



## Microarquitectura - El Sistema de Memoria

Alejandro Furfaro

7 de septiembre de 2023

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Visión Funcional

# Temario

## 1 El sistema de Memoria

### • Antecedentes

- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

### • Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

### 5 Memorias Dinámicas

- Introducción
- Organización interna

### 6 Standards

- Estado del arte
- JEDEC SDRAM

### 7 Controladores de Memoria

- Introducción General
- Arquitectura

### 8 Configuración

- Configuración del DRAM Device

### • Entrada Salida de Datos

### 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

### 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

### 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

### 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Evolución



Pioneros:

**Maurice Wilkes** en 1947 con la primer memoria de tanque de mercurio para la computadora EDSAC.

**Capacidad 2 bytes.**

# Evolución



Pioneros:

**Maurice Wilkes** en 1947 con la primer memoria de tanque de mercurio para la computadora EDSAC.

**Capacidad 2 bytes.**

Visionarios:

“640K debe ser suficiente memoria para cualquiera...”

Bill Gates. 1981

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# It's the Memory, Stupid!

# It's the Memory, Stupid!

Richard Sites, arquitecto senior de computadores en Digital Equipment Corporation (DEC), uno de los líderes del desarrollo del procesador Alfa, publicó en 1996 un reporte con este título provocador, en el prestigioso “Microprocessor Report”.

# It's the Memory, Stupid!

Richard Sites, arquitecto senior de computadores en Digital Equipment Corporation (DEC), uno de los líderes del desarrollo del procesador Alfa, publicó en 1996 un reporte con este título provocador, en el prestigioso “Microprocessor Report”.

- En 1992 DEC lanza el procesador Alfa 21064, con una CPU@200Mhz, superescalar de dos vías (envía dos instrucciones a ejecutar por ciclo de clock).

# It's the Memory, Stupid!

Richard Sites, arquitecto senior de computadores en Digital Equipment Corporation (DEC), uno de los líderes del desarrollo del procesador Alfa, publicó en 1996 un reporte con este título provocador, en el prestigioso “Microprocessor Report”.

- En 1992 DEC lanza el procesador Alfa 21064, con una CPU@200Mhz, superescalar de dos vías (envía dos instrucciones a ejecutar por ciclo de clock).
- En ese momento estiman una Prospección a 25 años (es decir a 2017):

# It's the Memory, Stupid!

Richard Sites, arquitecto senior de computadores en Digital Equipment Corporation (DEC), uno de los líderes del desarrollo del procesador Alfa, publicó en 1996 un reporte con este título provocador, en el prestigioso “Microprocessor Report”.

- En 1992 DEC lanza el procesador Alfa 21064, con una CPU@200Mhz, superescalar de dos vías (envía dos instrucciones a ejecutar por ciclo de clock).
- En ese momento estiman una Prospección a 25 años (es decir a 2017):
- Performance x1000,

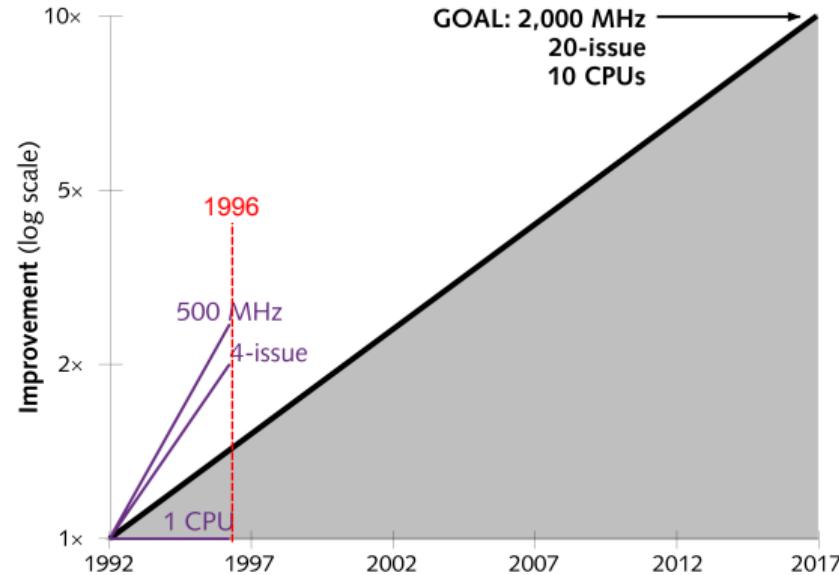
# It's the Memory, Stupid!

Richard Sites, arquitecto senior de computadores en Digital Equipment Corporation (DEC), uno de los líderes del desarrollo del procesador Alfa, publicó en 1996 un reporte con este título provocador, en el prestigioso “Microprocessor Report”.

- En 1992 DEC lanza el procesador Alfa 21064, con una CPU@200Mhz, superescalar de dos vías (envía dos instrucciones a ejecutar por ciclo de clock).
- En ese momento estiman una Prospección a 25 años (es decir a 2017):
- Performance x1000,
- Frecuencia de clock, # de instrucciones enviadas a ejecución por ciclo de clock, y # de cores x10

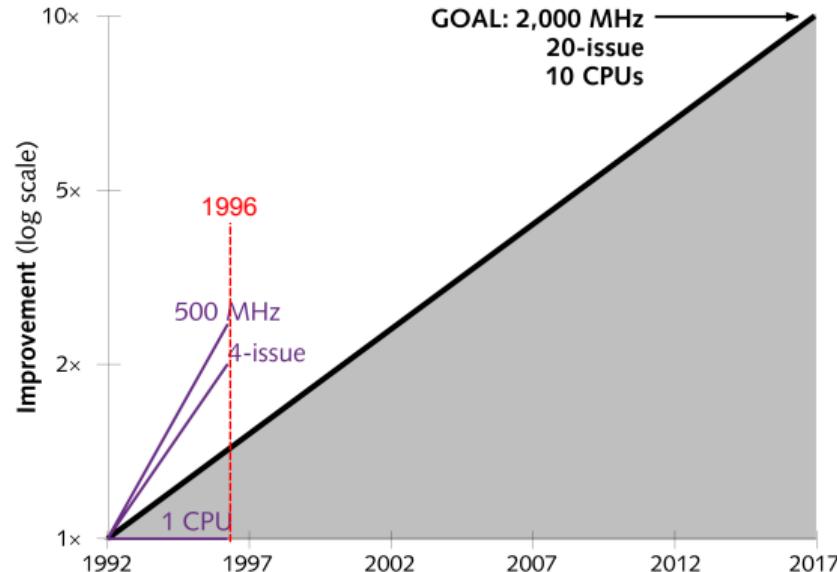
# La Prospección de DEC en 1992

- Según la prospección, 4 años después (1996), estas tres magnitudes deberían estar en promedio en x1.45.



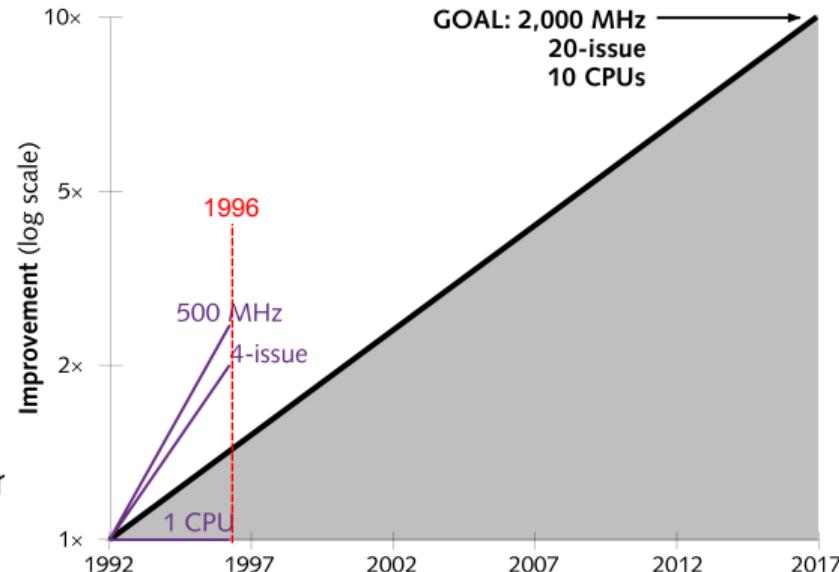
# La Prospección de DEC en 1992

- Según la prospección, 4 años después (1996), estas tres magnitudes deberían estar en promedio en x1.45.
- En 1996 Sites publica este artículo, en donde muestra que los pronósticos de estas tres magnitudes fueron conservadores.



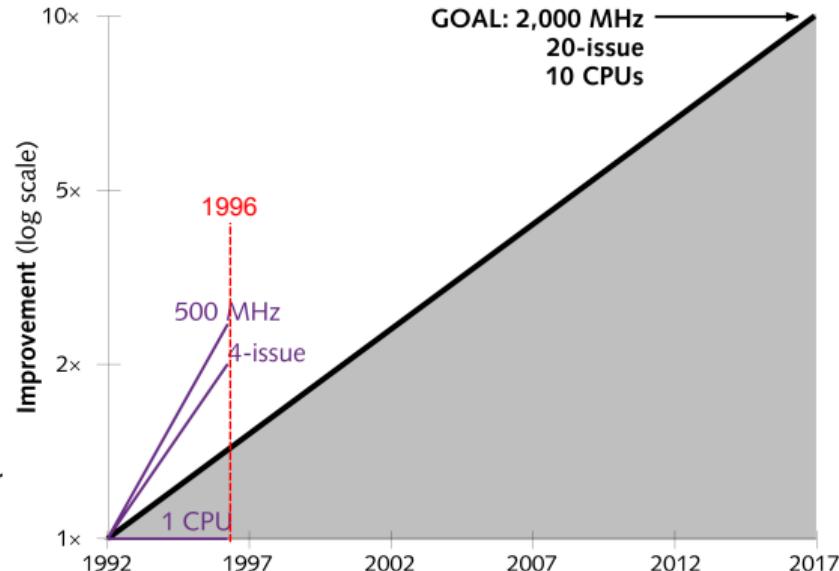
# La Prospección de DEC en 1992

- Según la prospección, 4 años después (1996), estas tres magnitudes deberían estar en promedio en x1.45.
- En 1996 Sites publica este artículo, en donde muestra que los pronósticos de estas tres magnitudes fueron conservadores.
- El Alfa 21164 tenía 1 CPU@500Mhz y enviaba 4 irías.



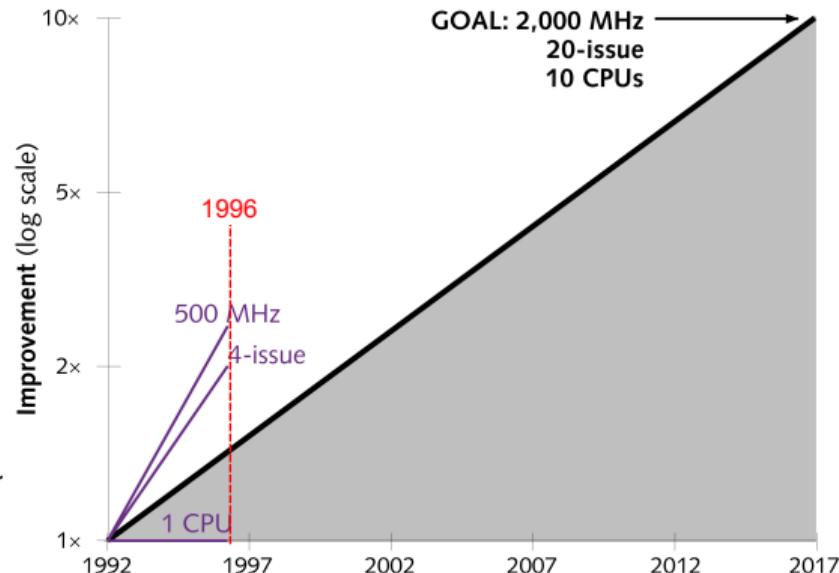
# La Prospección de DEC en 1992

- Según la prospección, 4 años después (1996), estas tres magnitudes deberían estar en promedio en x1.45.
- En 1996 Sites publica este artículo, en donde muestra que los pronósticos de estas tres magnitudes fueron conservadores.
- El Alfa 21164 tenía 1 CPU@500Mhz y enviaba 4 irías.
- Su frecuencia incrementó x2.5, la cantidad de instrucciones enviadas x2, y solo la cantidad de CPUs quedó atrás, con respecto a la situación de 1992.



# La Prospección de DEC en 1992

- Según la prospección, 4 años después (1996), estas tres magnitudes deberían estar en promedio en x1.45.
- En 1996 Sites publica este artículo, en donde muestra que los pronósticos de estas tres magnitudes fueron conservadores.
- El Alfa 21164 tenía 1 CPU@500Mhz y enviaba 4 irías.
- Su frecuencia incrementó x2.5, la cantidad de instrucciones enviadas x2, y solo la cantidad de CPUs quedó atrás, con respecto a la situación de 1992.
- Si con los datos actuales miramos la prospección a 2017, de las tres magnitudes por separado, concluimos en que el pronóstico tenía mucho sentido.



# Todo parece estar perfecto. Pero ¿Y la performance?

# Todo parece estar perfecto. Pero ¿Y la performance?

- La performance, resultaría una historia bastante diferente...sin dudas.

# Todo parece estar perfecto. Pero ¿Y la performance?

- La performance, resultaría una historia bastante diferente...sin dudas.
- Para medirla, utilizaron el benchmark de TPC-C (Transaction Processing Performance Council. C), utilizando bases de datos en dos equipos. Uno basado en un procesador Alfa 21164@400Mhz, y el otro en un Pentium-Pro@200Mhz.

# Todo parece estar perfecto. Pero ¿Y la performance?

- La performance, resultaría una historia bastante diferente... sin dudas.
- Para medirla, utilizaron el benchmark de TPC-C (Transaction Processing Performance Council. C), utilizando bases de datos en dos equipos. Uno basado en un procesador Alfa 21164@400Mhz, y el otro en un Pentium-Pro@200Mhz.
- Resultado: El Alfa entregaba un resultado cada 4.2 ciclos de clock, y el Pentium Pro uno cada 4.5.

# Todo parece estar perfecto. Pero ¿Y la performance?

- La performance, resultaría una historia bastante diferente... sin dudas.
- Para medirla, utilizaron el benchmark de TPC-C (Transaction Processing Performance Council. C), utilizando bases de datos en dos equipos. Uno basado en un procesador Alfa 21164@400Mhz, y el otro en un Pentium-Pro@200Mhz.
- Resultado: El Alfa entregaba un resultado cada 4.2 ciclos de clock, y el Pentium Pro uno cada 4.5.
- Dicho de otro modo, los procesadores están tres ciclos de clock esperando a la memoria.

# ¿Conclusión?

# ¿Conclusión?

El título de su artículo es una síntesis muy realista de la sensación que experimentamos al descubrir algo, apriori no tan evidente...

# ¿Conclusión?

El título de su artículo es una síntesis muy realista de la sensación que experimentamos al descubrir algo, apriori no tan evidente...

It's the Memory, Stupid!

# Temario

- 1 El sistema de Memoria
  - Antecedentes
  - Rol de la memoria en un computador
  - Jerarquía de memoria
- 2 Tecnologías de Memoria
  - Clasificación
  - Memorias No volátiles
  - Memorias Volátiles
  - Memorias y velocidad del Procesador
- 3 Memoria Cache
  - Principio de Funcionamiento
  - Métricas y Performance
  - Hardware dedicado = + complejidad
  - organización de un cache
- 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real
- 5 Memorias Dinámicas
  - Introducción
  - Organización interna
- 6 Standards
  - Estado del arte
  - JEDEC SDRAM
- 7 Controladores de Memoria
  - Introducción General
  - Arquitectura
- 8 Configuración
  - Configuración del DRAM Device
- Entrada Salida de Datos
- 9 Integración con el sistema Cache
  - Evitar cuellos de botella es la clave
- 10 Casos Prácticos
  - Beagle Bone Black
  - Memorias DDR en la BBB
  - Controlador de DDRn SDRAM en la BBB
- 11 SRAM Cuestiones de Implementación
  - Vistazo introductorio
  - Decodificación
  - Implementación
- 12 DRAM Detalles de implementación
  - Acceso a las celdas
  - JEDEC DDR SDRAM
  - Protocolo de acceso

# Memoria: Mas que un componente, un subsistema

# Memoria: Mas que un componente, un subsistema

- El sueño: tener un sistema de Memoria de una sola tecnología y sin jerarquía. Diseño simple, costo moderado. Fin.

# Memoria: Mas que un componente, un subsistema

- El sueño: tener un sistema de Memoria de una sola tecnología y sin jerarquía. Diseño simple, costo moderado. Fin.
- La realidad: Solo posible en sistemas triviales basados en Microcontroladores para actividades de control rudimentarias.

# Memoria: Mas que un componente, un subsistema

- El sueño: tener un sistema de Memoria de una sola tecnología y sin jerarquía. Diseño simple, costo moderado. Fin.
- La realidad: Solo posible en sistemas triviales basados en Microcontroladores para actividades de control rudimentarias.
- En un sistema de cómputo un poco mas sofisticado no hay alternativas. Necesitamos mirar la memoria como un subsistema.

# Memoria: Mas que un componente, un subsistema

- El sueño: tener un sistema de Memoria de una sola tecnología y sin jerarquía. Diseño simple, costo moderado. Fin.
- La realidad: Solo posible en sistemas triviales basados en Microcontroladores para actividades de control rudimentarias.
- En un sistema de cómputo un poco mas sofisticado no hay alternativas. Necesitamos mirar la memoria como un subsistema.
- A punto tal que es mas importante el diseño de este subsistema que el de la propia CPU.

