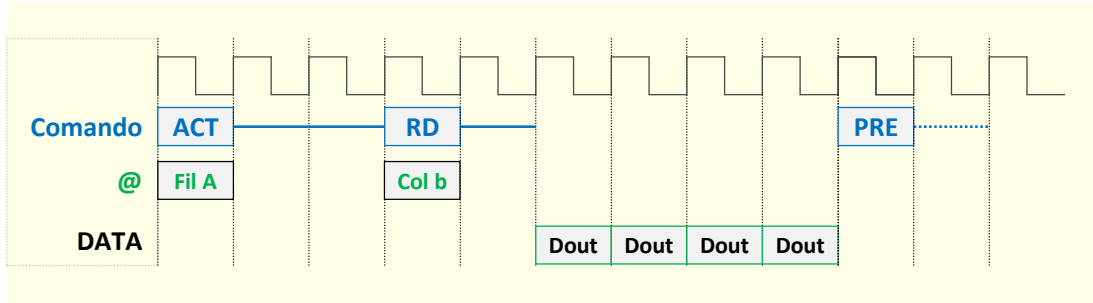


Estándar	CORE (MHz)	BUS CLK (MHz)	Prefetch	DATA RATE M datos / segundo
SDRAM	1x	1x	1n	1x
	100-166	100-166		100-166
DDR	1x	1x	2n	2x
	133-200	133-200		266-400
DDR2	1/2x	1x	4n	2x
	133-200	266-400		533-800
DDR3	1/4x	1x	8n	2x
	133-200	533-800		1066-1600
DDR4	1/8x	1x	8n *	2x
	133-200	1066-1600		2133-3200

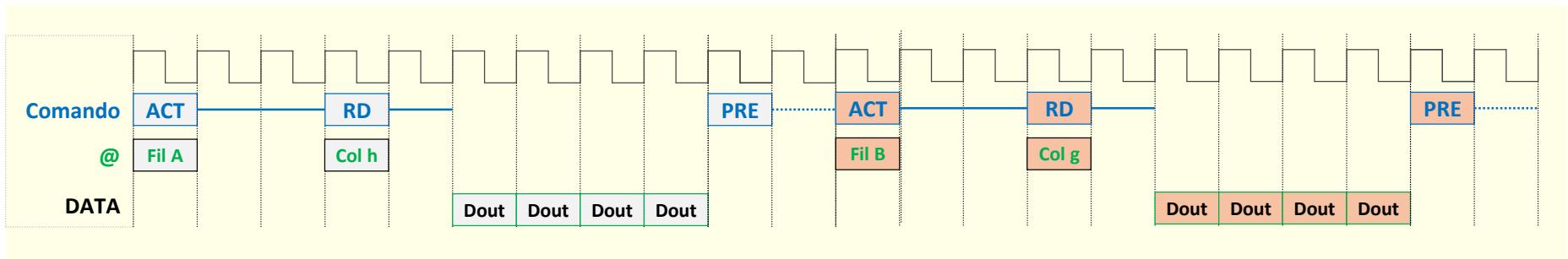
\* DDR4 mantiene el prefetch de DDR3 y añade el concepto de *Bank Group* (que no veremos). Para conseguir el ancho de banda (columna *Data Rate*) es necesario entrelazar entre *bank groups*.

# Accesos a una DRAM con 1 banco

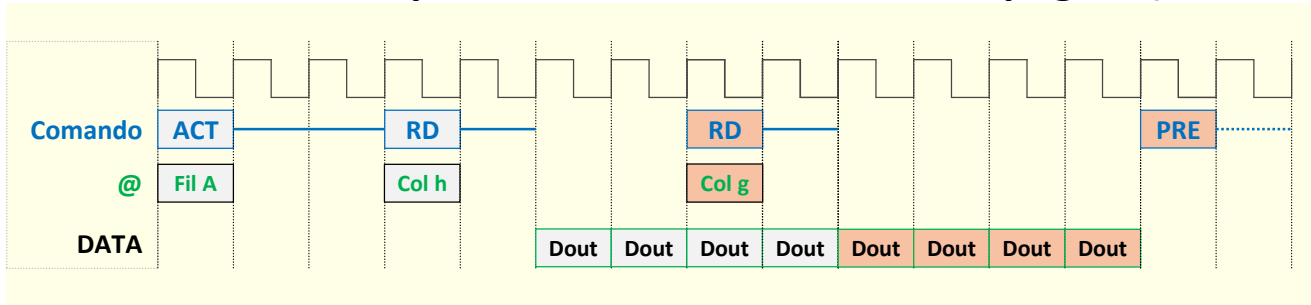
## ■ Lectura de un bloque de memoria (Fila A, columna b)



## ■ Lectura de 2 bloques de memoria en Filas diferentes (Fila A, columna h y Fila B, columna g)



## ■ Lectura de 2 bloques de memoria en la misma página (Fila A, columna h y Fila A, columna g)

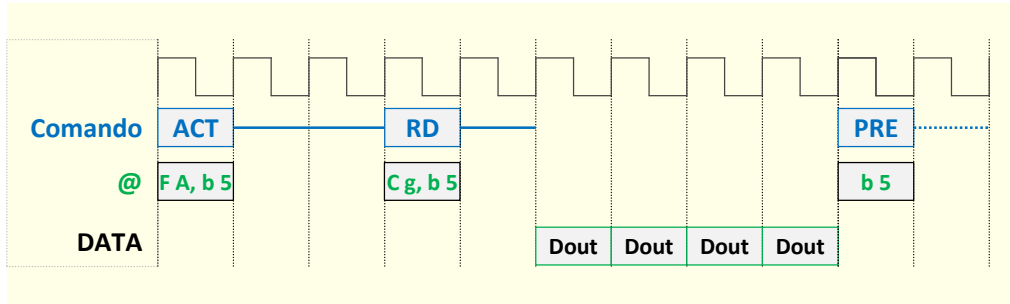


- 32 bytes por bloque
- ACT: 3 ciclos
- WR, RD: 2 ciclos
- PRE: 2 ciclos
- Transferencia: 8B por ciclo



# Accesos a una DRAM con múltiples bancos

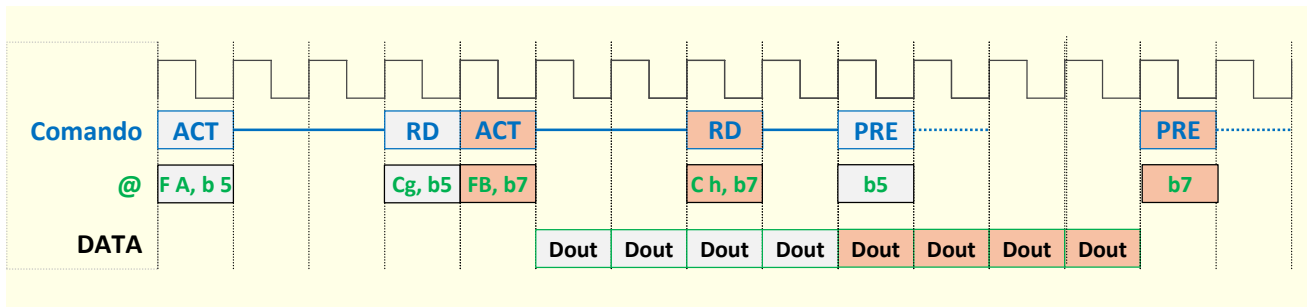
## ■ Lectura de un bloque (Banco 5, Fila A, Columna g)



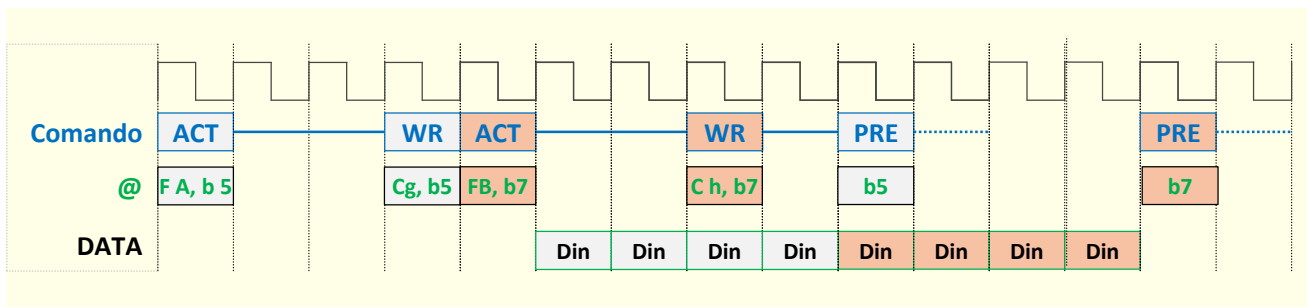
En una DRAM real existen bastantes restricciones que no hemos tenido en cuenta, p.e.:

- Entre una lectura y una escritura hay que esperar N ciclos.
- Entre 2 ACT consecutivos a bancos diferentes han de pasar M ciclos.