# Memoria: Mas que un componente, un subsistema

- El sueño: tener un sistema de Memoria de una sola tecnología y sin jerarquía. Diseño simple, costo moderado. Fin.
- La realidad: Solo posible en sistemas triviales basados en Microcontroladores para actividades de control rudimentarias.
- En un sistema de cómputo un poco mas sofisticado no hay alternativas. Necesitamos mirar la memoria como un subsistema.
- A punto tal que es mas importante el diseño de este subsistema que el de la propia CPU.
- Los avances en scaling, y frecuencia de señal, han vuelto sumamente complejo el diseño de este subsistema.

# Memoria: Mas que un componente, un subsistema

- El sueño: tener un sistema de Memoria de una sola tecnología y sin jerarquía. Diseño simple, costo moderado. Fin.
- La realidad: Solo posible en sistemas triviales basados en Microcontroladores para actividades de control rudimentarias.
- En un sistema de cómputo un poco mas sofisticado no hay alternativas. Necesitamos mirar la memoria como un subsistema.
- A punto tal que es mas importante el diseño de este subsistema que el de la propia CPU.
- Los avances en scaling, y frecuencia de señal, han vuelto sumamente complejo el diseño de este subsistema.
- Afloraron así, en el diseño del subsistema de memoria problemas que hasta hace 20 años no eran relevantes. Ejemplo: La integridad de señal entre los diferentes tipos y niveles jerárquicos de memoria y la CPU.

# Diseño de un subsistema de memoria

Los requerimientos de un subsistema de memoria plantean condiciones que resultan mutuamente excluyentes:

# Diseño de un subsistema de memoria

Los requerimientos de un subsistema de memoria plantean condiciones que resultan mutuamente excluyentes:

- Gran capacidad de almacenamiento.

# Diseño de un subsistema de memoria

Los requerimientos de un subsistema de memoria plantean condiciones que resultan mutuamente excluyentes:

- Gran capacidad de almacenamiento.
- Tiempo de acceso mínimo.

# Diseño de un subsistema de memoria

Los requerimientos de un subsistema de memoria plantean condiciones que resultan mutuamente excluyentes:

- Gran capacidad de almacenamiento.
- Tiempo de acceso mínimo.
- Capacidad de mantener los datos cuando se apaga el equipo.

# Diseño de un subsistema de memoria

Los requerimientos de un subsistema de memoria plantean condiciones que resultan mutuamente excluyentes:

- Gran capacidad de almacenamiento.
- Tiempo de acceso mínimo.
- Capacidad de mantener los datos cuando se apaga el equipo.
- Finalmente en ninguna lista aspiracional puede faltar: **Bajo costo.**

# Diseño de un subsistema de memoria

Los requerimientos de un subsistema de memoria plantean condiciones que resultan mutuamente excluyentes:

- Gran capacidad de almacenamiento.
- Tiempo de acceso mínimo.
- Capacidad de mantener los datos cuando se apaga el equipo.
- Finalmente en ninguna lista aspiracional puede faltar: **Bajo costo**.

Típico escenario de “manta corta”. Imposible de lograr con una sola tecnología, y sin una organización por niveles.

# Diseño de un subsistema de memoria

Los requerimientos de un subsistema de memoria plantean condiciones que resultan mutuamente excluyentes:

- Gran capacidad de almacenamiento.
- Tiempo de acceso mínimo.
- Capacidad de mantener los datos cuando se apaga el equipo.
- Finalmente en ninguna lista aspiracional puede faltar: **Bajo costo**.

Típico escenario de “manta corta”. Imposible de lograr con una sola tecnología, y sin una organización por niveles.

Conclusión: **Necesitamos pensar en un sistema de memoria basado en niveles jerárquicos.**

# ¿Como construir una jerarquía?

# ¿Como construir una jerarquía?

- El principio rector de la jerarquía de memoria se conoce como localidad de referencia.

# ¿Como construir una jerarquía?

- El principio rector de la jerarquía de memoria se conoce como localidad de referencia.
- Se basa en dos trabajos de investigación de intercambio de bloques de memoria y storage cuando se desarrollaron los algoritmos para manejo de memoria virtual.

# ¿Como construir una jerarquía?

- El principio rector de la jerarquía de memoria se conoce como localidad de referencia.
- Se basa en dos trabajos de investigación de intercambio de bloques de memoria y storage cuando se desarrollaron los algoritmos para manejo de memoria virtual.
- En 1966 L. A. Belady, publicó en IBM System Journal un estudio muy comprensivo y esclarecedor acerca del funcionamiento real del acceso a memoria de un computador.

# ¿Como construir una jerarquía?

- El principio rector de la jerarquía de memoria se conoce como localidad de referencia.
- Se basa en dos trabajos de investigación de intercambio de bloques de memoria y storage cuando se desarrollaron los algoritmos para manejo de memoria virtual.
- En 1966 L. A. Belady, publicó en IBM System Journal un estudio muy comprensivo y esclarecedor acerca del funcionamiento real del acceso a memoria de un computador.
- En pocas palabras, sus conclusiones fueron: *el acceso a memoria no es tan aleatorio como el “Random Access Memory” lo sugiere, sino que por el contrario es bastante predecible.*

# Principio de localidad

# Principio de localidad

## Observación empírica

Si tomamos cualquier programa razonablemente bien escrito, que respete lo que llamamos “buenas prácticas de programación”, se observa claramente que los accesos a memoria quedan restringidos durante un intervalo de tiempo significativo a un mismo intervalo de direcciones de memoria contiguas. Dicho de manera mas directa: los programas tienden a reutilizar datos e instrucciones que han utilizado recientemente.

# Principio de localidad

## Localidad temporal

Se refiere a un patrón de acceso a las mismas direcciones de memoria en un intervalo temporal finito y acotado.

La probabilidad que la CPU solicite en los próximos ciclos de acceso a memoria, las mismas direcciones de memoria que está utilizando actualmente, es muy alta.

En cambio, es muy baja la probabilidad de iniciar un ciclo de memoria hacia direcciones que no están siendo actualmente utilizadas.

# Principio de localidad

## Localidad espacial

Se refiere al espacio de direcciones que utiliza una CPU en un intervalo temporal finito y acotado. La probabilidad que la CPU utilice direcciones vecinas a la del objeto que está direccionando actualmente es muy alta.

Contrariamente es muy poco probable que utilice direcciones de memoria lejas a las actuales.

# Ejemplo: Algoritmo de Convolución

```
1   for (i = 0 ; i < 256 ; i++)
2   {
3       suma = 0.0f;
4       for (j = 0 ; (j <= i && j < 256) ; j++)
5           suma += v0[i-j] * v1[j];
6       fAux[i] = suma;
7   }
```

## Ejemplo: Algoritmo de Convolución

```
1   for (i = 0 ; i < 256 ; i++)
2   {
3       suma = 0.0f;
4       for (j = 0 ; (j <= i && j < 256) ; j++)
5           suma += v0[i-j] * v1[j];
6       fAux[i] = suma;
7   }
```

- Las variables *i, j, suma*, cumplen el principio de vecindad temporal. Sus direcciones se acceden con una frecuencia muy alta.

## Ejemplo: Algoritmo de Convolución

```
1   for (i = 0 ; i < 256 ; i++)
2   {
3       suma = 0.0f;
4       for (j = 0 ; (j <= i && j < 256) ; j++)
5           suma += v0[i-j] * v1[j];
6       fAux[i] = suma;
7   }
```

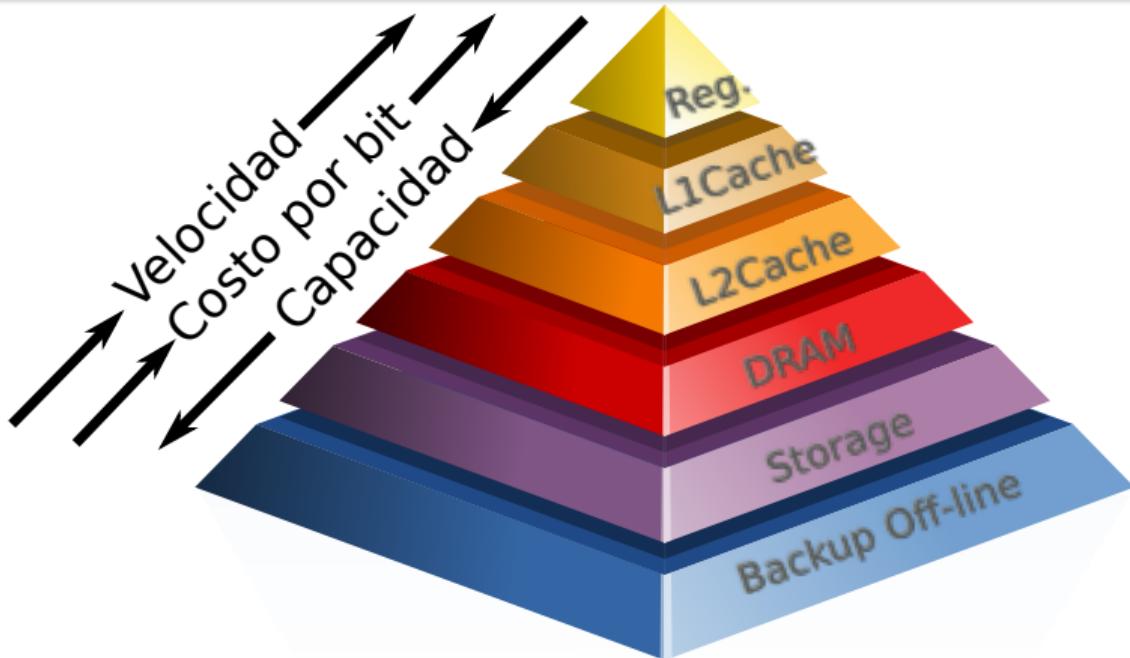
- Las variables *i, j, suma*, cumplen el principio de vecindad temporal. Sus direcciones se acceden con una frecuencia muy alta.
- El bloque de código verifica el principio de localidad espacial. La probabilidad de acceso a ese espacio de direcciones es alta mientras no haya conmutación de tareas, y no se anule el contador del loop exterior.

## Ejemplo: Algoritmo de Convolución

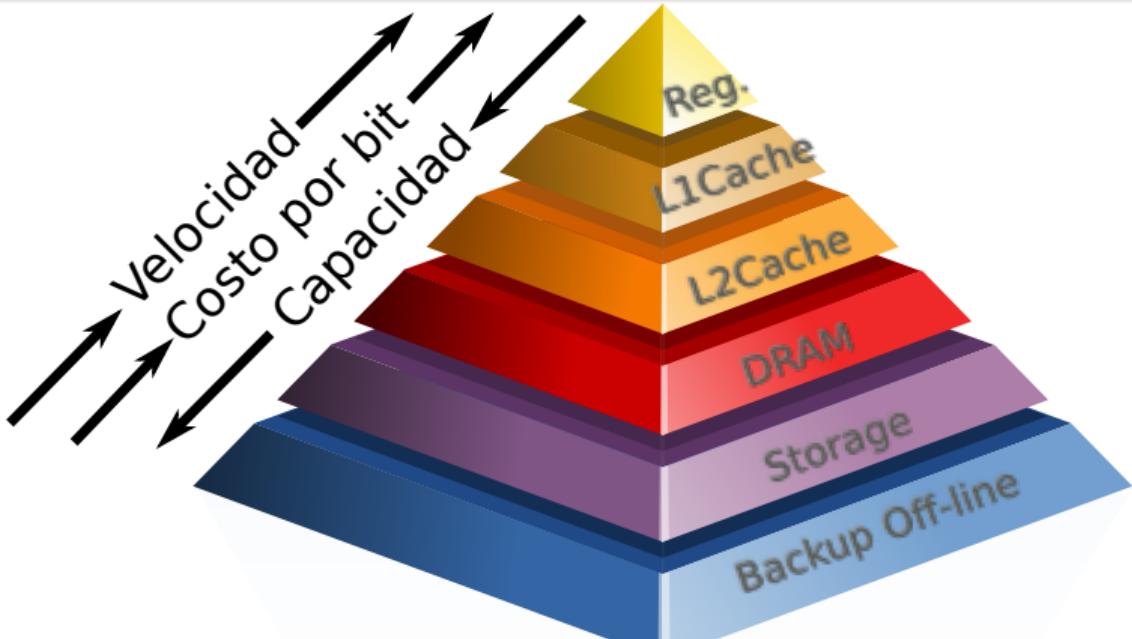
```
1   for (i = 0 ; i < 256 ; i++)
2   {
3       suma = 0.0f;
4       for (j = 0 ; (j <= i && j < 256) ; j++)
5           suma += v0[i-j] * v1[j];
6       fAux[i] = suma;
7   }
```

- Las variables *i, j, suma*, cumplen el principio de vecindad temporal. Sus direcciones se acceden con una frecuencia muy alta.
- El bloque de código verifica el principio de localidad espacial. La probabilidad de acceso a ese espacio de direcciones es alta mientras no haya conmutación de tareas, y no se anule el contador del loop exterior.
- **Idea fuerza:** Si logramos que este código y las variables estén en una memoria rápida cuando el procesador los necesite, su tiempo de acceso será mínimo (óptimo).

# Memoria: jerarquía piramidal



# Memoria: jerarquía piramidal



El sistema tiende más a un comportamiento ideal cuando los datos y el código que está utilizando se encuentran más cerca del vértice (memorias más rápidas).

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Tecnologías, Características, y jerarquía

# Tecnologías, Características, y jerarquía

- Hay diferentes criterios para clasificar memorias.

# Tecnologías, Características, y jerarquía

- Hay diferentes criterios para clasificar memorias.
- Algunos anacrónicos (ROM vs RAM)

# Tecnologías, Características, y jerarquía

- Hay diferentes criterios para clasificar memorias.
- Algunos anacrónicos (ROM vs RAM)
- Lo interesante es entender que rol juegan las diferentes tecnologías de memoria y su relación con la ubicación en la jerarquía.

# Tecnologías, Características, y jerarquía

- Hay diferentes criterios para clasificar memorias.
- Algunos anacrónicos (ROM vs RAM)
- Lo interesante es entender que rol juegan las diferentes tecnologías de memoria y su relación con la ubicación en la jerarquía.
- Veamos sino...

# Memorias No volátiles

# Memorias No volátiles

- Son dispositivos capaces de retener la información almacenada cuando se les desconecta la alimentación.

# Memorias No volátiles

- Son dispositivos capaces de retener la información almacenada cuando se les desconecta la alimentación.
- Han evolucionado tecnológicamente, partiendo de dispositivos grabados en fábrica, paulatinamente permitieron su modificación off-line, hasta llegar a los dispositivos actuales que pueden modificarse en tiempo real.

# Memorias No volátiles

- Son dispositivos capaces de retener la información almacenada cuando se les desconecta la alimentación.
- Han evolucionado tecnológicamente, partiendo de dispositivos grabados en fábrica, paulatinamente permitieron su modificación off-line, hasta llegar a los dispositivos actuales que pueden modificarse en tiempo real.
- Es anacrónica la denominación ROM (por **Read Only Memory**), ya que pueden ser modificadas, si bien no por medio de escrituras de memoria convencionales. Pero se puede escribir en ellas.

# Memorias Volátiles

# Memorias Volátiles

- Se caracterizan por que una vez interrumpida la alimentación eléctrica, la información que almacenaban se pierde.

# Memorias Volátiles

- Se caracterizan por que una vez interrumpida la alimentación eléctrica, la información que almacenaban se pierde.
- Conocidas como **RAM** (por **R**andom **A**ccess **M**emory), debido a que su tiempo de acceso es siempre el mismo independientemente de la ubicación del dato en la memoria.

# Memorias Volátiles

- Se caracterizan por que una vez interrumpida la alimentación eléctrica, la información que almacenaban se pierde.
- Conocidas como **RAM** (por **R**andom **A**ccess **M**emory), debido a que su tiempo de acceso es siempre el mismo independientemente de la ubicación del dato en la memoria.
- La cinta magnética, tenía acceso secuencial. El tiempo de acceso al dato dependía de su ubicación en la cinta.

# Memorias Volátiles

- Se caracterizan por que una vez interrumpida la alimentación eléctrica, la información que almacenaban se pierde.
- Conocidas como **RAM** (por **R**andom **A**ccess **M**emory), debido a que su tiempo de acceso es siempre el mismo independientemente de la ubicación del dato en la memoria.
- La cinta magnética, tenía acceso secuencial. El tiempo de acceso al dato dependía de su ubicación en la cinta.
- El término aleatorio asignado a las memorias de estado sólido, sugiere que tiempo de acceso no depende de la dispersión de las direcciones que se van solicitando.

# Memorias Volátiles

- Se caracterizan por que una vez interrumpida la alimentación eléctrica, la información que almacenaban se pierde.
- Conocidas como **RAM** (por **R**andom **A**ccess **M**emory), debido a que su tiempo de acceso es siempre el mismo independientemente de la ubicación del dato en la memoria.
- La cinta magnética, tenía acceso secuencial. El tiempo de acceso al dato dependía de su ubicación en la cinta.
- El término aleatorio asignado a las memorias de estado sólido, sugiere que tiempo de acceso no depende de la dispersión de las direcciones que se van solicitando.
- Tienen mayor capacidad que las No Volátiles, y pueden modificarse en tiempo real a gran velocidad.

# Memorias Volátiles

- Se caracterizan por que una vez interrumpida la alimentación eléctrica, la información que almacenaban se pierde.
- Conocidas como **RAM** (por **R**andom **A**ccess **M**emory), debido a que su tiempo de acceso es siempre el mismo independientemente de la ubicación del dato en la memoria.
- La cinta magnética, tenía acceso secuencial. El tiempo de acceso al dato dependía de su ubicación en la cinta.
- El término aleatorio asignado a las memorias de estado sólido, sugiere que tiempo de acceso no depende de la dispersión de las direcciones que se van solicitando.
- Tienen mayor capacidad que las No Volátiles, y pueden modificarse en tiempo real a gran velocidad.
- Se clasifican de acuerdo con la tecnología y su diseño interno en dinámicas (**DRAM**) y estáticas (**SRAM**).

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- **Memorias No volátiles**
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Memorias No volátiles: Evolución

# Memorias No volátiles: Evolución

- Primera generación. Memorias denominadas **ROM** (por **R**ead **O**nly **M**emory). Debían ser grabadas por el fabricante del chip, y no eran modificables.

# Memorias No volátiles: Evolución

- Primera generación. Memorias denominadas **ROM** (por **R**ead **O**nly **M**emory). Debían ser grabadas por el fabricante del chip, y no eran modificables.
- Segunda Generación. Componentes programables solo una vez (**OTPROM**).

# Memorias No volátiles: Evolución

- Primera generación. Memorias denominadas **ROM** (por **R**ead **O**nly **M**emory). Debían ser grabadas por el fabricante del chip, y no eran modificables.
- Segunda Generación. Componentes programables solo una vez (**OTPROM**).
- Tercera Generación. Programables y borrables con luz ultravioleta de una determinada longitud de onda (**EPROM**).

# Memorias No volátiles: Evolución

- Primera generación. Memorias denominadas **ROM** (por **R**ead **O**nly **M**emory). Debían ser grabadas por el fabricante del chip, y no eran modificables.
- Segunda Generación. Componentes programables solo una vez (**OTPROM**).
- Tercera Generación. Programables y borrables con luz ultravioleta de una determinada longitud de onda (**EPROM**).
- Cuarta Generación. Programables y borrables Eléctricamente (**EEPROM**).

# Memorias No volátiles: **EEPROM**

En la actualidad se denominan Flash por su tecnología de origen. Pueden ser grabadas “on the fly” mediante algoritmos de escritura específicos. Su ejemplo mas habitual son los discos de estado sólido de los equipos portátiles modernos, las tarjetas SD, los pen drives, y las memorias que componen el firmware de cualquier equipo de computo, o cualquier dispositivo consumer.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles**
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

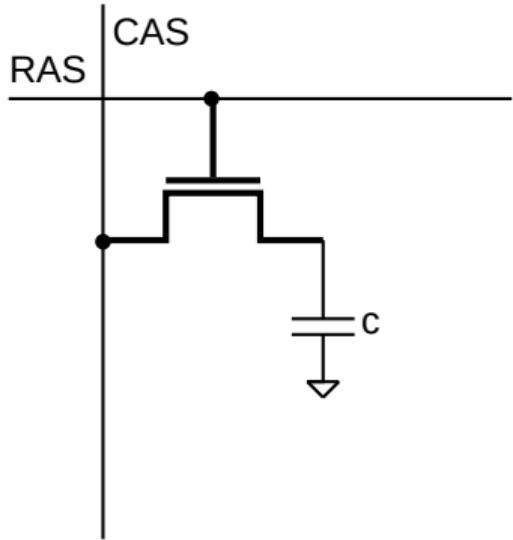
## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

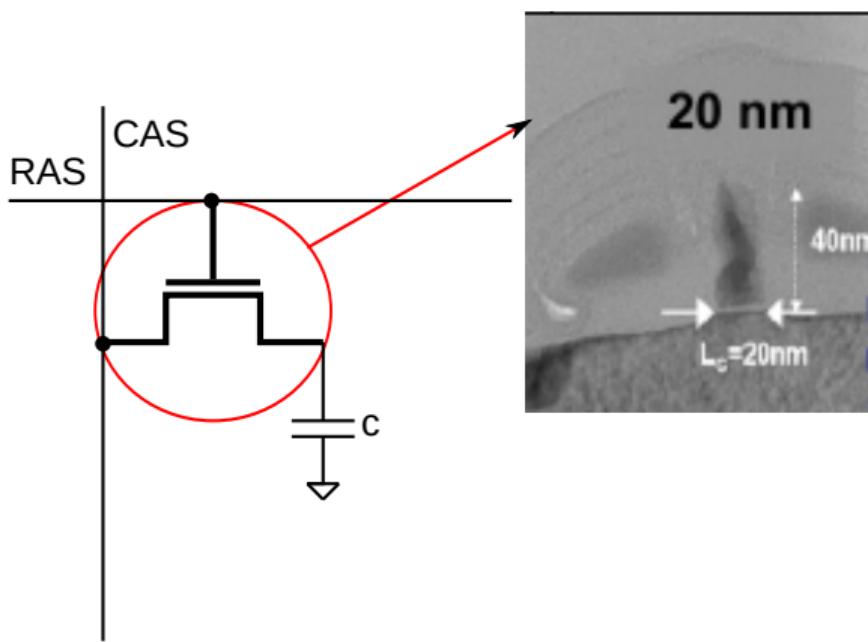
- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Memorias Volátiles dinámicas (DRAM)



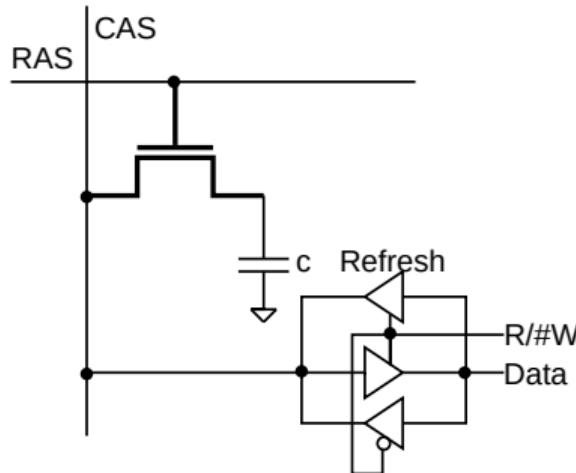
- Almacena la información en forma de estado de carga en un capacitor y la sostiene durante un breve lapso con la ayuda de un transistor.
- Una celda (un bit) se implementa con un solo transistor => máxima capacidad de almacenamiento por CI.
- Ese transistor está generalmente en estado de Corte, por lo cual consume mínima energía.

# Memorias Volátiles dinámicas (DRAM)



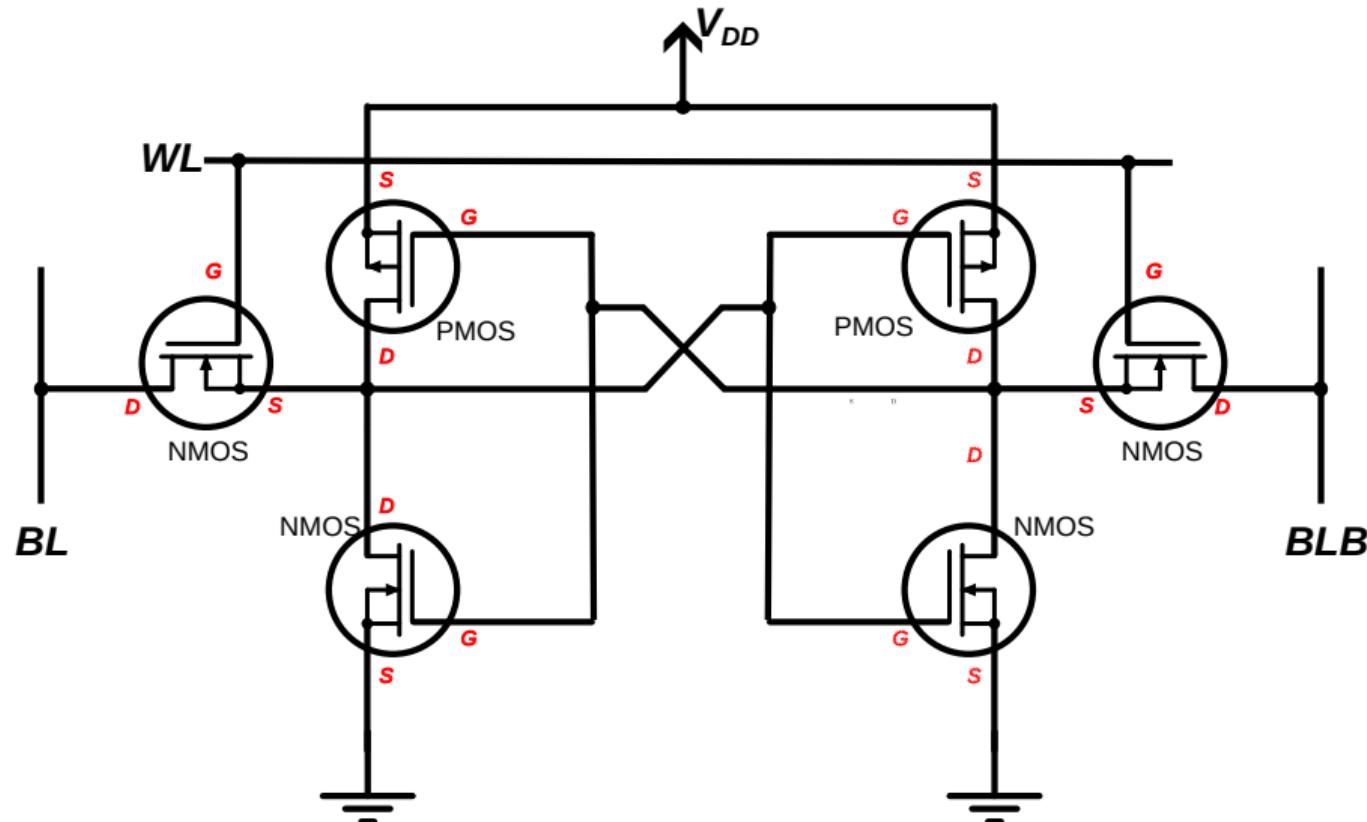
- Almacena la información en forma de estado de carga en un capacitor y la sostiene durante un breve lapso con la ayuda de un transistor.
- Una celda (un bit) se implementa con un solo transistor => máxima capacidad de almacenamiento por CI.
- Ese transistor está generalmente en estado de Corte, por lo cual consume mínima energía.

# Memorias volátiles dinámicas (Nada es perfecto)



- Al leer el bit, se descarga la capacidad (lectura destructiva).
- Necesita regenerar la carga** cada vez que se la lee.
- Esta operación se realiza por realimentación mediante buffers.
- Aumenta entonces el tiempo total que demanda el acceso de la celda. Cada operación incluye la carga del capacitor.
- Aun sin ser accedida el capacitor se descarga (Impedancia finita). Requiere Refresco.

# Memorias Volátiles estáticas (SRAM)



# Memorias Volátiles estáticas (SRAM)

- Almacena el estado de cada bit en un biestable.

# Memorias Volátiles estáticas (SRAM)

- Almacena el estado de cada bit en un biestable.
- Una celda (un bit) se compone de seis transistores. Por lo tanto tiene menor capacidad de almacenamiento por Cl.

# Memorias Volátiles estáticas (SRAM)

- Almacena el estado de cada bit en un biestable.
- Una celda (un bit) se compone de seis transistores. Por lo tanto tiene menor capacidad de almacenamiento por Cl.
- Tres de los seis transistores están saturados (conducen la máxima corriente posible en forma permanente) y los otros tres al corte (conducen una corriente prácticamente insignificante, pero no nula). Esto genera mayor consumo de energía por celda.

# Memorias Volátiles estáticas (SRAM)

- Almacena el estado de cada bit en un biestable.
- Una celda (un bit) se compone de seis transistores. Por lo tanto tiene menor capacidad de almacenamiento por Cl.
- Tres de los seis transistores están saturados (conducen la máxima corriente posible en forma permanente) y los otros tres al corte (conducen una corriente prácticamente insignificante, pero no nula). Esto genera mayor consumo de energía por celda.
- La lectura es directa y no destructiva por lo cual el tiempo de acceso es muy bajo en comparación con las memorias dinámicas. De hecho, luego de los registros del procesador, son el medio de almacenamiento de menor tiempo de acceso.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- **Memorias y velocidad del Procesador**

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Uso de las memorias en un computador

# Uso de las memorias en un computador

- La memoria no volátil se usa fundamentalmente para almacenar el programa de arranque de cualquier sistema. Le seguimos diciendo **ROM**, mas por costumbre que por corrección técnica.

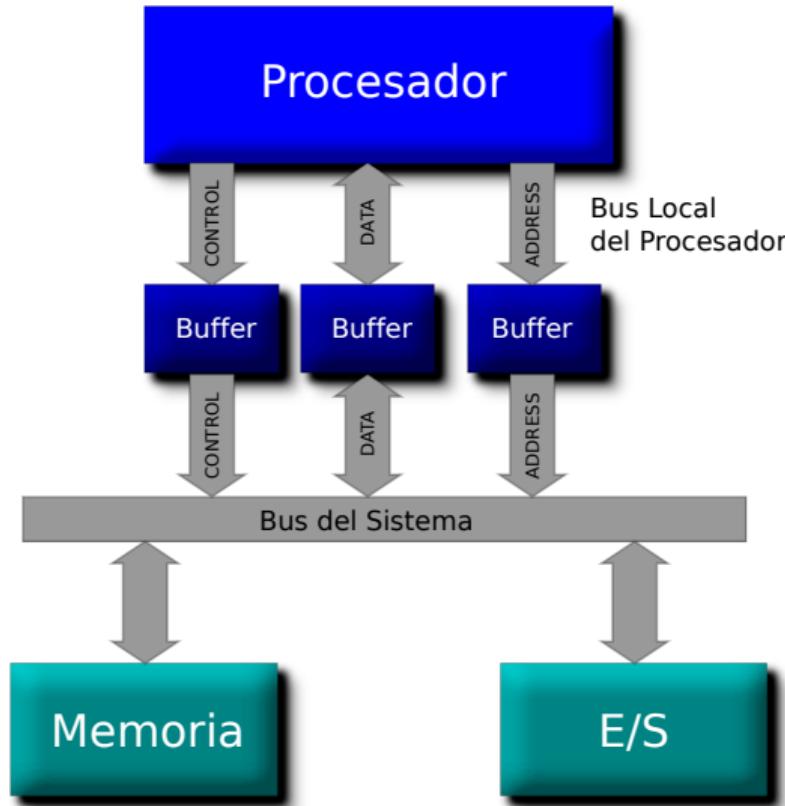
# Uso de las memorias en un computador

- La memoria no volátil se usa fundamentalmente para almacenar el programa de arranque de cualquier sistema. Le seguimos diciendo **ROM**, mas por costumbre que por corrección técnica.
- Se conectan en un espacio de direcciones determinado por el propio microprocesador de acuerdo a la dirección en la que éste irá a buscar la primer instrucción luego de encender el equipo.

# Uso de las memorias en un computador

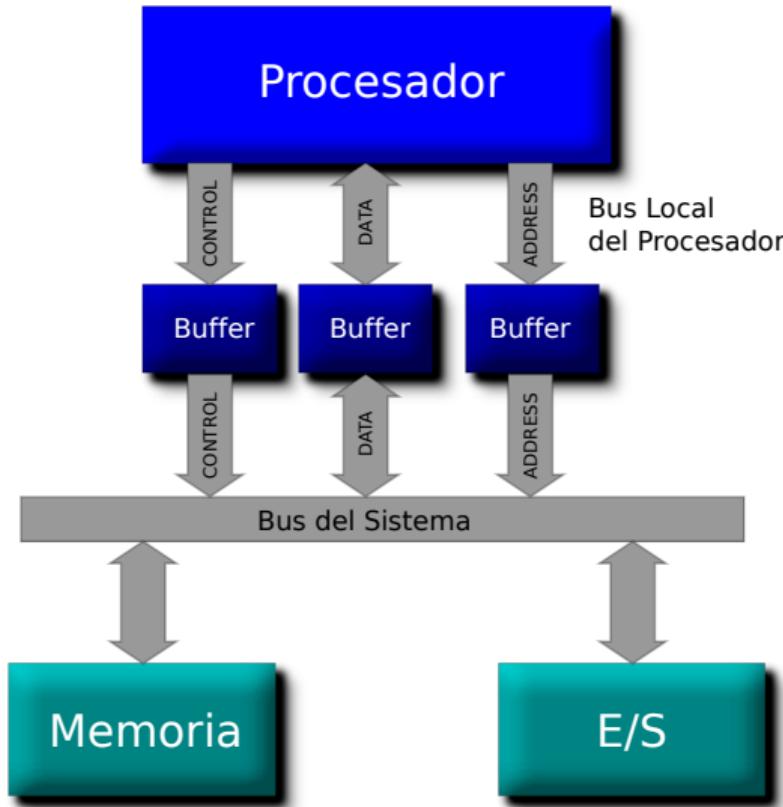
- La memoria no volátil se usa fundamentalmente para almacenar el programa de arranque de cualquier sistema. Le seguimos diciendo **ROM**, mas por costumbre que por corrección técnica.
- Se conectan en un espacio de direcciones determinado por el propio microprocesador de acuerdo a la dirección en la que éste irá a buscar la primer instrucción luego de encender el equipo.
- El resto es **RAM** y allí el sistema copia incluso buena parte del código de arranque residente en la memoria No Volátil, para que se ejecute mas rápido ya que las memorias volátiles tienen menor tiempo de acceso, (menos ciclos de clock para acceder al contenido de una dirección).