## ■ Lectura de 2 bloques en bancos diferentes (Banco 5, Fila A, Columna g y Banco 7, Fila B, columna g)



## ■ Escritura de 2 bloques en bancos diferentes (Banco 5, Fila A, Columna g y Banco 7, Fila B, columna g)



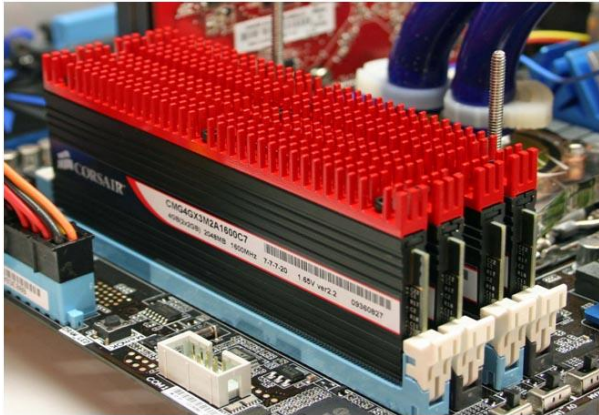
- 32 bytes por bloque
- ACT: 3 ciclos
- WR, RD: 2 ciclos
- PRE: 2 ciclos
- Transferencia: 8B por ciclo



# Organización de la Memoria Principal

## ■ ¿Qué tenemos aquí?

- 8GB de memoria DDR3 1600 MHz
- Direcciones de 33 bits



[www.blog.corsair.com](http://www.blog.corsair.com)

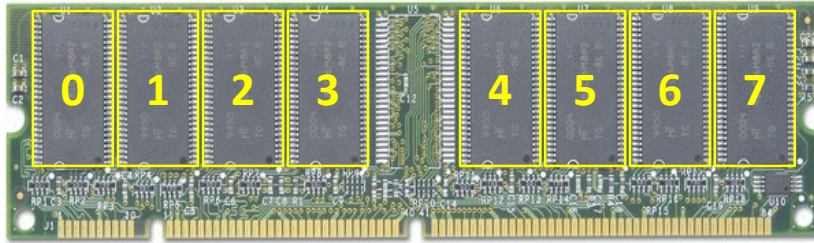
## ■ ¿Cómo está organizado?

- Canales (2)
- DIMMs (2 por canal)
- Chips (8 por DIMM)
- Bancos (8 por chip)
- Filas (32·1024 por Banco)
- Columnas (1024 por Fila)
- 1 byte por columna

## ■ ¿Dónde esta la dirección 0x1DF038A6B?

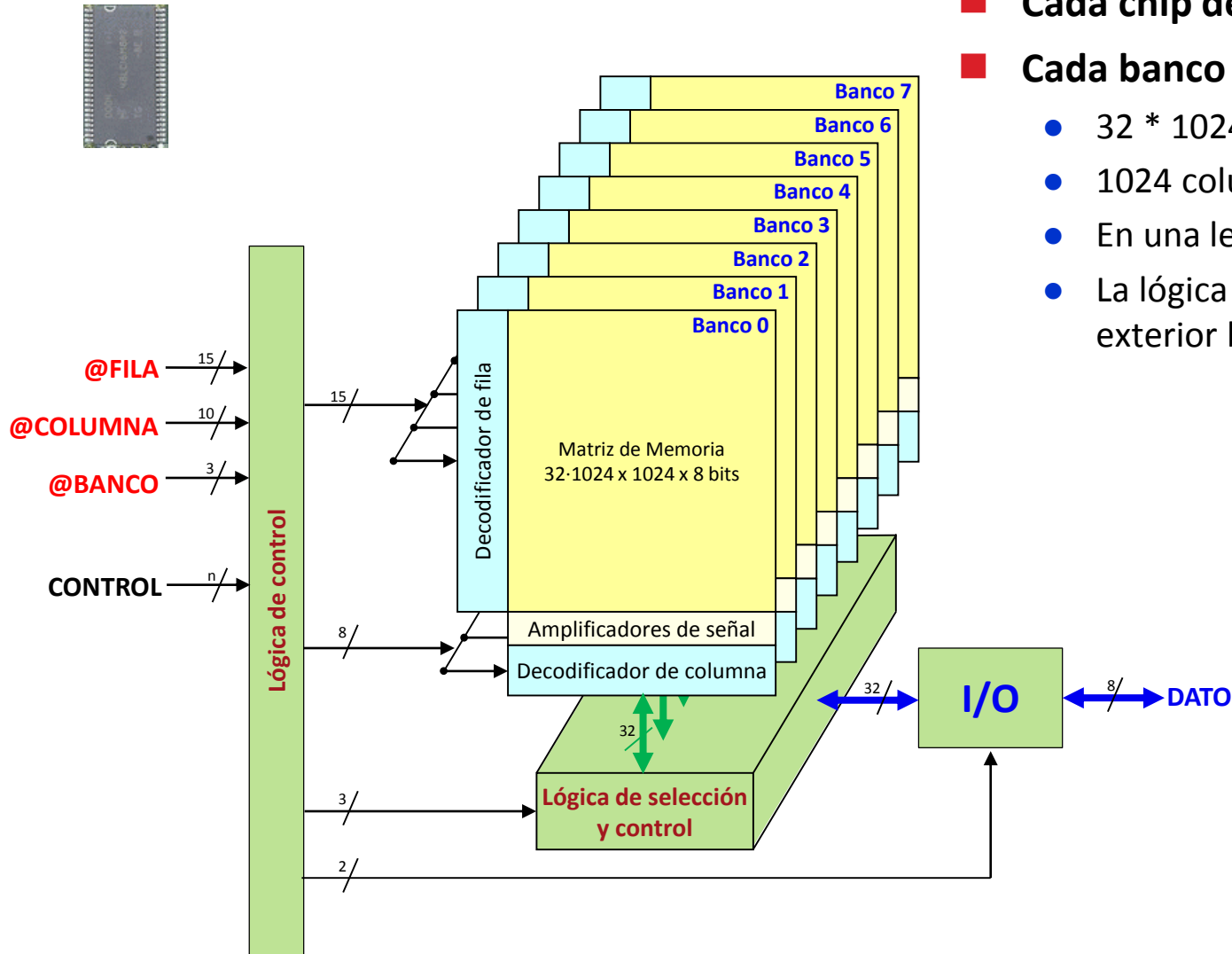
## ■ Dada una dirección, ¿cómo identificamos canal, DIMM, chip, ...?