# Conexión básica (Modelo clásico de Von Newmann)



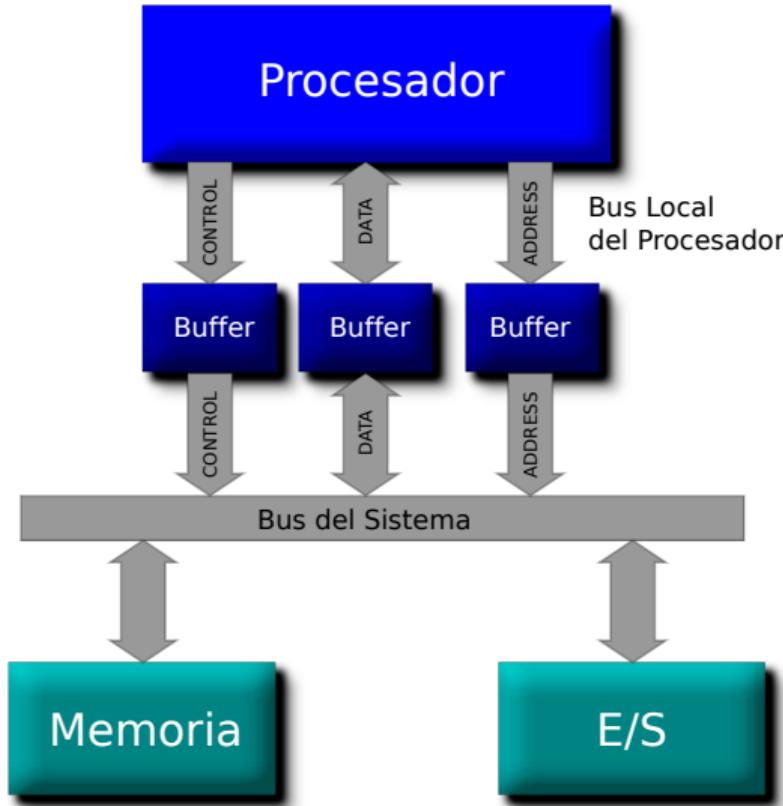
- Desde los años 80, la mejora de velocidad de las CPU comienzan a superar año a año a los tiempos de acceso a memoria DRAM.

# Conexión básica (Modelo clásico de Von Newmann)



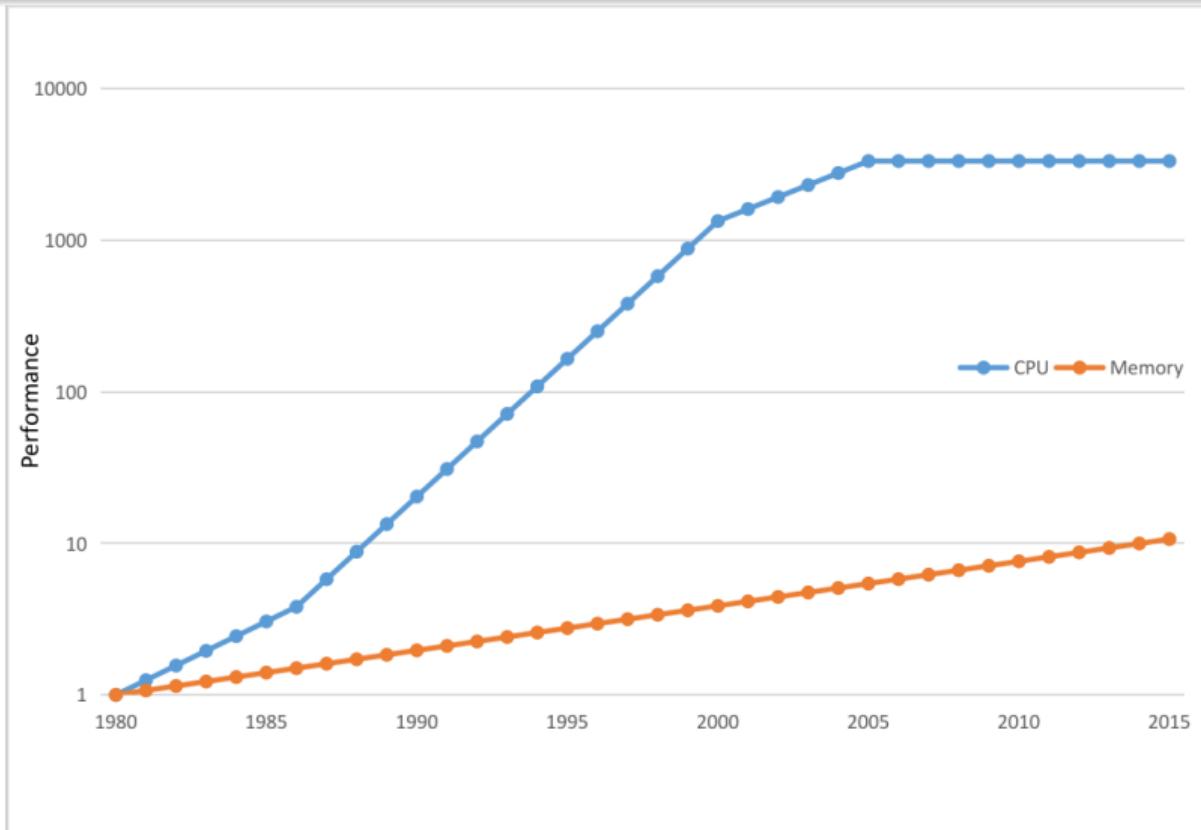
- Desde los años 80, la mejora de velocidad de las CPU comienzan a superar año a año a los tiempos de acceso a memoria DRAM.
- En este escenario, el procesador necesita insertar **Wait States** en cada ciclo de acceso a memoria, para esperar que ésta señale que está **READY** para el acceso, mediante una línea de hardware.

# Conexión básica (Modelo clásico de Von Newmann)



- Desde los años 80, la mejora de velocidad de las CPU comienzan a superar año a año a los tiempos de acceso a memoria DRAM.
- En este escenario, el procesador necesita insertar **Wait States** en cada ciclo de acceso a memoria, para esperar que ésta señale que está **READY** para el acceso, mediante una línea de hardware.
- ¿Para que incrementar la frecuencia de Clock con toda la complejidad que esto implica si al acceder a DRAM se necesitan **Wait States**?

# Performance Relativa Memoria DRAM vs. CPU



# Performance Relativa Memoria DRAM vs. CPU. Notas

- La escala logarítmica clarifica la magnitud del gap en performance.

# Performance Relativa Memoria DRAM vs. CPU. Notas

- La escala logarítmica clarifica la magnitud del gap en performance.
- La progresión de memoria es a razón de 1.07 por año. Es constante, y parte de un baseline de 64 KiB en 1980. Terminan en un incremento de aproximadamente x10.

# Performance Relativa Memoria DRAM vs. CPU. Notas

- La escala logarítmica clarifica la magnitud del gap en performance.
- La progresión de memoria es a razón de 1.07 por año. Es constante, y parte de un baseline de 64 KiB en 1980. Terminan en un incremento de aproximadamente x10.
- Para la CPU la base es de una sola CPU por sistema.

# Performance Relativa Memoria DRAM vs. CPU. Notas

- La escala logarítmica clarifica la magnitud del gap en performance.
- La progresión de memoria es a razón de 1.07 por año. Es constante, y parte de un baseline de 64 KiB en 1980. Terminan en un incremento de aproximadamente x10.
- Para la CPU la base es de una sola CPU por sistema.
- El crecimiento tiene tramos diferentes: Hasta 1986 1.25 por año, luego hasta 2000 1.52 por año, entre 2000, y 2005 1.2, y a partir de ese año el mejoramiento de la performance entra en una meseta.

# Performance Relativa Memoria DRAM vs. CPU. Notas

- La escala logarítmica clarifica la magnitud del gap en performance.
- La progresión de memoria es a razón de 1.07 por año. Es constante, y parte de un baseline de 64 KiB en 1980. Terminan en un incremento de aproximadamente x10.
- Para la CPU la base es de una sola CPU por sistema.
- El crecimiento tiene tramos diferentes: Hasta 1986 1.25 por año, luego hasta 2000 1.52 por año, entre 2000, y 2005 1.2, y a partir de ese año el mejoramiento de la performance entra en una meseta.
- Casualmente en ese año comienzan a presentarse los chips multicore.

# Performance Relativa Memoria DRAM vs. CPU. Notas

- La escala logarítmica clarifica la magnitud del gap en performance.
- La progresión de memoria es a razón de 1.07 por año. Es constante, y parte de un baseline de 64 KiB en 1980. Terminan en un incremento de aproximadamente x10.
- Para la CPU la base es de una sola CPU por sistema.
- El crecimiento tiene tramos diferentes: Hasta 1986 1.25 por año, luego hasta 2000 1.52 por año, entre 2000, y 2005 1.2, y a partir de ese año el mejoramiento de la performance entra en una meseta.
- Casualmente en ese año comienzan a presentarse los chips multicore.

## Conclusión

Necesitamos una estrategia de diseño para al menos minimizar el efecto de este gap.

# El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema. Hay dos opciones...

# El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema. Hay dos opciones...

- **RAM dinámica (DRAM)**

- **RAM estática (SRAM)**

# El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema. Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.

- **RAM estática (SRAM)**

- Alto consumo relativo.

# El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema. Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.
- Capacidad de almacenamiento comparativamente alta.

- **RAM estática (SRAM)**

- Alto consumo relativo.
- Capacidad de almacenamiento comparativamente baja.

# El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema. Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.
- Capacidad de almacenamiento comparativamente alta.
- Costo por bit bajo.

- **RAM estática (SRAM)**

- Alto consumo relativo.
- Capacidad de almacenamiento comparativamente baja.
- Costo por bit alto.

# El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema. Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.
- Capacidad de almacenamiento comparativamente alta.
- Costo por bit bajo.
- Tiempo de acceso alto (lento), debido al circuito de regeneración de carga.

- **RAM estática (SRAM)**

- Alto consumo relativo.
- Capacidad de almacenamiento comparativamente baja.
- Costo por bit alto.
- Tiempo de acceso bajo (es mas rápida).

# El problema...

El problema consiste en decidir que tipo de RAM usar en el sistema. Hay dos opciones...

- **RAM dinámica (DRAM)**

- Consumo mínimo.
- Capacidad de almacenamiento comparativamente alta.
- Costo por bit bajo.
- Tiempo de acceso alto (lento), debido al circuito de regeneración de carga.

- **RAM estática (SRAM)**

- Alto consumo relativo.
- Capacidad de almacenamiento comparativamente baja.
- Costo por bit alto.
- Tiempo de acceso bajo (es mas rápida).

## Conclusión:

Si utilizamos solo SRAM, obtenemos velocidad acorde con la del procesador, pero el costo y el consumo de la computadora son inviables. Si utilizamos solo DRAM, resolvemos costo y energía, pero la velocidad es inaceptable.

# La solución: Memoria Cache

- En Inglés: **cache**: “*a hidden store of things, or the place where they are kept.*”

# La solución: Memoria Cache

- En Inglés: **cache**: “*a hidden store of things, or the place where they are kept.*”
- En base al concepto de estructura jerárquica, la **Memoria Cache** ocupa el primer nivel de la jerarquía. Es la primer memoria que se encuentra la dirección que la CPU coloca en el Address Bus.

# La solución: Memoria Cache

- En Inglés: **cache**: “*a hidden store of things, or the place where they are kept.*”
- En base al concepto de estructura jerárquica, la **Memoria Cache** ocupa el primer nivel de la jerarquía. Es la primer memoria que se encuentra la dirección que la CPU coloca en el Address Bus.
- Físicamente se implementa conectando un bloque de memoria SRAM de muy bajo tiempo de acceso.

# La solución: Memoria Cache

- En Inglés: **cache**: “*a hidden store of things, or the place where they are kept.*”
- En base al concepto de estructura jerárquica, la **Memoria Cache** ocupa el primer nivel de la jerarquía. Es la primer memoria que se encuentra la dirección que la CPU coloca en el Address Bus.
- Físicamente se implementa conectando un bloque de memoria SRAM de muy bajo tiempo de acceso.
- Mantiene **una copia** de los datos e instrucciones que el procesador está utilizando, o cuya probabilidad de ser requeridos en lo inmediato es muy alta (Principio de localidad).

# La solución: Memoria Cache

- En Inglés: **cache**: “*a hidden store of things, or the place where they are kept.*”
- En base al concepto de estructura jerárquica, la **Memoria Cache** ocupa el primer nivel de la jerarquía. Es la primer memoria que se encuentra la dirección que la CPU coloca en el Address Bus.
- Físicamente se implementa conectando un bloque de memoria SRAM de muy bajo tiempo de acceso.
- Mantiene **una copia** de los datos e instrucciones que el procesador está utilizando, o cuya probabilidad de ser requeridos en lo inmediato es muy alta (Principio de localidad).
- Los mismos datos e instrucciones permanecen en los restantes niveles de la jerarquía de memoria. Esto sugiere la necesidad de mantenerlos coherentes cuando un dato se modifica.

# La solución: Memoria Cache

- El arte consiste en que esta copia esté disponible en la SRAM justo cuando el procesador la necesita.

## La solución: Memoria Cache

- El arte consiste en que esta copia esté disponible en la SRAM justo cuando el procesador la necesita.
- Con esta solución, el procesador puede acceder a estos items sin insertar **wait states** en el ciclo de lectura o escritura de memoria.

## La solución: Memoria Cache

- El arte consiste en que esta copia esté disponible en la SRAM justo cuando el procesador la necesita.
- Con esta solución, el procesador puede acceder a estos items sin insertar **wait states** en el ciclo de lectura o escritura de memoria.
- En un nivel inferior en la estructura jerárquica de memoria, se mapea una gran cantidad de DRAM, en donde se almacenará el total de código y datos que se requieren en memoria física, por parte de la totalidad de procesos que se ejecuten en el computador.

## La solución: Memoria Cache

- El arte consiste en que esta copia esté disponible en la SRAM justo cuando el procesador la necesita.
- Con esta solución, el procesador puede acceder a estos items sin insertar **wait states** en el ciclo de lectura o escritura de memoria.
- En un nivel inferior en la estructura jerárquica de memoria, se mapea una gran cantidad de DRAM, en donde se almacenará el total de código y datos que se requieren en memoria física, por parte de la totalidad de procesos que se ejecuten en el computador.
- Esta solución requiere hardware adicional para asegurar que este pequeño banco de memoria SRAM (de ahora en mas **cache**) contenga los datos e instrucciones mas frecuentemente utilizados por el procesador.

# Sistemas Cache. Principal aplicación de las SRAM

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Características y métricas

Tamaño ideal del banco de memoria cache:

# Características y métricas

Tamaño ideal del banco de memoria cache:

- 1 Suficientemente grande para que el procesador resuelva la mayor cantidad posible de búsquedas de código y datos en esta memoria asegurando una alta performance.

# Características y métricas

Tamaño ideal del banco de memoria cache:

- ① Suficientemente grande para que el procesador resuelva la mayor cantidad posible de búsquedas de código y datos en esta memoria asegurando una alta performance.
- ② Suficientemente pequeña para no afectar el consumo ni el costo del sistema.

# Características y métricas

Tamaño ideal del banco de memoria cache:

- ① Suficientemente grande para que el procesador resuelva la mayor cantidad posible de búsquedas de código y datos en esta memoria asegurando una alta performance.
- ② Suficientemente pequeña para no afectar el consumo ni el costo del sistema.

**Hit** Acceso a un ítem (dato o código) que *se encuentra* en la memoria cache

# Características y métricas

Tamaño ideal del banco de memoria cache:

- ① Suficientemente grande para que el procesador resuelva la mayor cantidad posible de búsquedas de código y datos en esta memoria asegurando una alta performance.
- ② Suficientemente pequeña para no afectar el consumo ni el costo del sistema.

**Hit** Acceso a un ítem (dato o código) que *se encuentra* en la memoria cache

**Miss** Acceso a un ítem (dato o código) que *no se encuentra* en la memoria cache

# Características y métricas

Tamaño ideal del banco de memoria cache:

- ① Suficientemente grande para que el procesador resuelva la mayor cantidad posible de búsquedas de código y datos en esta memoria asegurando una alta performance.
- ② Suficientemente pequeña para no afectar el consumo ni el costo del sistema.

**Hit** Acceso a un ítem (dato o código) que *se encuentra* en la memoria cache

**Miss** Acceso a un ítem (dato o código) que *no se encuentra* en la memoria cache

$$\text{hit rate} \text{ hitrate} = \frac{\text{Cantidad de Accesos con hit}}{\text{Cantidad de Accesos Totales}}$$

# Características y métricas

Tamaño ideal del banco de memoria cache:

- ① Suficientemente grande para que el procesador resuelva la mayor cantidad posible de búsquedas de código y datos en esta memoria asegurando una alta performance.
- ② Suficientemente pequeña para no afectar el consumo ni el costo del sistema.

**Hit** Acceso a un ítem (dato o código) que *se encuentra* en la memoria cache

**Miss** Acceso a un ítem (dato o código) que *no se encuentra* en la memoria cache

$$\text{hit rate} \text{ hitrate} = \frac{\text{Cantidad de Accesos con hit}}{\text{Cantidad de Accesos Totales}}$$

*Se espera un hit rate lo mas alto posible.*

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Tiempo de un Miss

# Tiempo de un Miss

- Cuando el ítem requerido por la CPU no se encuentra en el cache, el tiempo de recuperación del Miss no es fácil de determinar, ya que depende de las características del Sistema.

## Tiempo de un Miss

- Cuando el ítem requerido por la CPU no se encuentra en el cache, el tiempo de recuperación del Miss no es fácil de determinar, ya que depende de las características del Sistema.
- La memoria cache se administra por bloques o líneas, basado en el principio de vecindad. Al requerirse un item traerá al cache la mayor cantidad posible de vecinos (además de aprovechar al máximo la capacidad de transferencia del bus).

## Tiempo de un Miss

- Cuando el ítem requerido por la CPU no se encuentra en el cache, el tiempo de recuperación del Miss no es fácil de determinar, ya que depende de las características del Sistema.
- La memoria cache se administra por bloques o líneas, basado en el principio de vecindad. Al requerirse un item traerá al cache la mayor cantidad posible de vecinos (además de aprovechar al máximo la capacidad de transferencia del bus).
- Si está en los niveles inferiores de memoria, el tiempo de recupero del ítem dependerá del ***latency*** y del ***bandwidth***, de la memoria con que esté implementado ese nivel de la jerarquía.

## Tiempo de un Miss

- Cuando el ítem requerido por la CPU no se encuentra en el cache, el tiempo de recuperación del Miss no es fácil de determinar, ya que depende de las características del Sistema.
- La memoria cache se administra por bloques o líneas, basado en el principio de vecindad. Al requerirse un item traerá al cache la mayor cantidad posible de vecinos (además de aprovechar al máximo la capacidad de transferencia del bus).
- Si está en los niveles inferiores de memoria, el tiempo de recupero del ítem dependerá del ***latency*** y del ***bandwidth***, de la memoria con que esté implementado ese nivel de la jerarquía.
- ***Latency*** es el tiempo para traer desde la CPU la primer palabra de la línea, y ***bandwidth*** determina el tiempo adicional para traer al cache el resto de las palabras que caben dentro de la línea.

# Tiempo de un Miss

- Si el procesador ejecuta instrucciones en orden, se detiene hasta conseguir el ítem requerido. Si ejecuta fuera de orden, puede seguir con otras instrucciones dejando en espera a la que generó el Miss.

# Tiempo de un Miss

- Si el procesador ejecuta instrucciones en orden, se detiene hasta conseguir el ítem requerido. Si ejecuta fuera de orden, puede seguir con otras instrucciones dejando en espera a la que generó el Miss.
- La memoria principal normalmente está administrada por bloques fijos a los que llamamos páginas.

## Tiempo de un Miss

- Si el procesador ejecuta instrucciones en orden, se detiene hasta conseguir el ítem requerido. Si ejecuta fuera de orden, puede seguir con otras instrucciones dejando en espera a la que generó el Miss.
- La memoria principal normalmente está administrada por bloques fijos a los que llamamos páginas.
- El elemento buscado podría entonces no estar ubicado en una página que se encuentre alojada en la memoria principal. En tal caso estará en la memoria Virtual, es decir, el disco (hard disk o en el mejor de los casos de estado sólido).

## Tiempo de un Miss

- Si el procesador ejecuta instrucciones en orden, se detiene hasta conseguir el ítem requerido. Si ejecuta fuera de orden, puede seguir con otras instrucciones dejando en espera a la que generó el Miss.
- La memoria principal normalmente está administrada por bloques fijos a los que llamamos páginas.
- El elemento buscado podría entonces no estar ubicado en una página que se encuentre alojada en la memoria principal. En tal caso estará en la memoria Virtual, es decir, el disco (hard disk o en el mejor de los casos de estado sólido).
- En caso en que el dato no corresponda a una página presente en la memoria física, se genera una detención del procesamiento, y el Sistema Operativo junto con la CPU se concentrarán en cargar la página que contiene el ítem desde la memoria virtual a la física.

# Performance de un cache

- La siguiente expresión determina de manera general el Tiempo de Ejecución de una CPU:

$$CPU_{ExecTime} = (CPU_{ClockCycles} + Memory_{StallCycles}) * Periodo_{Clock}$$

# Performance de un cache

- La siguiente expresión determina de manera general el Tiempo de Ejecución de una CPU:

$$CPU_{ExecTime} = (CPU_{ClockCycles} + Memory_{StallCycles}) * Periodo_{Clock}$$

- Se asume que  $CPU_{ClockCycles}$  incluye los ciclos de búsqueda en el cache si se trata de un Hit, y que el procesador se atasca ante un Miss.

# Performance de un cache

- La siguiente expresión determina de manera general el Tiempo de Ejecución de una CPU:

$$CPU_{ExecTime} = (CPU_{ClockCycles} + Memory_{StallCycles}) * Periodo_{Clock}$$

- Se asume que  $CPU_{ClockCycles}$  incluye los ciclos de búsqueda en el cache si se trata de un Hit, y que el procesador se atasca ante un Miss.
- El tiempo de Atasco de memoria depende del número de Miss y del tiempo que cuesta cada Miss, de ahora en mas *Miss penalty*:

$$\begin{aligned}Memory_{StallCycles} &= \#Misses * Miss_{Penalty} \\&= IC * \frac{\text{Misses}}{\text{Instruction}} * Miss_{Penalty}; IC = \text{Instruction Count} \\&= IC * \frac{\text{MemoryAccesses}}{\text{Instruction}} * Miss_{rate} * Miss_{Penalty}\end{aligned}$$

## Performance de un cache

- La siguiente expresión determina de manera general el Tiempo de Ejecución de una CPU:  
$$CPU_{ExecTime} = (CPU_{ClockCycles} + Memory_{StallCycles}) * Periodo_{Clock}$$
- Se asume que  $CPU_{ClockCycles}$  incluye los ciclos de búsqueda en el cache si se trata de un Hit, y que el procesador se atasca ante un Miss.
- El tiempo de Atasco de memoria depende del número de Miss y del tiempo que cuesta cada Miss, de ahora en mas *Miss penalty*:

$$\begin{aligned}Memory_{StallCycles} &= \#Misses * Miss_{Penalty} \\&= IC * \frac{\text{Misses}}{\text{Instruction}} * Miss_{Penalty}; IC = \text{Instruction Count} \\&= IC * \frac{\text{MemoryAccesses}}{\text{Instruction}} * Miss_{rate} * Miss_{Penalty}\end{aligned}$$

- La última expresión es mas conveniente ya que se compone en su totalidad de factores factibles de ser medidos con precisión.

# Performance de un cache

- $\text{Miss}_{\text{Penalty}}$  es un valor fuertemente dependiente de cada sistema, ya que depende de un gran número de valores de diseño ya enunciados. Además puede sufrir fuertes variaciones de un acceso al siguiente por las mismas razones. De modo que lo mejor que podemos hacer es obtener un valor promedio en base a experimentar con una cantidad de casos suficientemente grande como para ser representativa.

# Performance de un cache

- $\text{Miss}_{\text{Penalty}}$  es un valor fuertemente dependiente de cada sistema, ya que depende de un gran número de valores de diseño ya enunciados. Además puede sufrir fuertes variaciones de un acceso al siguiente por las mismas razones. De modo que lo mejor que podemos hacer es obtener un valor promedio en base a experimentar con una cantidad de casos suficientemente grande como para ser representativa.
- $\text{Miss}_{\text{rate}}$  puede ser determinado con simuladores de acceso a memoria por plataforma, o en el caso de muchos procesadores, se dispone de hardware específico que provee la cantidad de accesos a memoria y la de Misses de modo que puede ser fácilmente establecido.

# Performance de un cache

- $\text{Miss}_{\text{Penalty}}$  es un valor fuertemente dependiente de cada sistema, ya que depende de un gran número de valores de diseño ya enunciados. Además puede sufrir fuertes variaciones de un acceso al siguiente por las mismas razones. De modo que lo mejor que podemos hacer es obtener un valor promedio en base a experimentar con una cantidad de casos suficientemente grande como para ser representativa.
- $\text{Miss}_{\text{rate}}$  puede ser determinado con simuladores de acceso a memoria por plataforma, o en el caso de muchos procesadores, se dispone de hardware específico que provee la cantidad de accesos a memoria y la de Misses de modo que puede ser fácilmente establecido.
- Además  $\text{Miss}_{\text{rate}}$  y  $\text{Miss}_{\text{Penalty}}$  varían de manera notable para lecturas y escrituras. Aquí también es conveniente trabajar con valores promedio.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad**
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

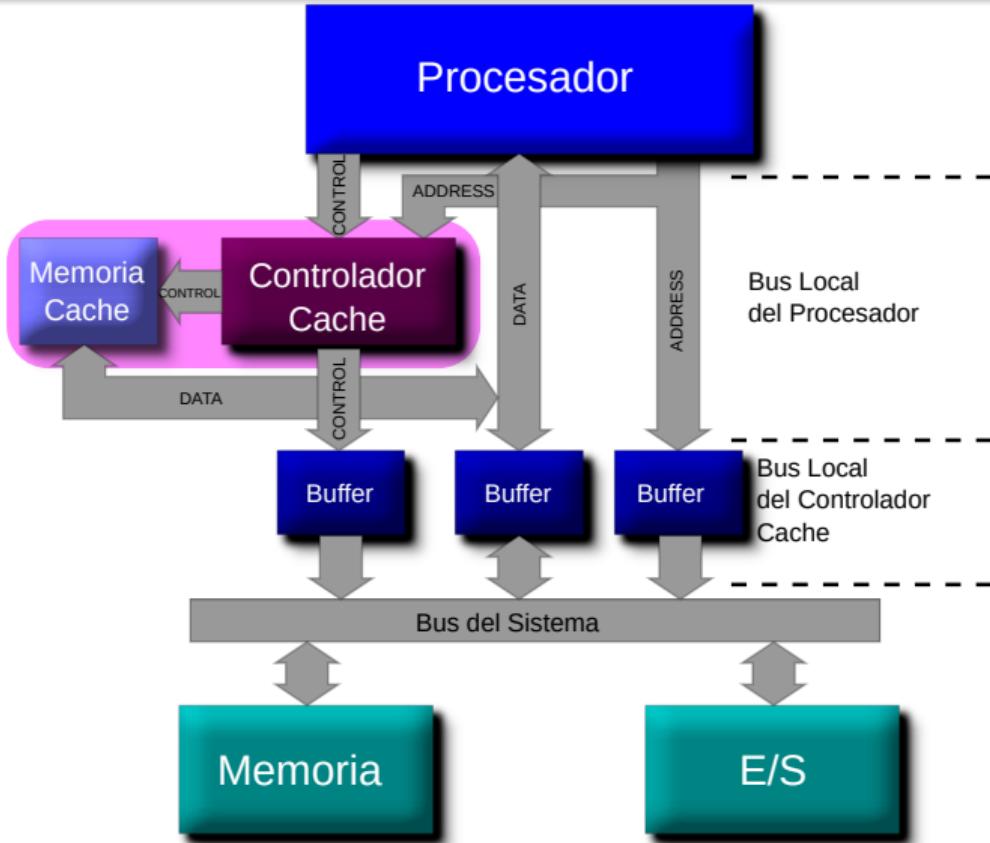
## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

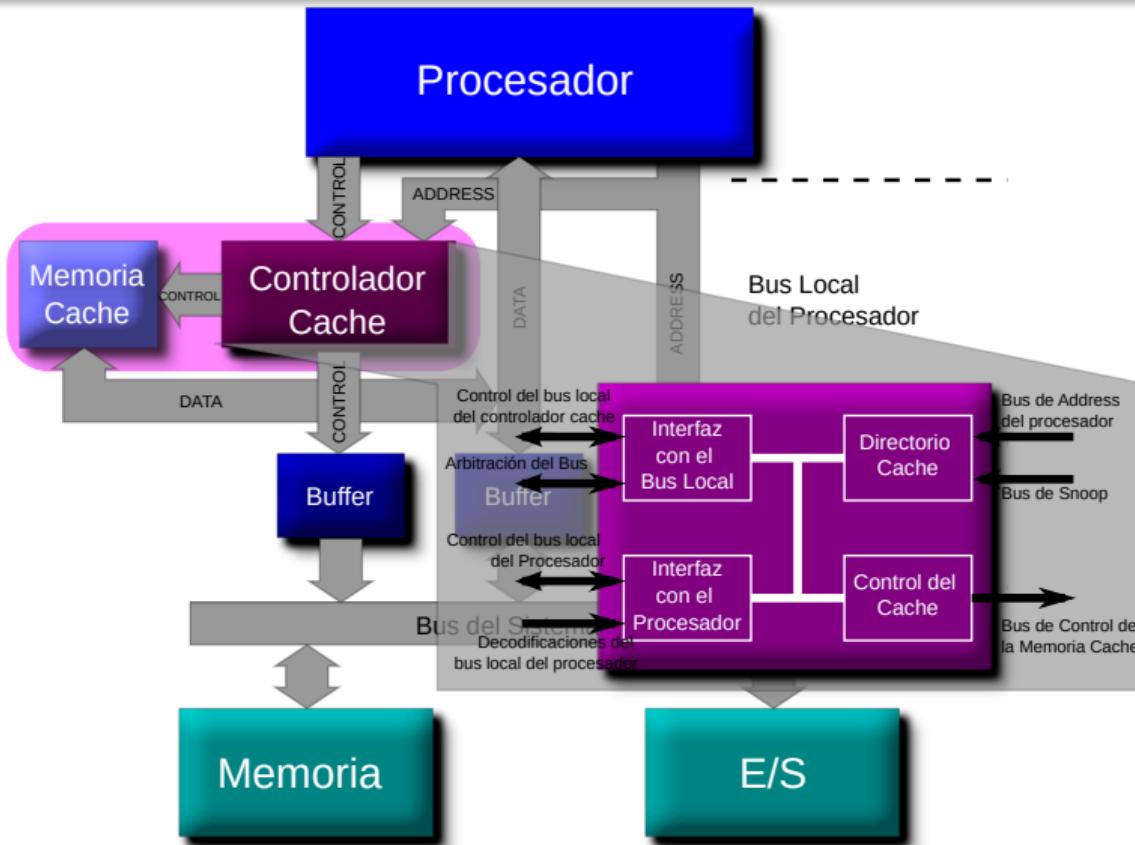
## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