# Organización de la Memoria Principal



- La Memoria Principal utiliza DIMMS (*Dual Inline Memory Modules*).
- A nivel de ejemplo, un DIMM podría tener las siguientes características:
  - 2GB de capacidad (requiere 31 bits de dirección)
  - Permite leer 64 bits (8 bytes) de datos en un acceso individual
  - 8 chips
- Cada DIMM de memoria tiene un número determinado de chips.
- A nivel de ejemplo, un chip podría tener las siguientes características:
  - 256 MB de capacidad (requiere 28 bits de dirección)
  - Permite leer 8 bits (1 byte) de datos en un acceso individual
  - 8 bancos

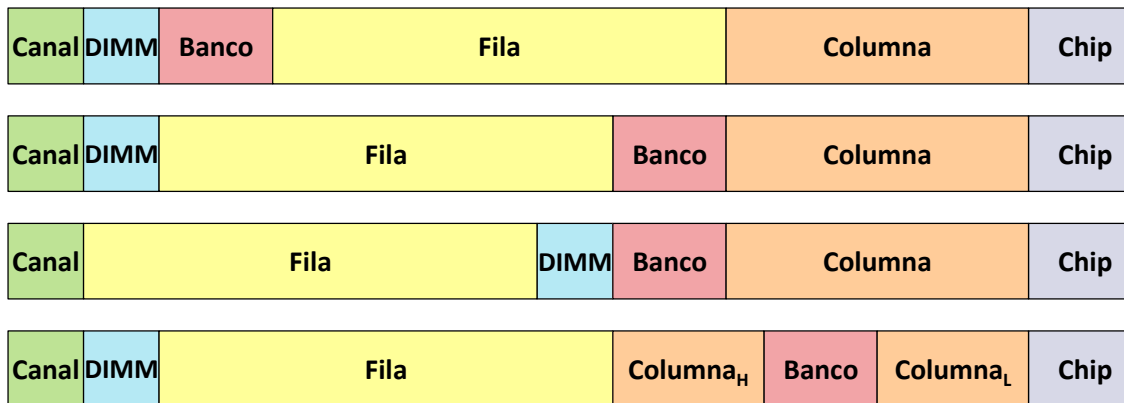
# Organización de la Memoria Principal



- Cada chip de memoria tiene 8 bancos.
- Cada banco es un array de memoria con
  - 32 \* 1024 filas
  - 1024 columnas de 1 byte
  - En una lectura se leen 4 bytes consecutivos
  - La lógica de I/O se encarga de enviarlos al exterior byte a byte.

# ¿Dónde esta la dirección 0x1DF038A6B?

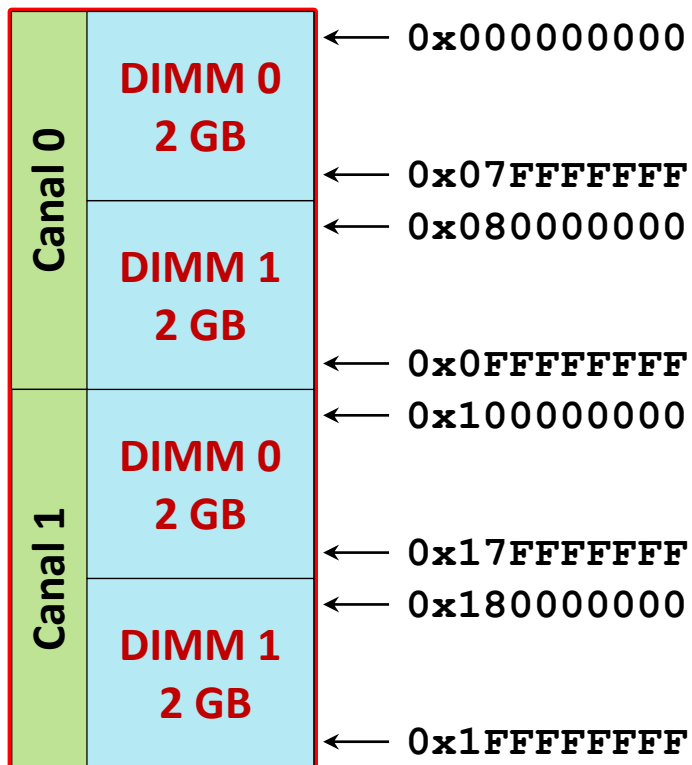
- Nuestra memoria tiene una capacidad de 8GB (33 bits de dirección)
- En nuestra Memoria tenemos:
  - Canales (2): 1 bit para el canal
  - DIMMs (2 por canal): 1 bit para el DIMM
  - Chips (8 por DIMM): 3 bits para el chip
  - Bancos (8 por chip): 3 bits para el banco
  - Filas (32·1024 por Banco): 15 bits para la fila
  - Columnas (1024 por Fila): 10 bits para la columna
- Existen muchas posibilidades válidas:



**¡ El rendimiento del Sistema de Memoria está muy influenciado por esta decisión !**

# ¿Dónde esta la dirección 0x1DF038A6B?

- El mapa de memoria viene determinado por esta descomposición:





# ¿Dónde esta la dirección 0x1DF038A6B?

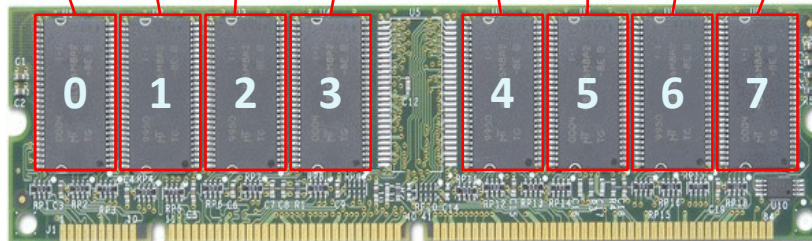
- Es **imprescindible** que los bits bajos de la dirección seleccionen el chip dentro del DIMM

→ **MEMORIA ENTRELAZADA**

- Por ejemplo, las direcciones en el DIMM 0 del Canal 0 se distribuyen así:

0	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	28	29	30	31
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
$2^{31}-8$	$2^{31}-7$	$2^{31}-6$	$2^{31}-5$	$2^{31}-4$	$2^{31}-3$	$2^{31}-2$	$2^{31}-1$

La memoria entrelazada permite realizar accesos de 64 bits de ancho.



Canal, DIMM, Banco, Fila, Columna

Chip

Posiciones consecutivas de memoria se encuentran en el mismo «canal, DIMM, banco, fila y columna» de chips diferentes.

# ¿Dónde esta la dirección 0x1DF038A6B?

- Todos los accesos a Memoria Principal leen (o escriben) bloques de memoria.

# bloque	byte en el bloque
----------	-------------------

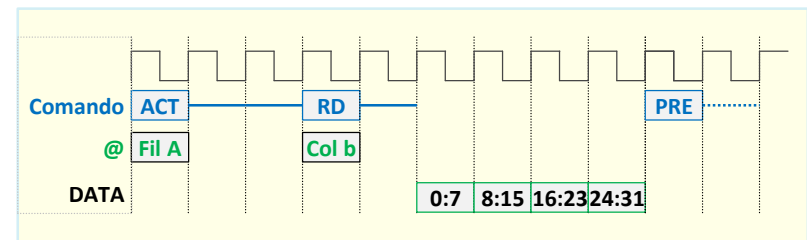
- Las DRAM actuales permiten leer (o escribir) de forma muy eficiente ráfagas de datos.
- Para que esas ráfagas coincidan con bloques de memoria es **imprescindible** que el campo correspondiente al byte dentro del bloque corresponda a los bits que seleccionan chip y columna<sub>L</sub>.

#bloque= (Canal, DIMM, Banco, Fila, Columna <sub>H</sub> )	Col <sub>L</sub>	Chip
--	------------------	------

- Por ejemplo, con bloques de 32 bytes, el bloque 0 se encuentra en el canal 0, DIMM 0:

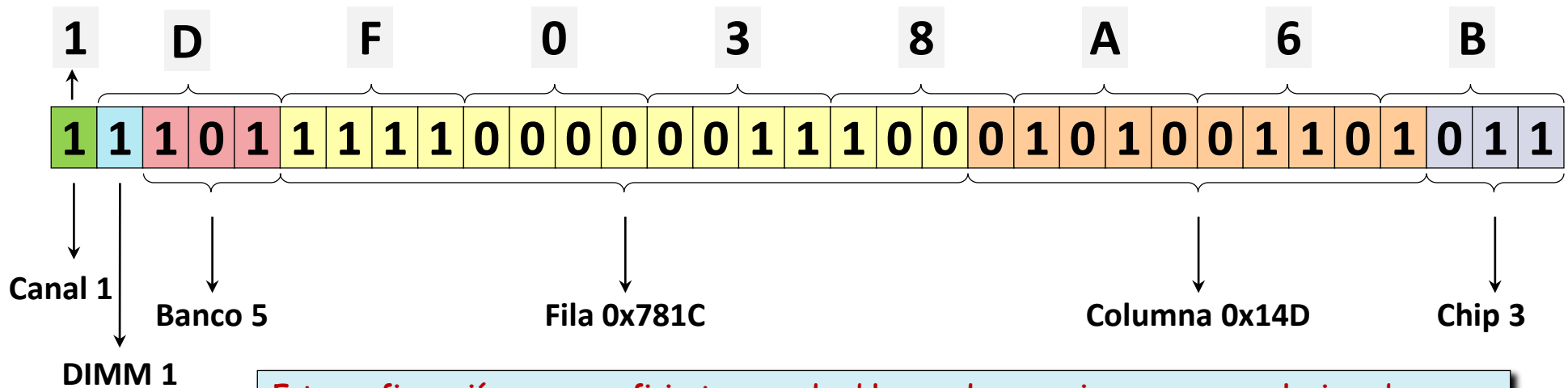
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
$2^{31-8}$	$2^{31-7}$	$2^{31-6}$	$2^{31-5}$	$2^{31-4}$	$2^{31-3}$	$2^{31-2}$	$2^{31-1}$

Para leer un bloque de 32 bytes necesitamos una ráfaga de 4 accesos.