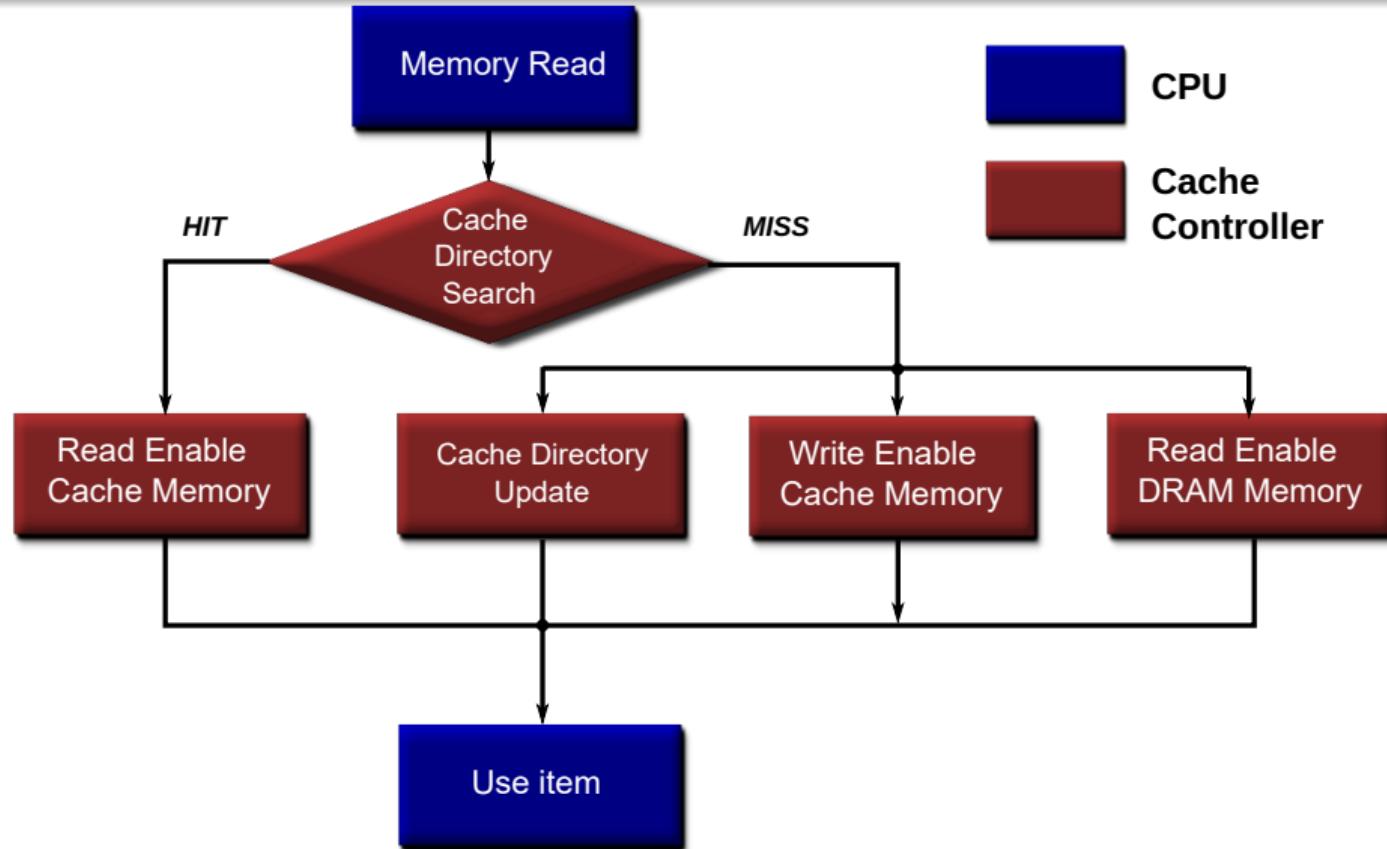
# Subsistema Cache de Hardware



# El Controlador Cache



# Operación de acceso a memoria para lectura



# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

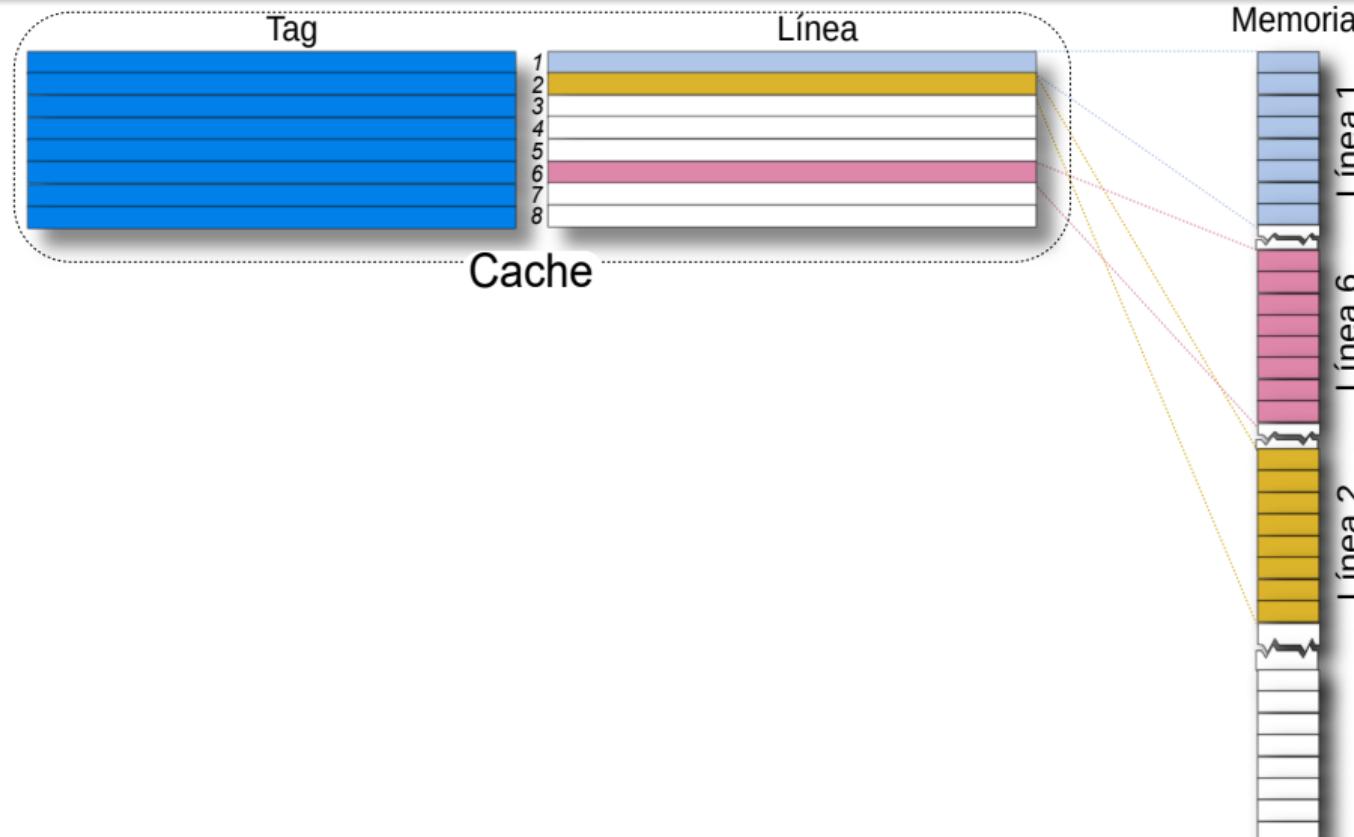
- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache**

## 4 Cache en Sistemas Multiprocesador

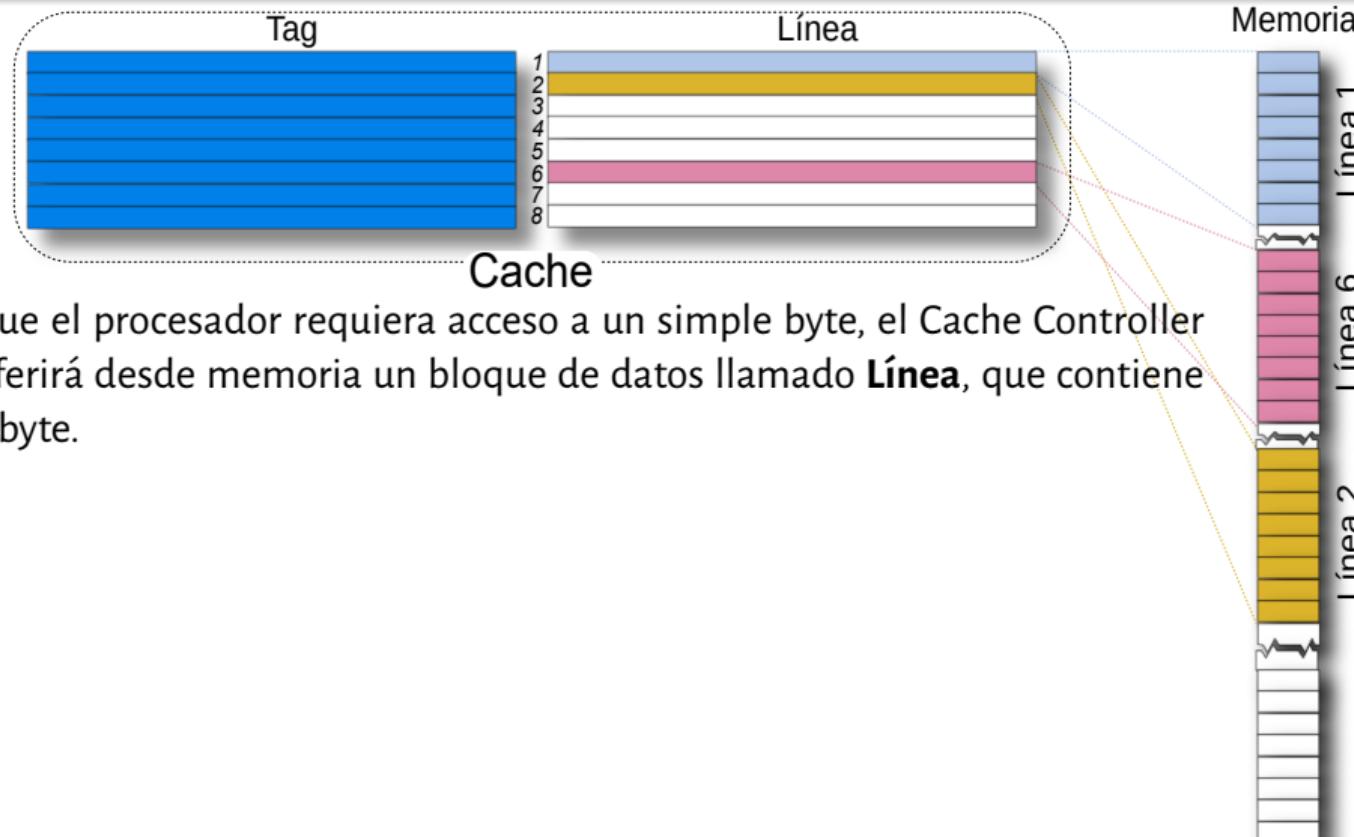
- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real
- 5 Memorias Dinámicas**
  - Introducción
  - Organización interna
- 6 Standards**
  - Estado del arte
  - JEDEC SDRAM
- 7 Controladores de Memoria**
  - Introducción General
  - Arquitectura
- 8 Configuración**
  - Configuración del DRAM Device

- Entrada Salida de Datos
- 9 Integración con el sistema Cache**
  - Evitar cuellos de botella es la clave
- 10 Casos Prácticos**
  - Beagle Bone Black
  - Memorias DDR en la BBB
  - Controlador de DDRn SDRAM en la BBB
- 11 SRAM Cuestiones de Implementación**
  - Vistazo introductorio
  - Decodificación
  - Implementación
- 12 DRAM Detalles de implementación**
  - Acceso a las celdas
  - JEDEC DDR SDRAM
  - Protocolo de acceso

# Organización del cache. Líneas

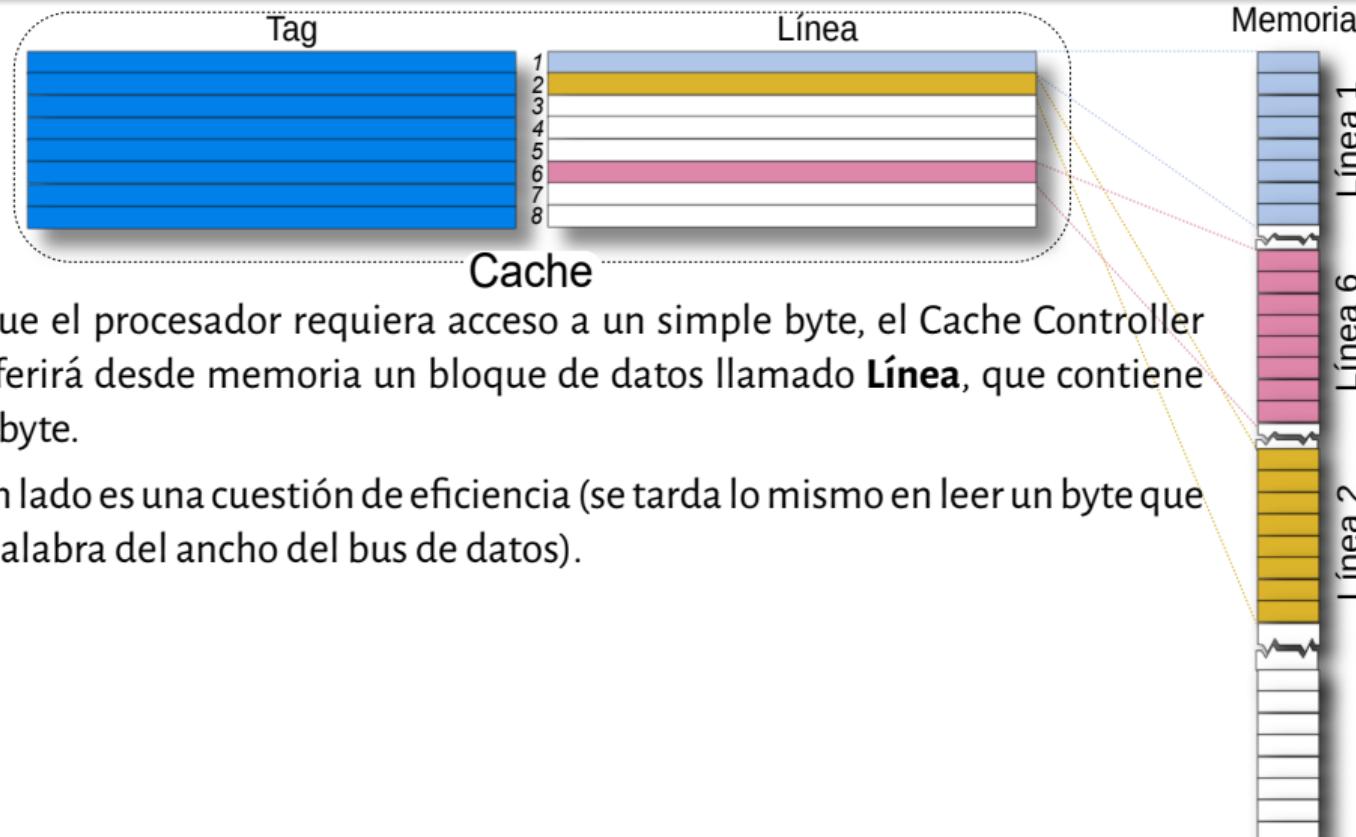


# Organización del cache. Líneas

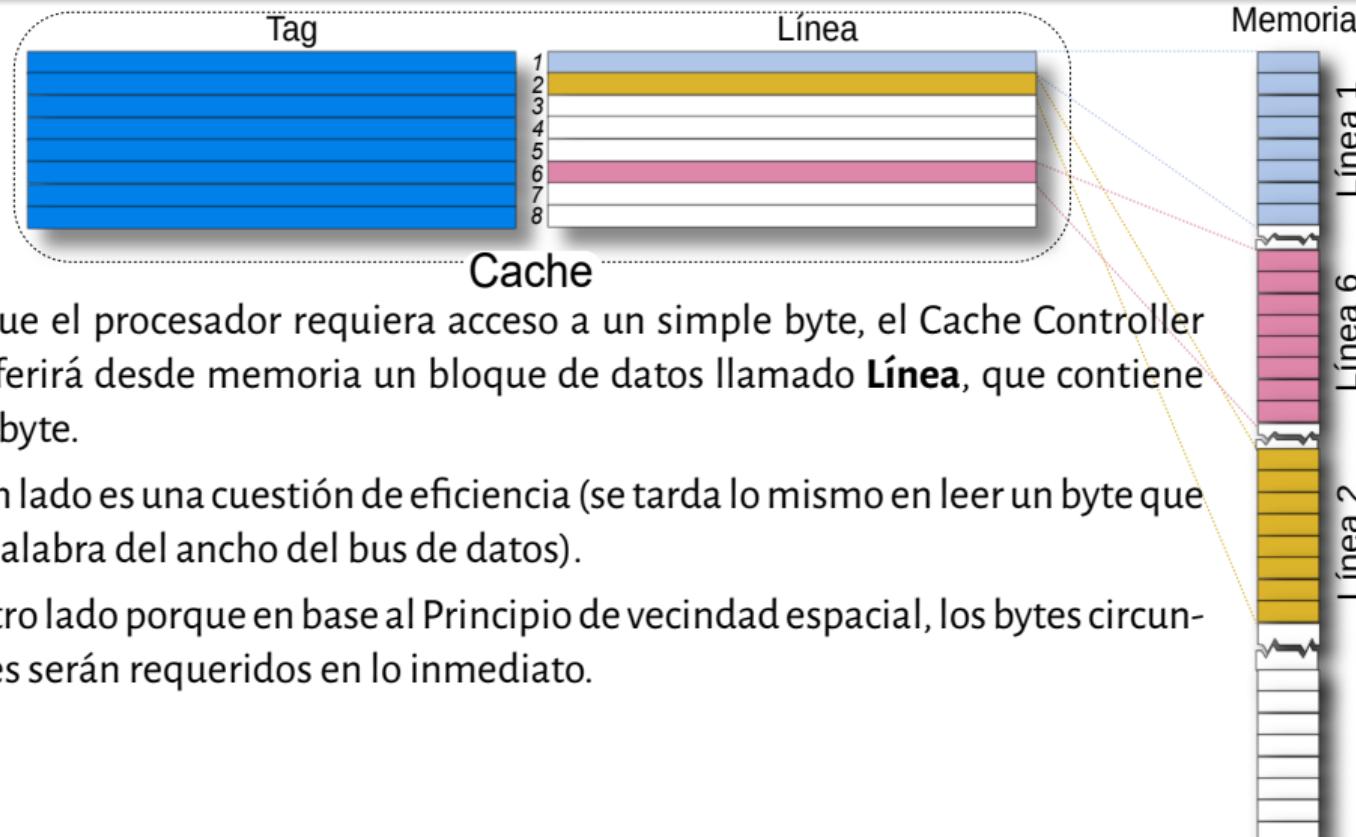


- Aunque el procesador requiera acceso a un simple byte, el Cache Controller transferirá desde memoria un bloque de datos llamado **Línea**, que contiene a ese byte.