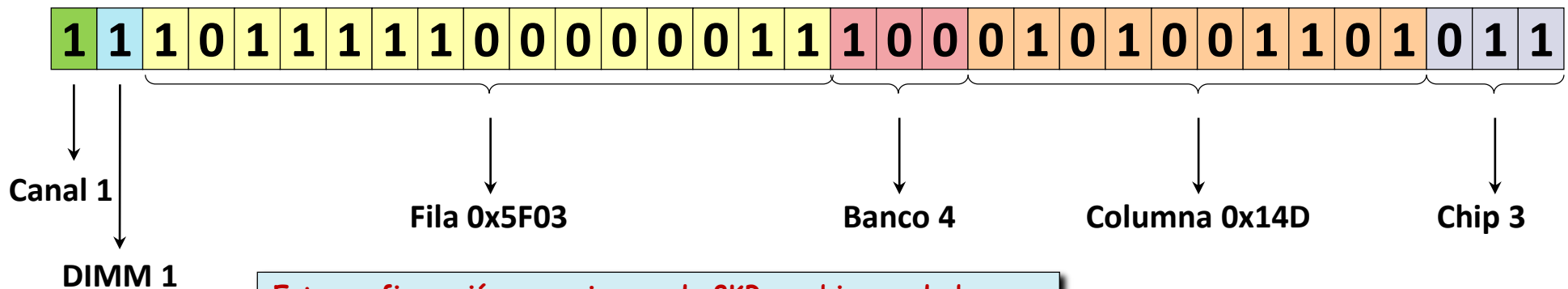




# ¿Dónde esta la dirección 0x1DF038A6B?

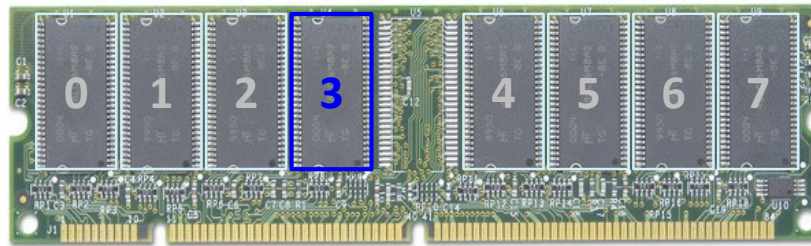
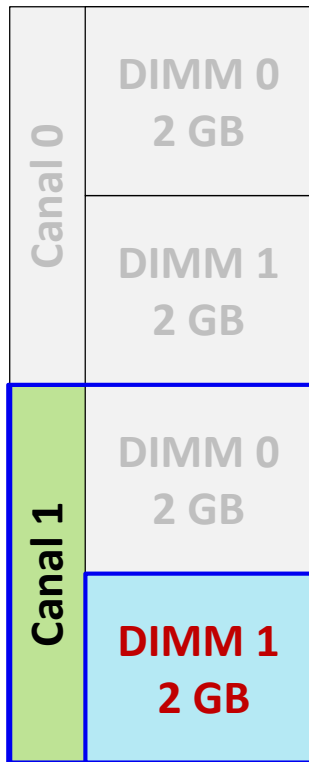
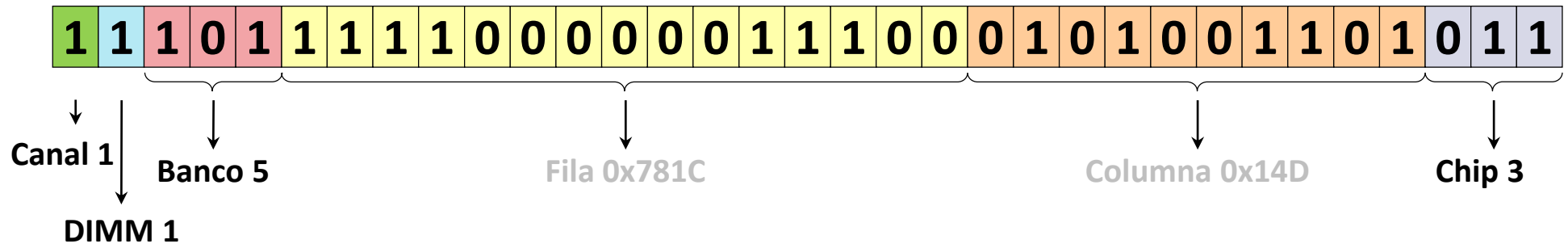


Esta configuración es poco eficiente, grandes bloques de memoria van a parar al mismo banco. Se pierde la posibilidad de tener accesos concurrentes a diferentes bancos.

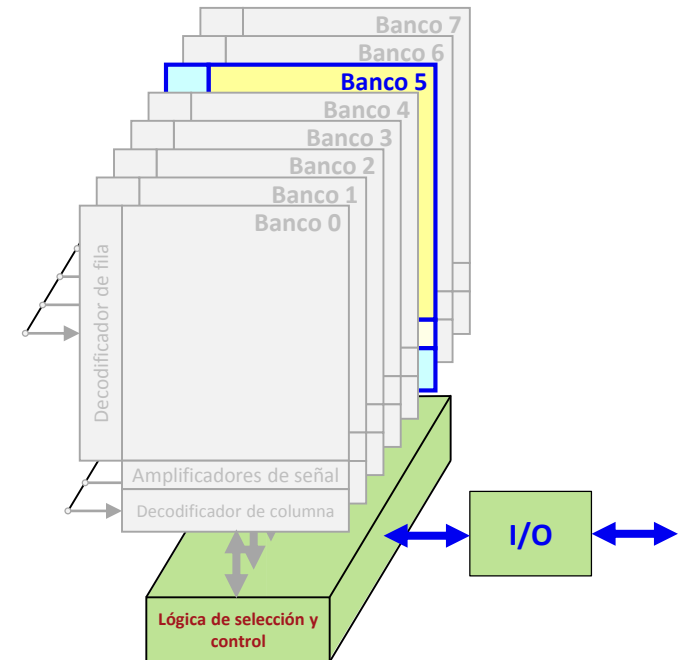


Esta configuración es mejor, cada 8KB cambiamos de banco.

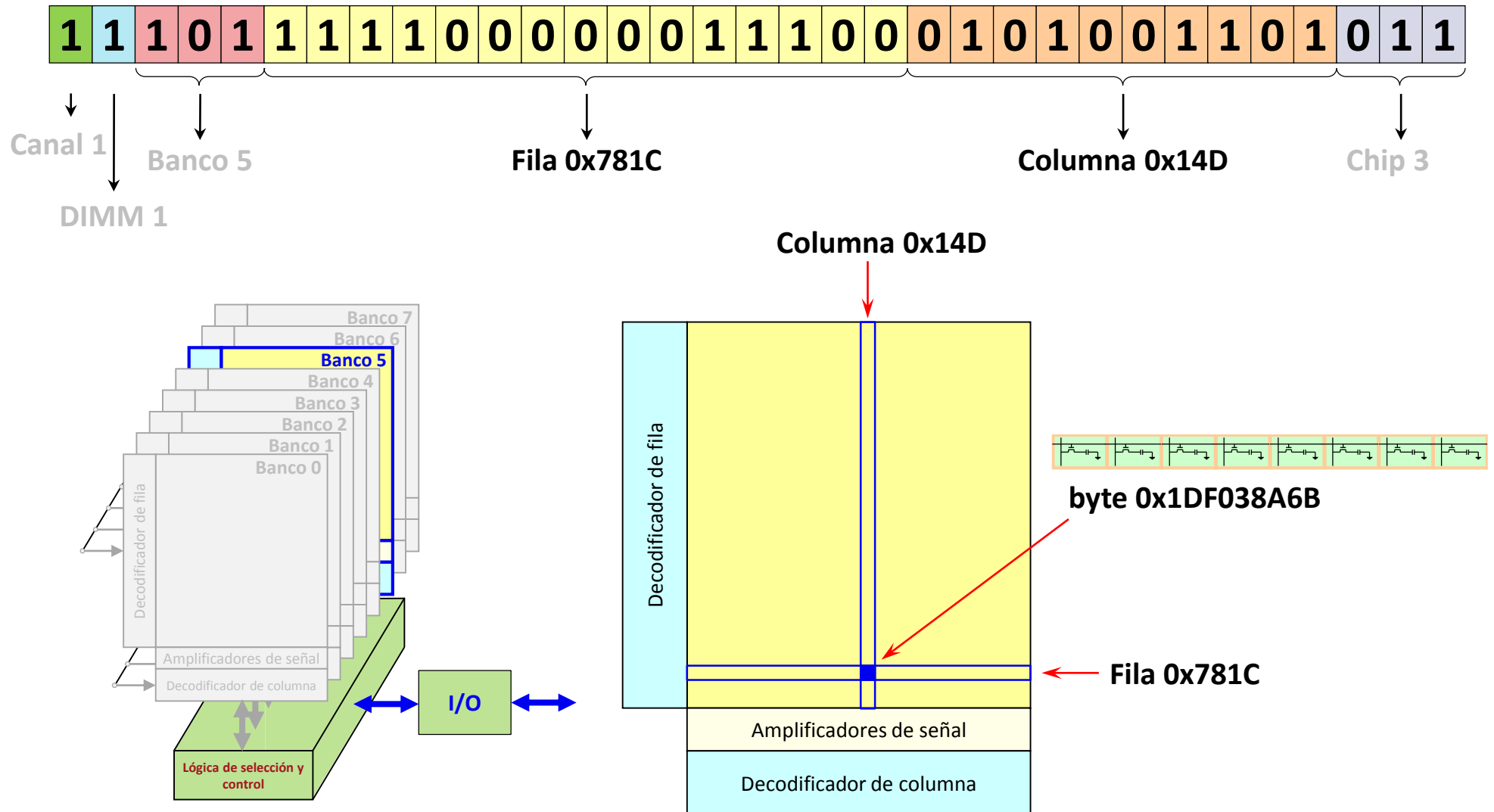
# ¿Dónde esta la dirección 0x1DF038A6B?



Esta configuración sólo es un ejemplo de implementación.

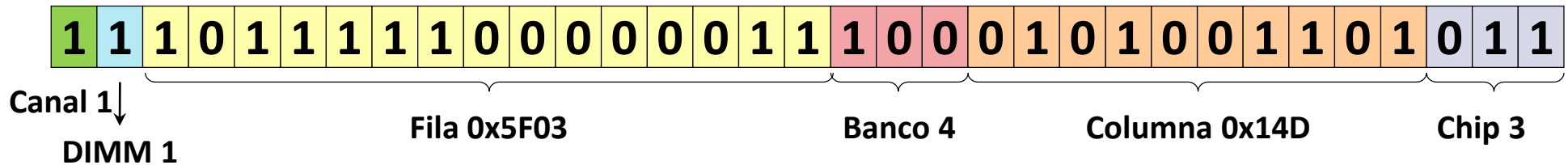


# ¿Dónde esta la dirección 0x1DF038A6B?

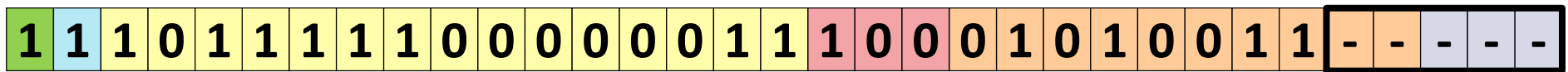


# Lectura línea

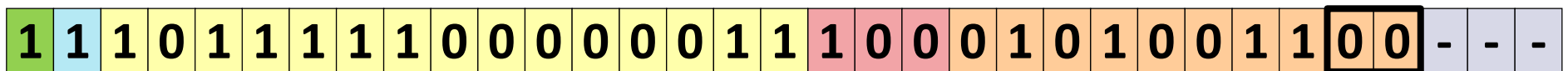
- La transmisión de datos entre MP y MC se hace por bloques completos
- Ejemplo: 32 bytes por bloque, acceso provocado por un fallo en la dirección 0x1DF038A6B



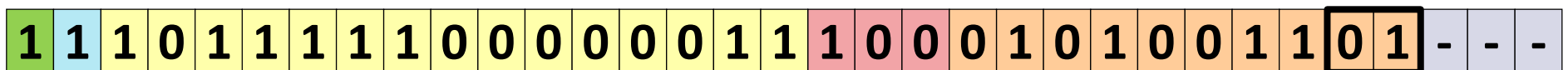
- Hay que leer todos los bytes del bloque:



- Si transferimos el bloque en orden hay que iniciar el acceso con la siguiente dirección:



- Si transferimos en primer lugar el dato que ha provocado en fallo, la dirección es:

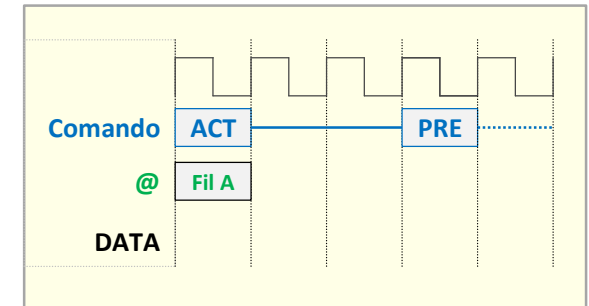
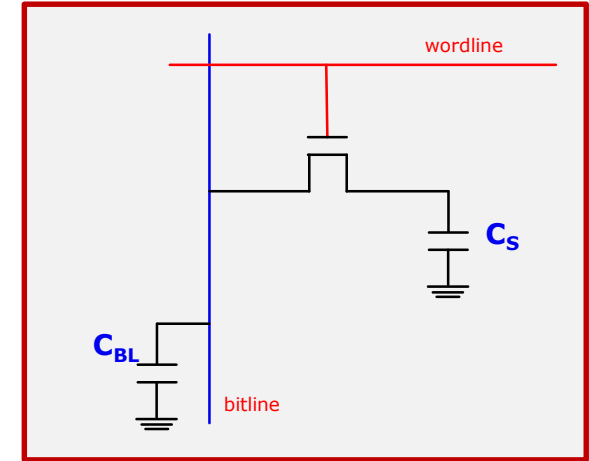


# Problema del refresco en una DRAM

- Cada celda de una DRAM equivale a un condensador, como tal, va perdiendo carga de forma exponencial:
  - Corriente de fuga.
  - La lectura es destructiva.
- Para evitar la pérdida de información almacenada en las celdas, la carga debe regenerarse periódicamente