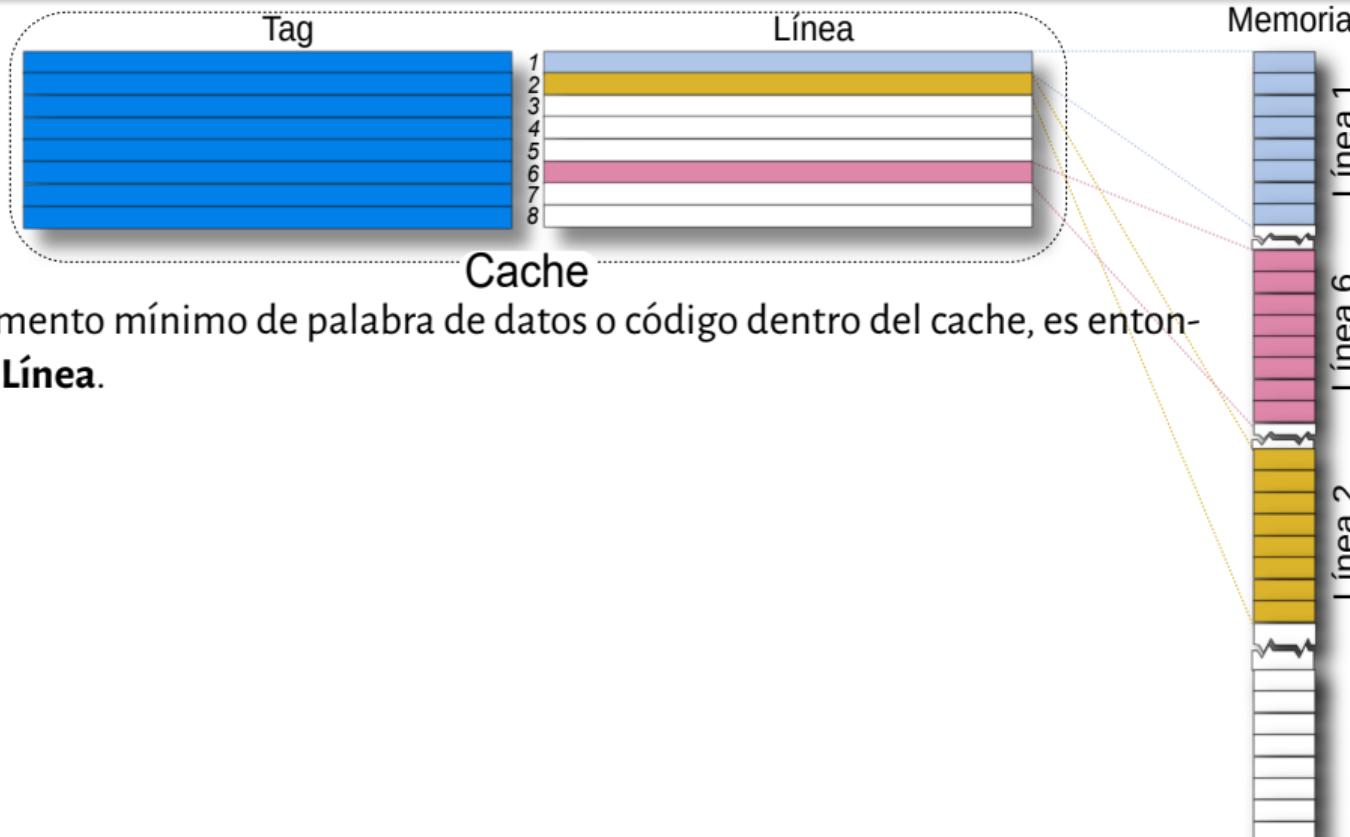
# Organización del cache. Líneas



# Organización del cache. Líneas

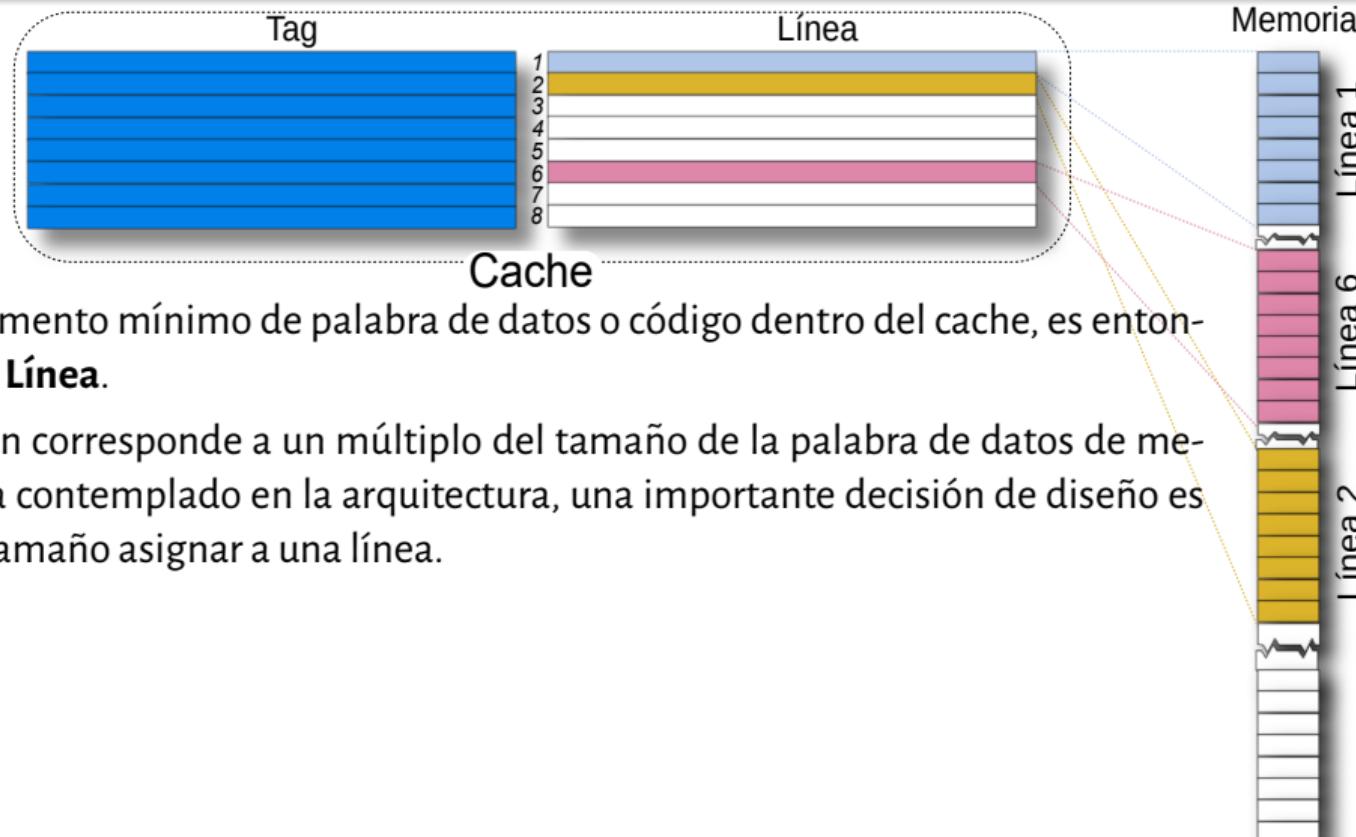


# Organización del cache. Líneas

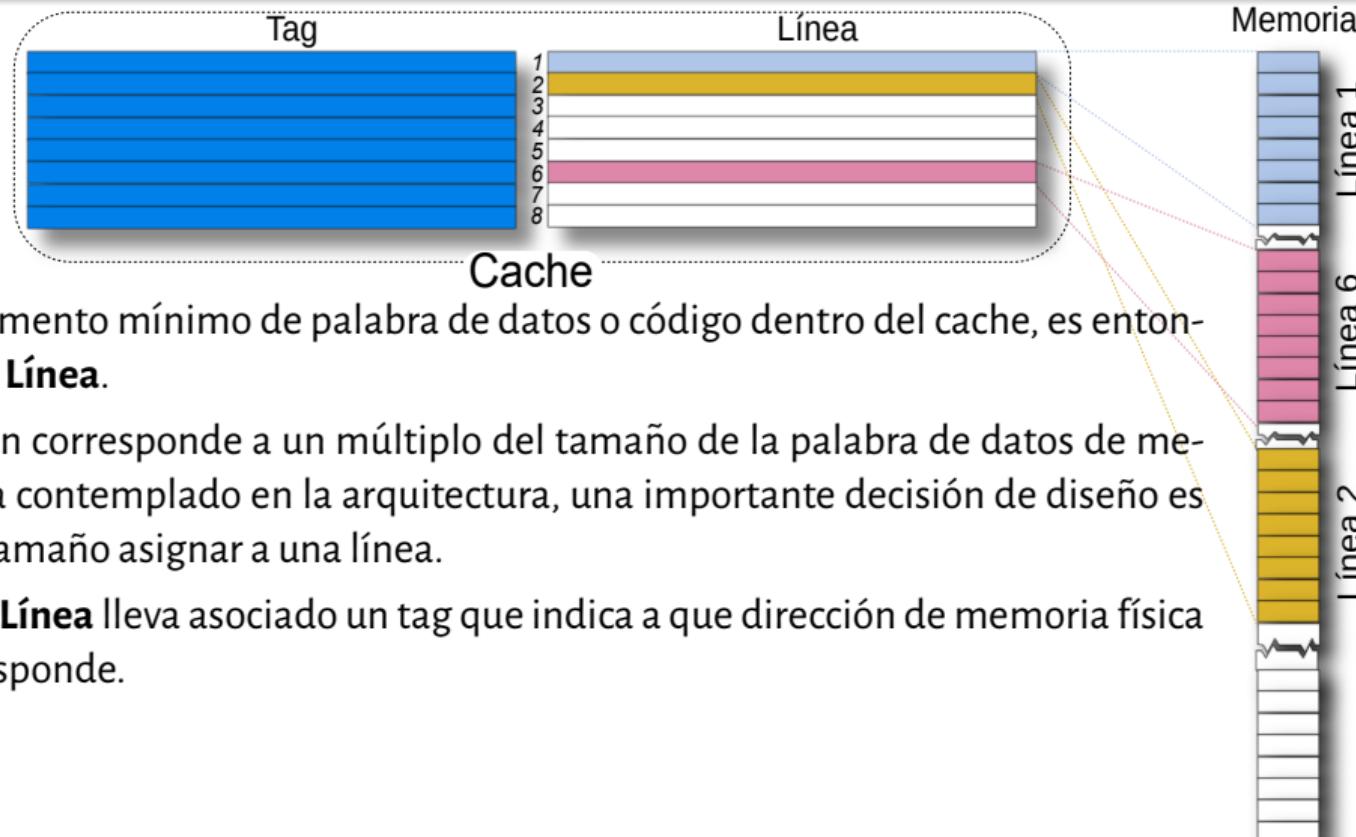


- El elemento mínimo de palabra de datos o código dentro del cache, es entonces la **Línea**.

# Organización del cache. Líneas



# Organización del cache. Líneas



# Organización del cache.

La organización de un Cache debe contemplar una serie de factores en función de las operaciones que solicita la CPU. En particular, se necesita:

# Organización del cache.

La organización de un Cache debe contemplar una serie de factores en función de las operaciones que solicita la CPU. En particular, se necesita:

- Definir como ubicar una línea de memoria principal en el cache (Line Placement).

# Organización del cache.

La organización de un Cache debe contemplar una serie de factores en función de las operaciones que solicita la CPU. En particular, se necesita:

- Definir como ubicar una línea de memoria principal en el cache (Line Placement).
- Una vez ubicada la línea se necesita un mecanismo de identificación (Line Identification).

# Organización del cache.

La organización de un Cache debe contemplar una serie de factores en función de las operaciones que solicita la CPU. En particular, se necesita:

- Definir como ubicar una línea de memoria principal en el cache (Line Placement).
- Una vez ubicada la línea se necesita un mecanismo de identificación (Line Identification).
- Un mecanismo de reemplazo de líneas existentes en respuesta a un Miss (Line Replacement).

# Organización del cache.

La organización de un Cache debe contemplar una serie de factores en función de las operaciones que solicita la CPU. En particular, se necesita:

- Definir como ubicar una línea de memoria principal en el cache (Line Placement).
- Una vez ubicada la línea se necesita un mecanismo de identificación (Line Identification).
- Un mecanismo de reemplazo de líneas existentes en respuesta a un Miss (Line Replacement).
- Una estrategia para manejar las escrituras (Write policies).

# Line Placement: ¿Donde va una línea en el cache?

## Line Placement: ¿Donde va una línea en el cache?

- Un posible criterio es una vez leída la línea desde la memoria principal colocarla en cualquiera de las líneas libres del cache. Se conoce como **Fully Associative**. El menos restrictivo posible. Una línea de memoria principal puede ubicarse en cualquiera de las líneas del cache.

## Line Placement: ¿Donde va una línea en el cache?

- Un posible criterio es una vez leída la línea desde la memoria principal colocarla en cualquiera de las líneas libres del cache. Se conoce como **Fully Associative**. El menos restrictivo posible. Una línea de memoria principal puede ubicarse en cualquiera de las líneas del cache.
- En el otro extremo, **Direct Mapped**, es el más restrictivo: Una línea de memoria principal puede ubicarse en una única línea del cache, cuyo número queda determinado por la siguiente expresión:  
$$(\text{Line Frame Address}) \bmod (\text{Cache } \# \text{Lines})$$

## Line Placement: ¿Donde va una línea en el cache?

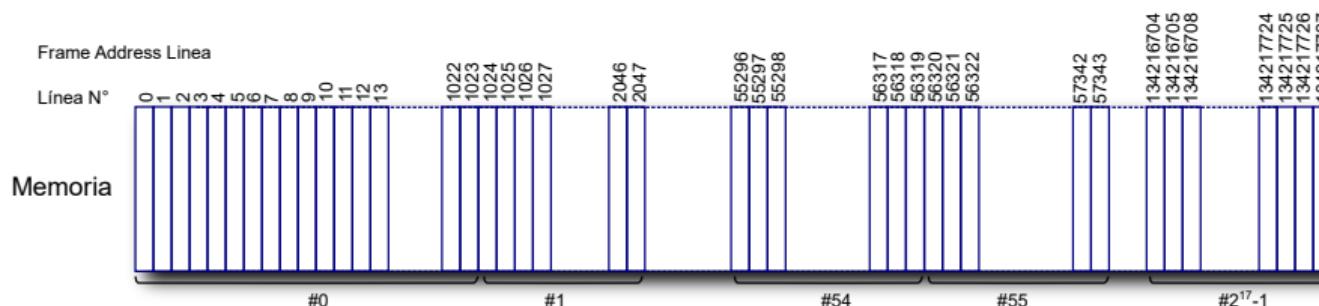
- Un posible criterio es una vez leída la línea desde la memoria principal colocarla en cualquiera de las líneas libres del cache. Se conoce como **Fully Associative**. El menos restrictivo posible. Una línea de memoria principal puede ubicarse en cualquiera de las líneas del cache.
- En el otro extremo, **Direct Mapped**, es el más restrictivo: Una línea de memoria principal puede ubicarse en una única línea del cache, cuyo número queda determinado por la siguiente expresión:  
 $(\text{Line Frame Address}) \bmod (\text{Cache } \# \text{Lines})$
- Un esquema intermedio de organización es **set associative**, en donde hay un grupo (set) de líneas en el cache en las que puede guardarse una nueva línea leída desde la memoria principal, cuyo número queda establecido por la siguiente expresión:  
 $(\text{Line Frame Address}) \bmod (\text{Cache } \# \text{Sets})$   
En general si tenemos  $n$  sets posibles para almacenar la línea el sistema se refiere como  **$n$ -way set associative**.

# Line Placement: ¿Donde va una línea en el cache?

Ejemplo: Memoria del Sistema: 4 GiB, Memoria Cache 32 KiB, Line size 32 bytes.

En términos del Controlador Cache, todo se maneja con líneas, por lo tanto, la Memoria del Sistema se compone de 131072 (128 K) bloques o grupos cada uno de igual cantidad de líneas que el cache (en este ejemplo 1024 (1K) líneas).

El procesador requiere leer la dirección 1788439.



## Line Placement: ¿Dónde va una línea en el cache?

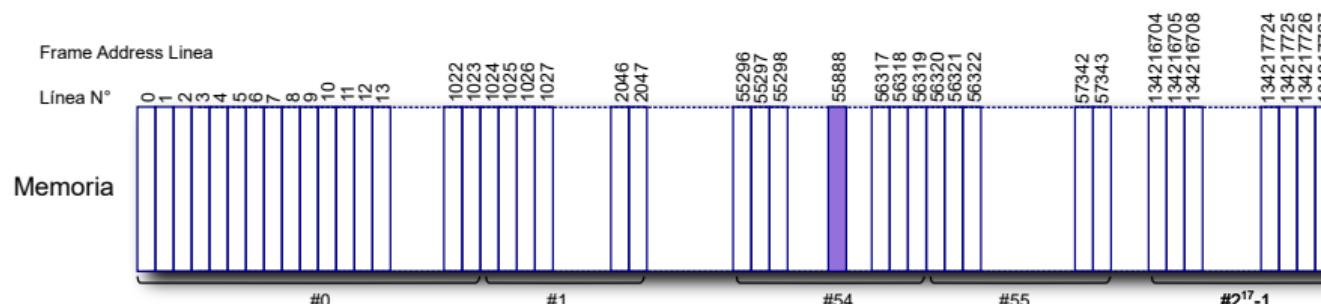
Ejemplo: Memoria del Sistema: 4 GiB, Memoria Cache 32 KiB, Line size 32 bytes.

En términos del Controlador Cache, todo se maneja con líneas, por lo tanto, la Memoria del Sistema se compone de 131072 (128 K) bloques o grupos cada uno de igual cantidad de líneas que el cache (en este ejemplo 1024 (1K) líneas).

El procesador requiere leer la dirección 1788439.

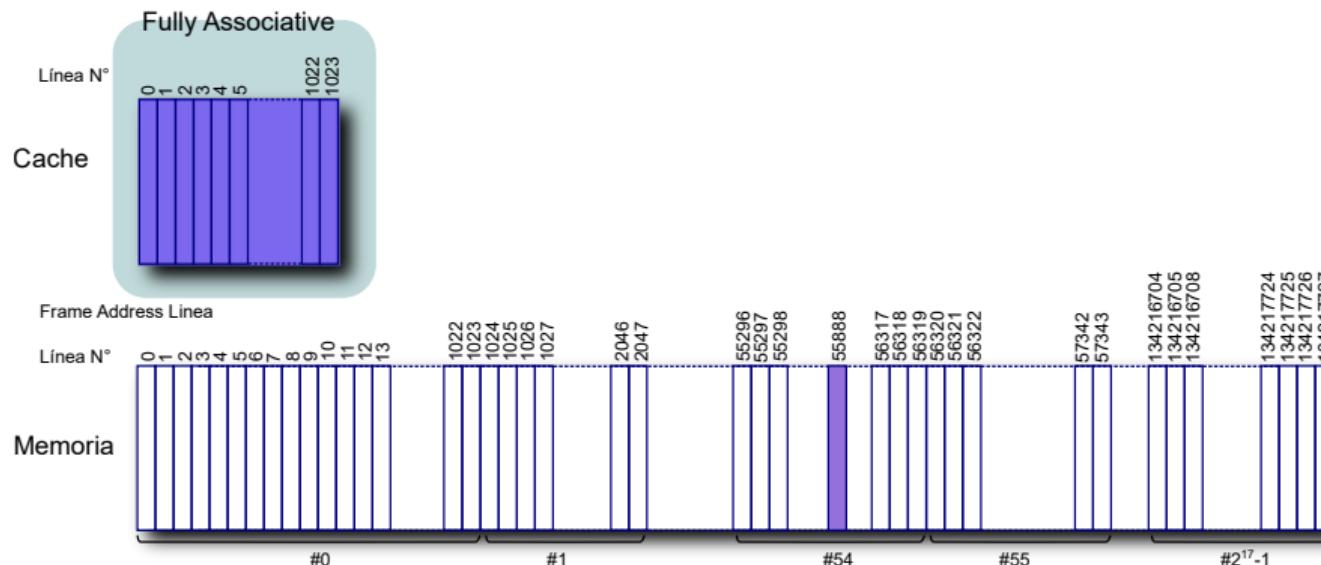
El Frame address de la línea que contiene este elemento es  $\left\lfloor \frac{1788439}{\text{Line Size}} \right\rfloor = \left\lfloor \frac{1788439}{32} \right\rfloor = 55888$

¿Dónde va una línea en el cache para cada uno de los criterios de Placement definidos?