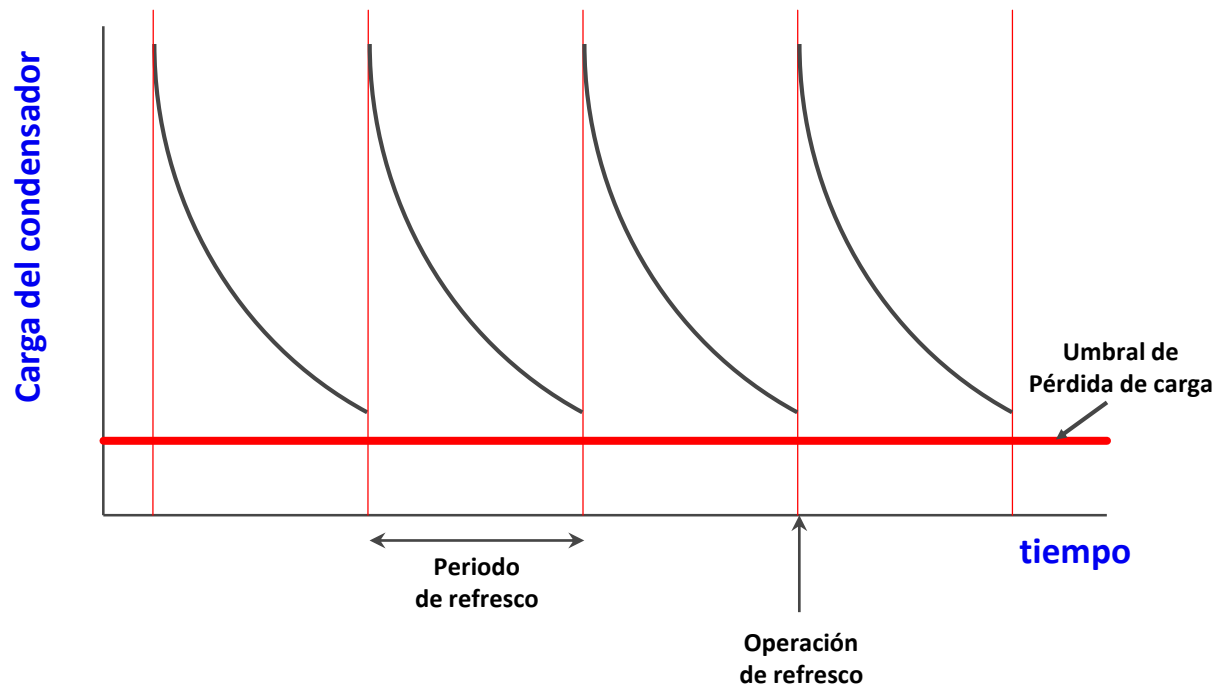
→ **REFRESCO**

- El refresco hay que aplicarlo a todas las celdas de la memoria. Se aplica fila a fila.
- El refresco de la fila A se puede hacer con un ACT y una PRE.
- Existen comandos especiales para hacer el refresco.
  - Refrescan todos los bancos a la vez.
  - La propia memoria controla que fila hay que refrescar.



# Problema del refresco en una DRAM

- Actualmente el refresco es una operación estandarizada establecida por el JEDEC (Junction Electronic Devices Engineering Council)
  - Periodo de refresco de 64ms.
  - Si tenemos  $32 \cdot 1024$  filas, *en media*, hay que lanzar una operación de refresco cada  $2\mu\text{s}$ .



- Frecuencia: 1 GHz
- Coste refrescar fila: 20 ciclos
- Cada 64ms se pierden 655360 ciclos.
- Se pierden  $10,2 \cdot 10^6$  ciclos por segundo.
- Si el refresco NO se puede solapar, se pierde el 1% del tiempo refrescando la memoria.

**¡El coste de refrescar la memoria no es despreciable!**



# Detección y Corrección de Errores

- Al acceder a memoria se pueden producir errores:

1 0 0 0 0 0 0 1 1

Dato almacenado en Memoria



1 0 0 0 0 1 0 1 1

Dato que le llega a la CPU

- Estos errores se deben a diversas causas:
  - Interferencias electrónicas o magnéticas (**transitorios**)
  - Problemas de transmisión de datos (**transitorios**)
  - Problemas hardware (**permanentes**), como en cualquier otro chip y por las mismas causas.

En el IBM Watson Research Center se midió para una muestra de DRAMs clónicas una tasa de error de 5950 fallos cada  $10^9$  horas de funcionamiento a nivel del mar (1 fallo cada 19 años, 2 meses, 7 días y 19 horas). La misma prueba en una mina, bajo una capa de rocas de 15 metros, no produjo ningún error. Estos errores son debidos a los rayos cósmicos.



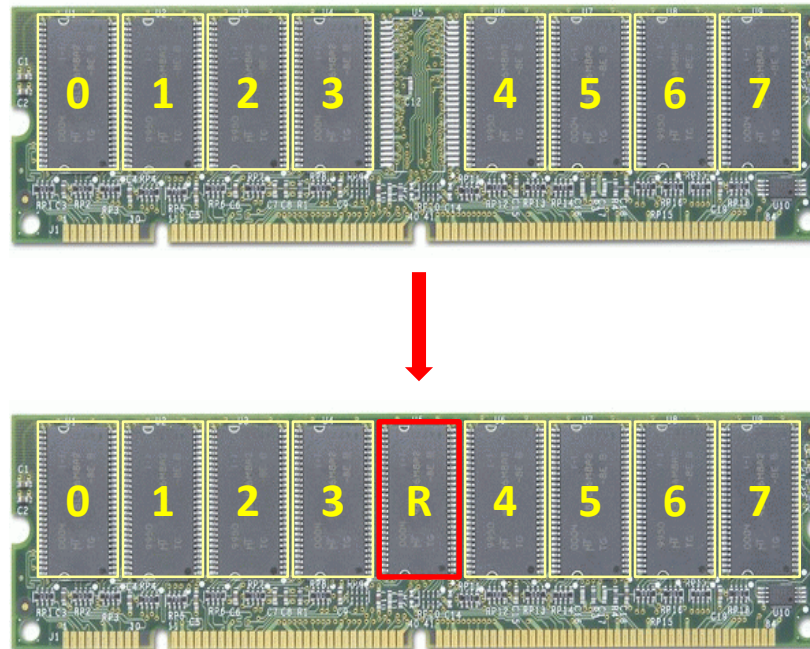
En una instalación como el BSC con alrededor de 48.000 procesadores y contando un mínimo de 8 chips de memoria por procesador se produce un fallo cada 26 minutos (55 por día, 1651 al mes, 20086 al año, ...)



# Detección y Corrección de Errores

- En algunas instalaciones los errores han de estar bajo control.
- Los errores permanentes sólo se resuelven sustituyendo el elemento dañado.
- Los fabricantes de memoria diseñan sistemas que permiten detectar y en algunos casos corregir los errores transitorios.

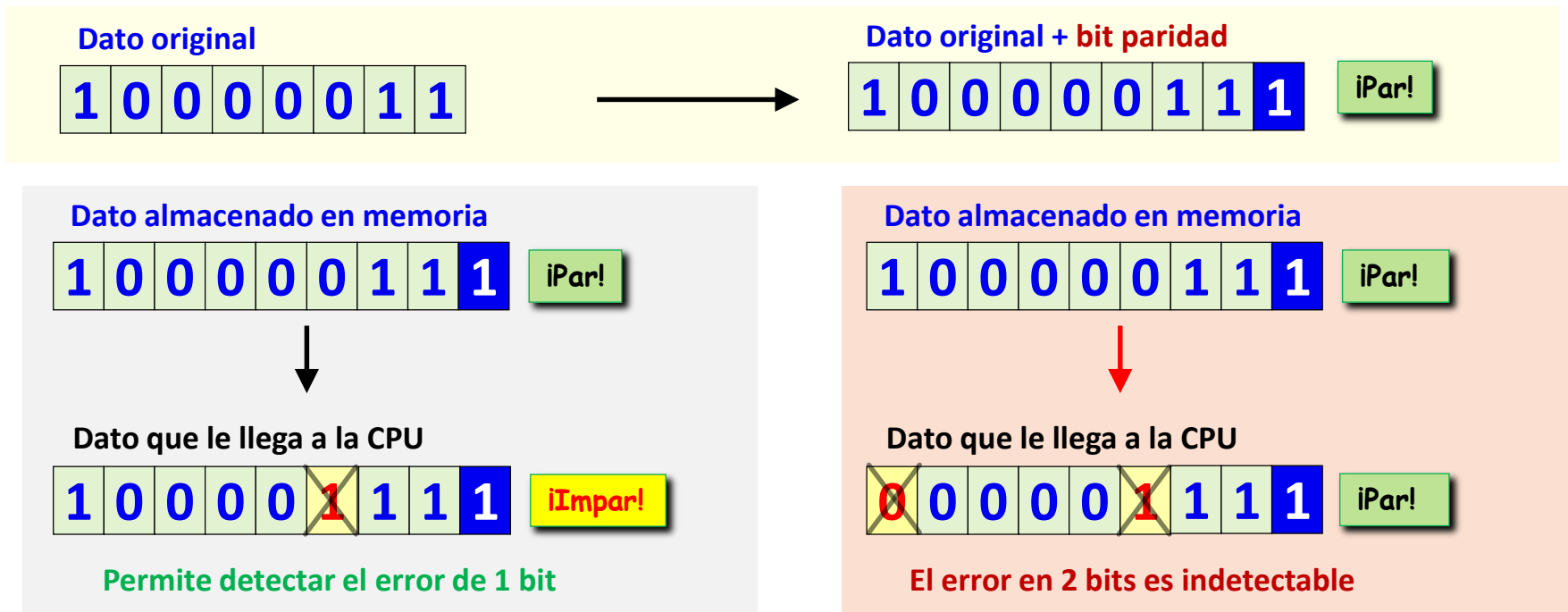
→ **AÑADIR INFORMACIÓN REDUNDANTE**



# Detección y Corrección de Errores

## Bit de Paridad

- El sistema más simple es añadir 1 bit, de tal forma que cada porción de información tenga SIEMPRE un número impar (o par) de 1's.



Este sistema sólo sirve para detectar un número **impar** de errores.  
No permite **corregir** errores.

# Detección y Corrección de Errores

## Código Hamming (7, 3): 4 bits de datos y 3 redundantes

- Bits redundantes: 1, 2, 4
- Bits de datos: 3, 5, 6 y 7

7	6	5	4	3	2	1	
111	110	101	100	011	010	001	
b3	b2	b1	p2	b0	p1	p0	Palabra de 7 bits
111	-	101	-	011	-	001	p0 = b3 xor b1 xor b0
b3	-	b1	-	b0	-	p0	
111	110	-	-	011	010	-	p1 = b3 xor b2 xor b0
b3	b2	-	-	b0	p1	-	
111	110	101	100	-	-	-	p2 = b3 xor b2 xor b1
b3	b2	b1	p2	-	-	-	

7	6	5	4	3	2	1	
1	1	0	p2	1	p1	p0	Palabra de 7 bits
1	-	0	-	1	-	0	p0 = 1 xor 0 xor 1 = 0
1	1	-	-	1	1	-	p1 = 1 xor 1 xor 1 = 1
1	1	0	0	-	-	-	p2 = 1 xor 1 xor 0 = 0

Posibles implementaciones		
Bits Paridad	Bits Datos	total
3	4	7
4	11	15
5	26	31
6	57	63
7	120	127
m	$2^m - m - 1$	$2^m - 1$

# Detección y Corrección de Errores

## Código Hamming (7, 3): 4 bits de datos y 3 redundantes

- Permite **corregir** un error en 1 bit.

1 1 0 0 1 1 0



1 1 ~~1~~ 0 1 1 0

Error en el bit 5

b3	b2	b1	p2	b0	p1	p0	Test Hamming
7	6	5	4	3	2	1	
1	1	1	0	1	1	0	
1	-	1	-	1	-	0	$T0 = p0 \text{ xor } b3 \text{ xor } b1 \text{ xor } b0 = 0^1^1^1 = 1$
1	1	-	-	1	1	-	$T1 = p1 \text{ xor } b3 \text{ xor } b2 \text{ xor } b0 = 1^1^1^1 = 0$
1	1	1	0	-	-	-	$T2 = p2 \text{ xor } b3 \text{ xor } b2 \text{ xor } b1 = 0^1^1^1 = 1$

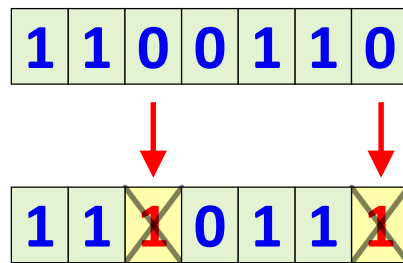
El test Hamming da **T2T1T0=101**,  
indicando que el error está en el bit 5.

Si **T2T1T0=000** no ha habido ningún error.

# Detección y Corrección de Errores

Código Hamming extendido (7, 3) + 1 bit de paridad.

- El código Hamming **no funciona** cuando se produce un **error en 2 bits**.



Error en los bits 1 y 5

b3	b2	b1	p2	b0	p1	p0	Test Hamming
7	6	5	4	3	2	1	
1	1	1	0	1	1	1	
1	-	1	-	1	-	1	$T0 = p0 \text{ xor } b3 \text{ xor } b1 \text{ xor } b0 = 1^1 1^1 1^1 = 0$
1	1	-	-	1	1	-	$T1 = p1 \text{ xor } b3 \text{ xor } b2 \text{ xor } b0 = 1^1 1^1 1^1 = 0$
1	1	1	0	-	-	-	$T2 = p2 \text{ xor } b3 \text{ xor } b2 \text{ xor } b1 = 0^1 1^1 1^1 = 1$

El test Hamming da **T2T1T0=100**,  
indicando que hay un error en el bit 4 (¿?).

- Añadiendo un bit de paridad es posible detectar un error en 2 bits o corregir un error de 1 bit.

# Detección y Corrección de Errores

Código Hamming extendido (7, 3) + 1 bit de paridad.

1 1 0 0 1 1 0 0

iPar!

1 1 1 0 1 1 1 0

iPar!

Error en los bits 1 y 5

No hay error de paridad.  
El test Hamming da  $T_2T_1T_0=100$   
Esto implica que se ha producido un error en 2 bits.

No se puede recuperar.

1 1 0 0 1 1 0 0

iPar!

1 1 1 0 1 1 0 1

iImpar!

Error en el bit 5

Error de paridad.  
El test Hamming da  $T_2T_1T_0=101$   
Esto implica que se ha producido un error en el bit 5.

Se puede recuperar.



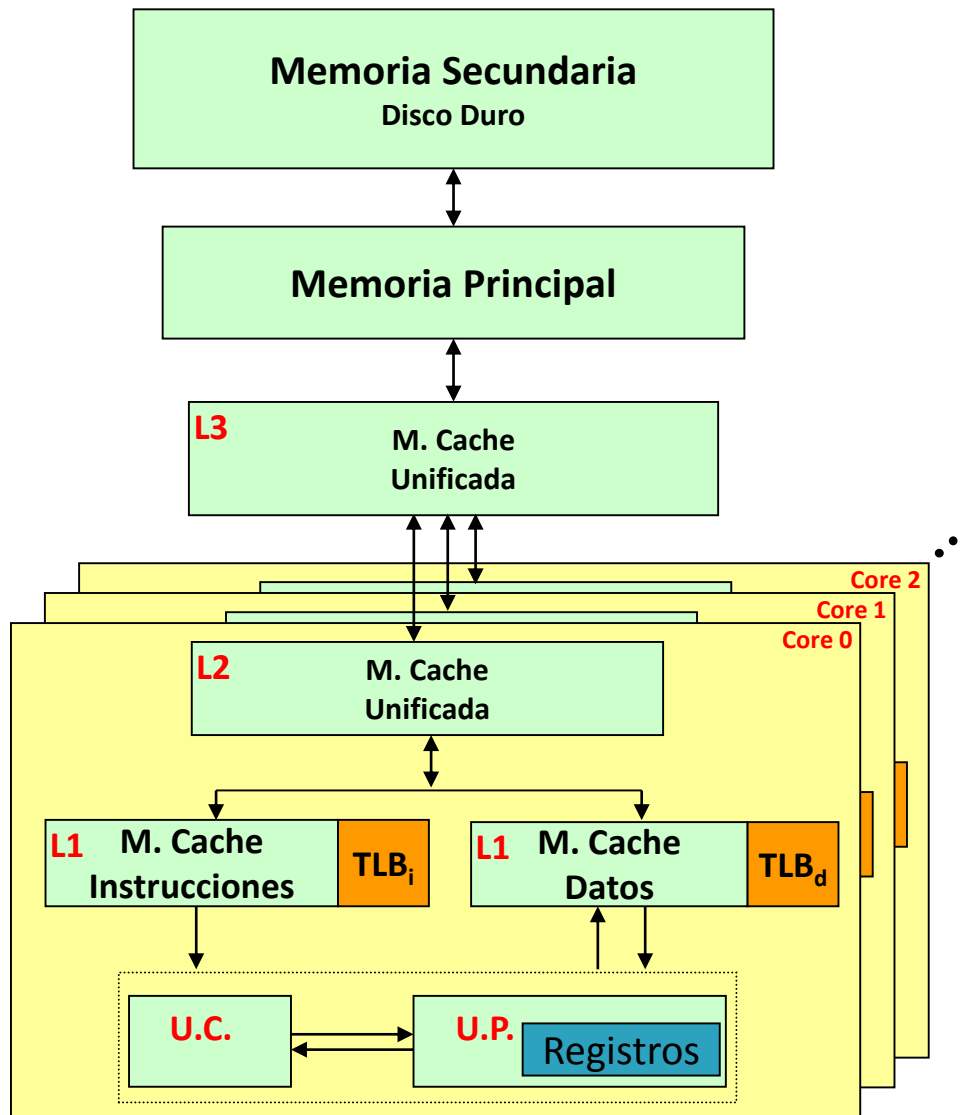
# Detección y Corrección de Errores

## Memoria ECC (Error Correction Code)

- La memoria que incluye estos mecanismos se utiliza exclusivamente en servidores y estaciones de trabajo de gama alta.
- Basado en **códigos de Hamming**.  
A cada porción de datos se añade información redundante que permite detectar y corregir errores.
- El código ECC más utilizado en memorias es el **SEC-DED** (Single Error Correction – Double Error Detection).  
Permite corregir 1 error o detectar 2 en la misma porción de datos.
- En DIMMs de memoria se suele utilizar un código extendido (71, 64) + bit de paridad.  
72 bits totales, con 64 bits de datos y 8 redundantes.  
Perfecto para utilizar en un DIMM con 9 chips de 8 bits de ancho.



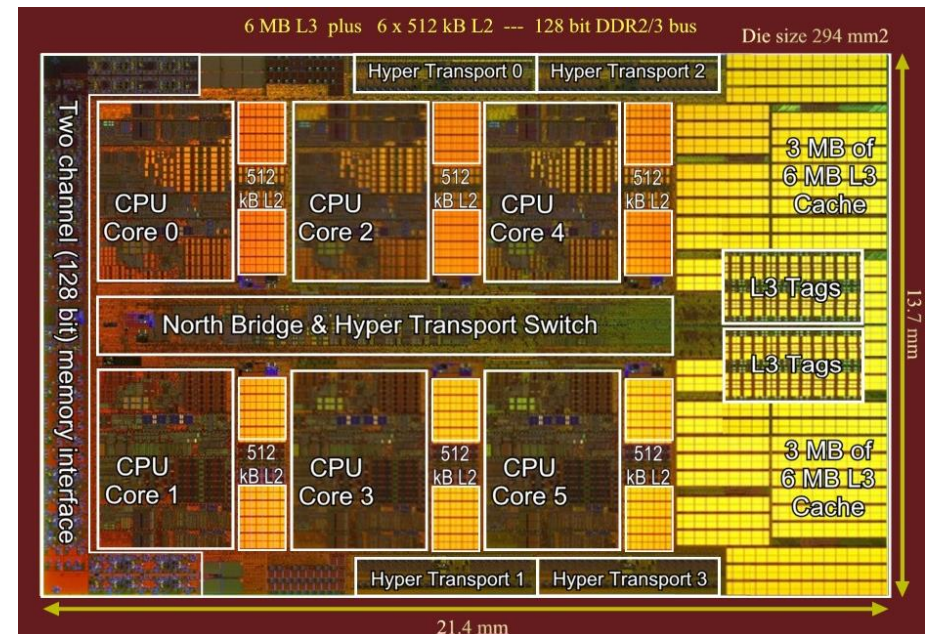
# Juntándolo todo



## Ejecución de instrucción **incl (%ebx)**

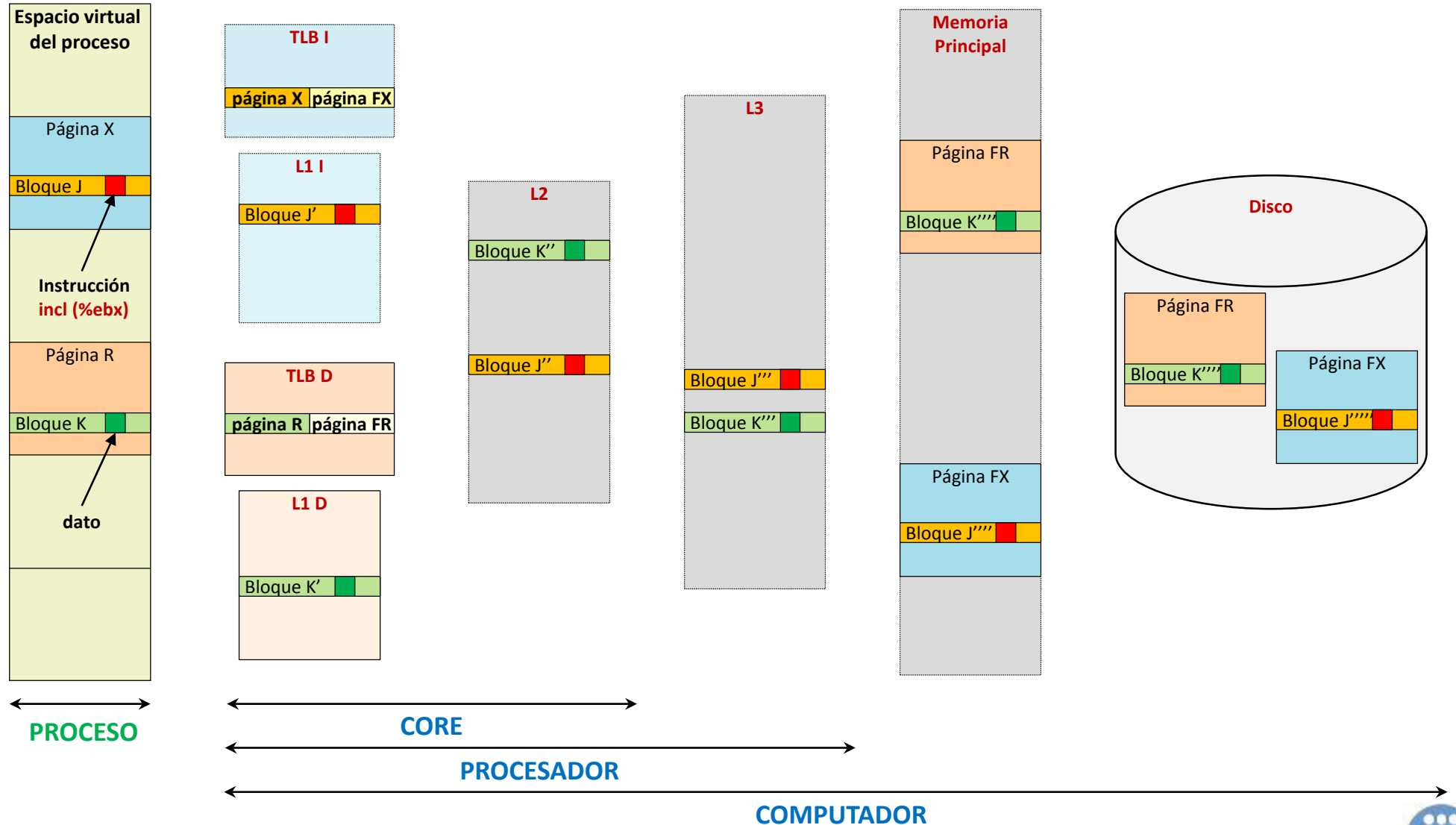
En el **PEOR CASO** puede provocar:

- Al leer instrucción: fallo TLBi, fallo página (acceso a disco), fallo L1 instrucciones, fallo L2, fallo L3
- Al leer dato: fallo TLBd, fallo página (acceso a disco), fallo L1 datos, fallo L2, fallo L3
- Al escribir dato: acierto TLBd, acierto L1 datos

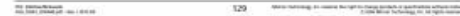
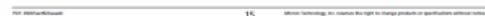


The Istanbul Opteron die. Source: AMD

# Juntándolo todo



- Así durante 207 páginas...



**Imágenes obtenidas en [www.micron.com](http://www.micron.com)**