# Line Placement: ¿Dónde va una línea en el cache?

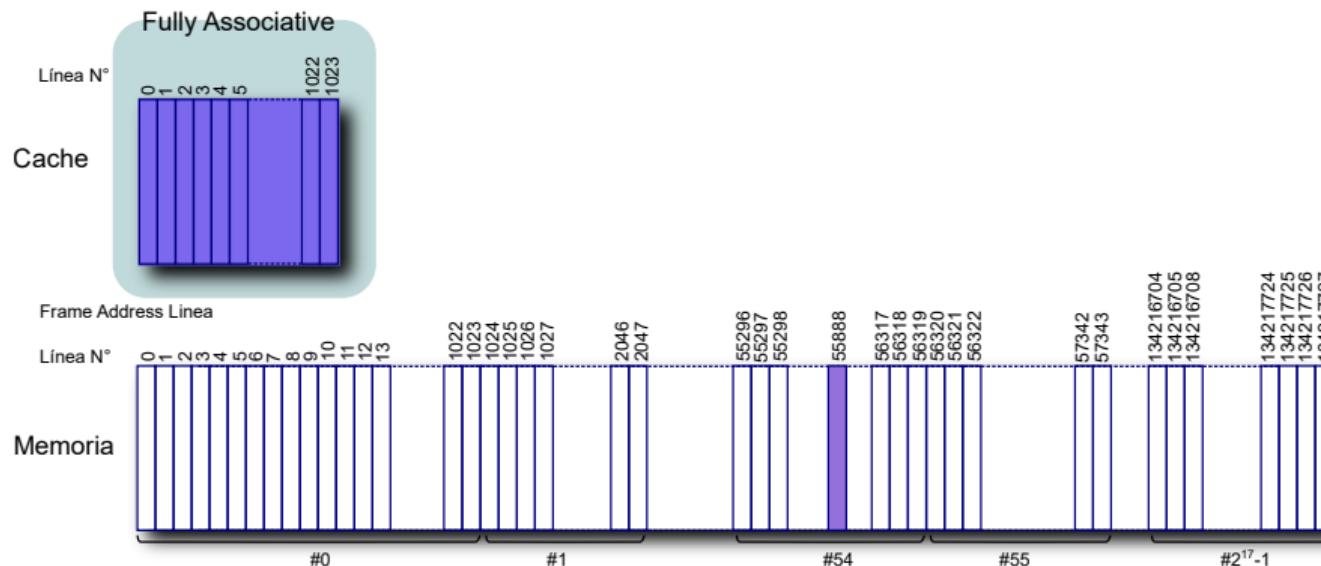
Puede pensarse como un ***n-way Set Associative*** en el que ***n*** coincide con el número de líneas del cache.



# Line Placement: ¿Dónde va una línea en el cache?

Puede pensarse como un ***n-way Set Associative*** en el que ***n*** coincide con el número de líneas del cache.

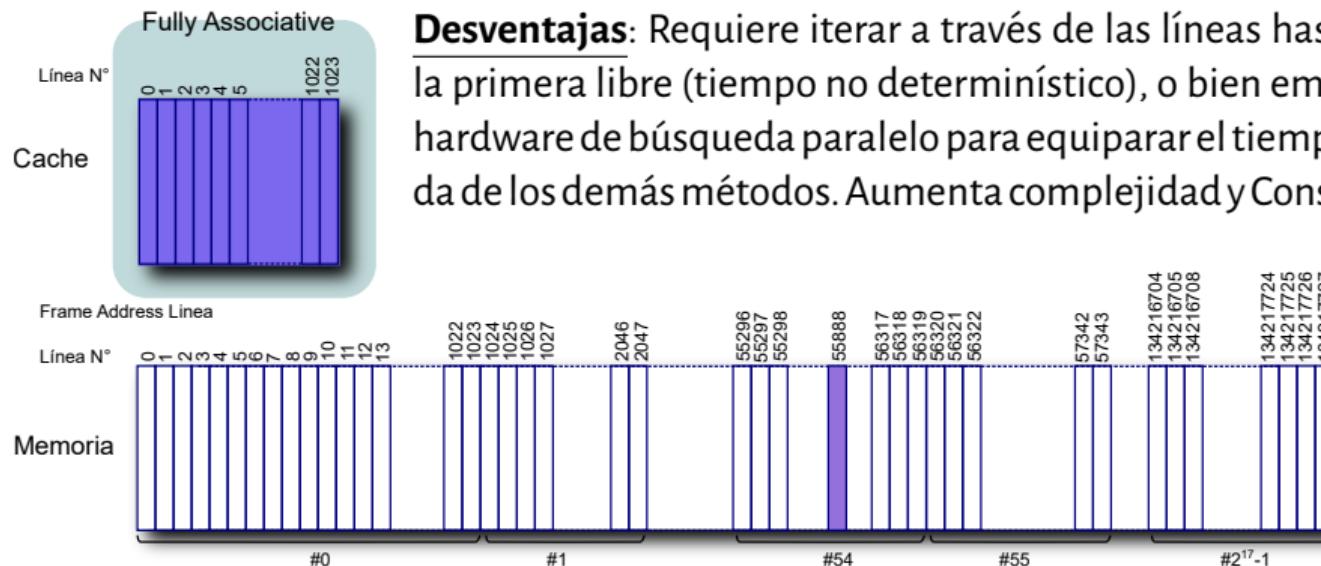
**Ventajas:** Ofrece máxima flexibilidad para el Ubicar la línea. Permite obtener el máximo Hit Rate. Permite utilizar una amplia variedad de algoritmos de reemplazo de línea.



# Line Placement: ¿Donde va una línea en el cache?

Puede pensarse como un ***n-way Set Associative*** en el que ***n*** coincide con el número de líneas del cache.

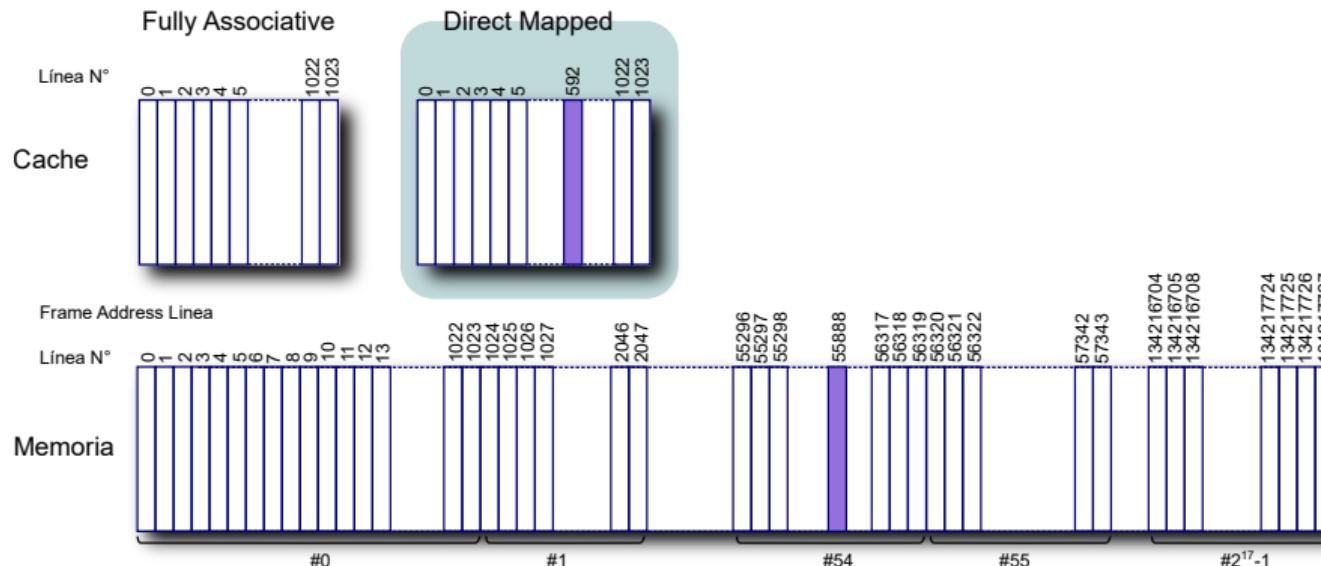
**Ventajas:** Ofrece máxima flexibilidad para el Ubicar la línea. Permite obtener el máximo Hit Rate. Permite utilizar una amplia variedad de algoritmos de reemplazo de línea.



**Desventajas:** Requiere iterar a través de las líneas hasta encontrar la primera libre (tiempo no determinístico), o bien emplear mucho hardware de búsqueda paralelo para equiparar el tiempo de búsqueda de los demás métodos. Aumenta complejidad y Consumo de energía.

# Line Placement: ¿Dónde va una línea en el cache?

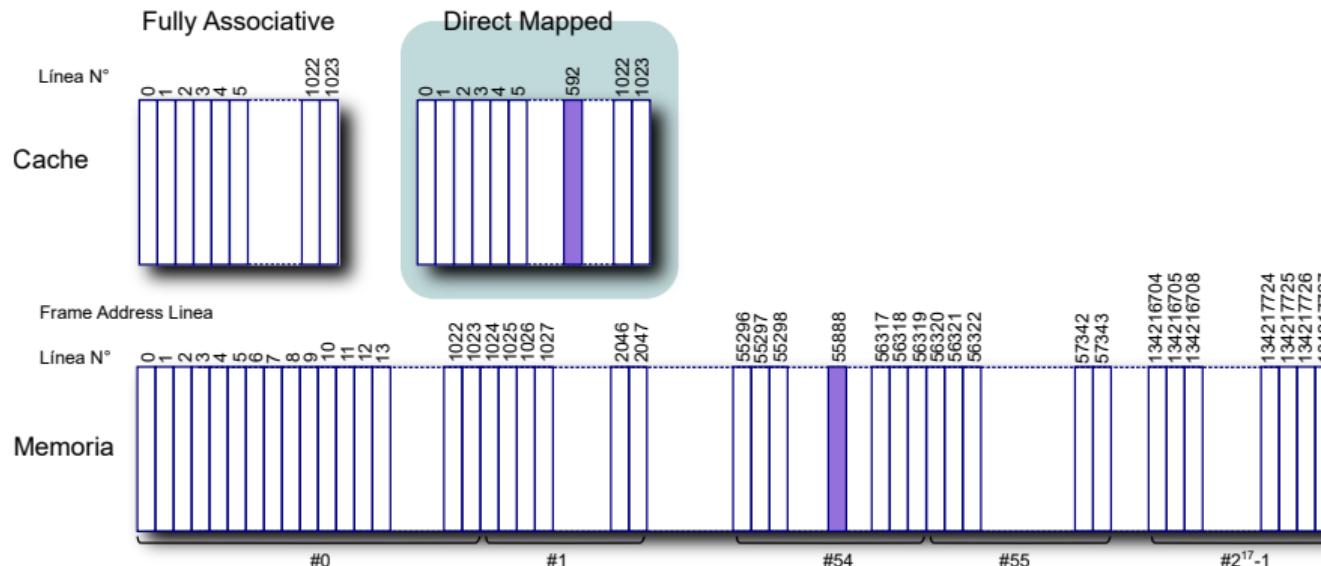
Puede pensarse como un ***n-way Set Associative*** en el que ***n*** es 1.



# Line Placement: ¿Dónde va una línea en el cache?

Puede pensarse como un ***n-way Set Associative*** en el que ***n*** es 1.

**Ventajas:** Simplifica al máximo el Placement y la Búsqueda. Minimiza el soporte de hardware y en consecuencia el consumo de energía.

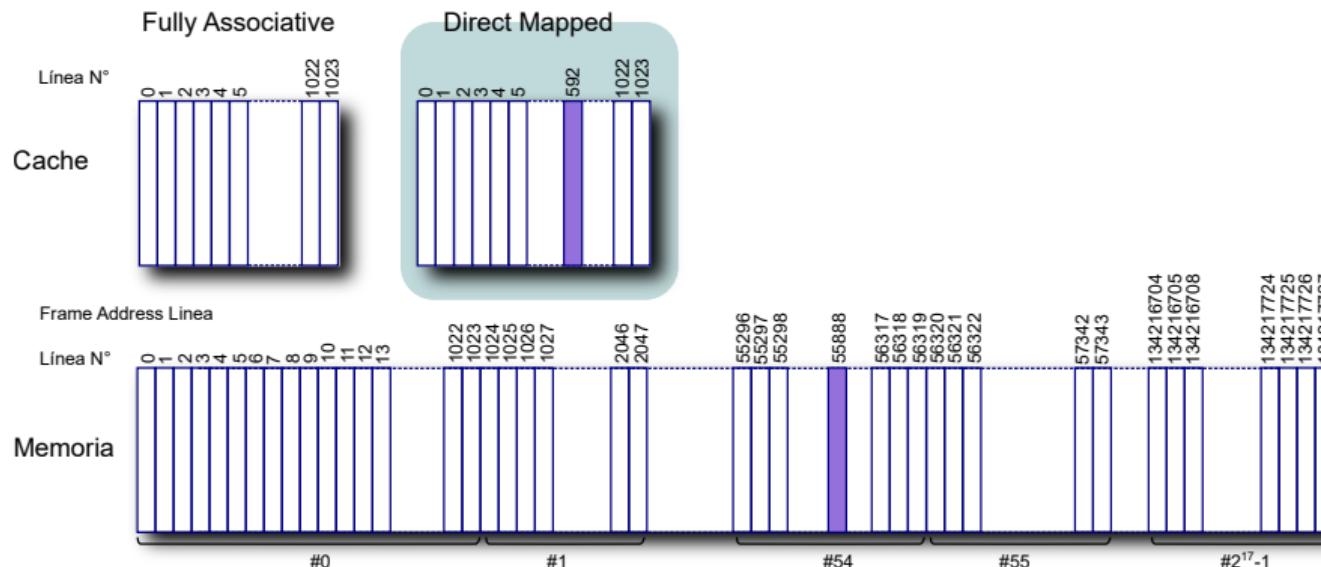


# Line Placement: ¿Dónde va una línea en el cache?

Puede pensarse como un ***n-way Set Associative*** en el que ***n*** es 1.

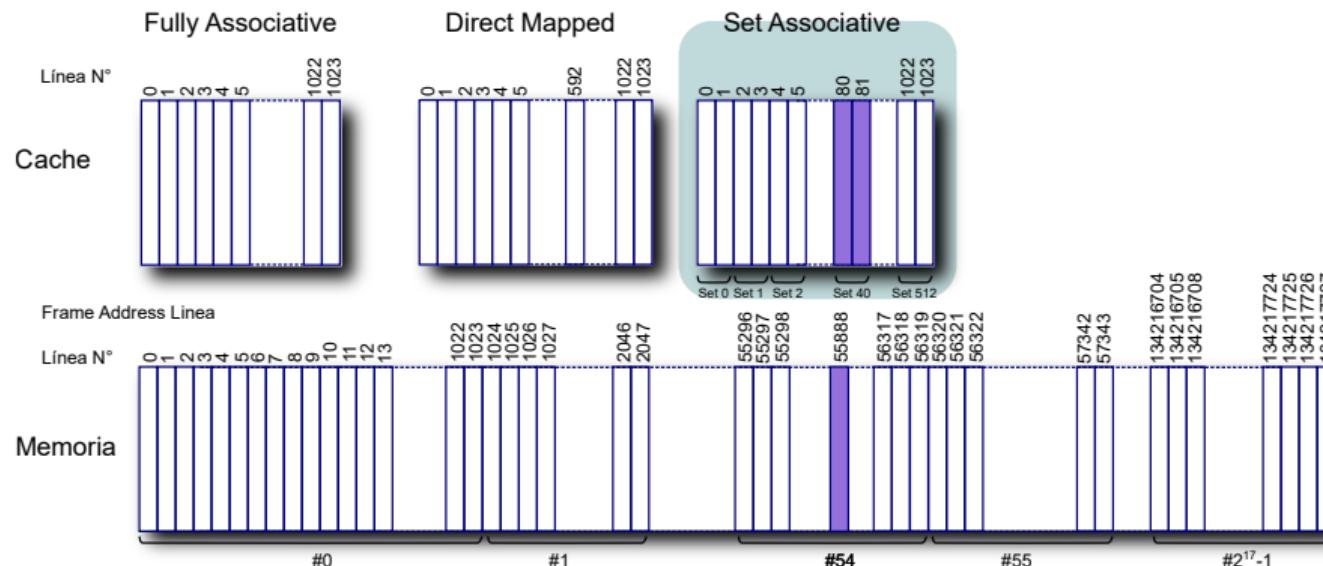
**Ventajas:** Simplifica al máximo el Placement y la Búsqueda. Minimiza el soporte de hardware y en consecuencia el consumo de energía.

**Desventajas:** Ofrece el peor Hit Rate. Dos líneas frecuentemente utilizadas que mapeen sobre la misma línea del cache, generan desalojos recíprocos, situación conocida como ***Conflict Miss***.



# Line Placement: ¿Donde va una línea en el cache?

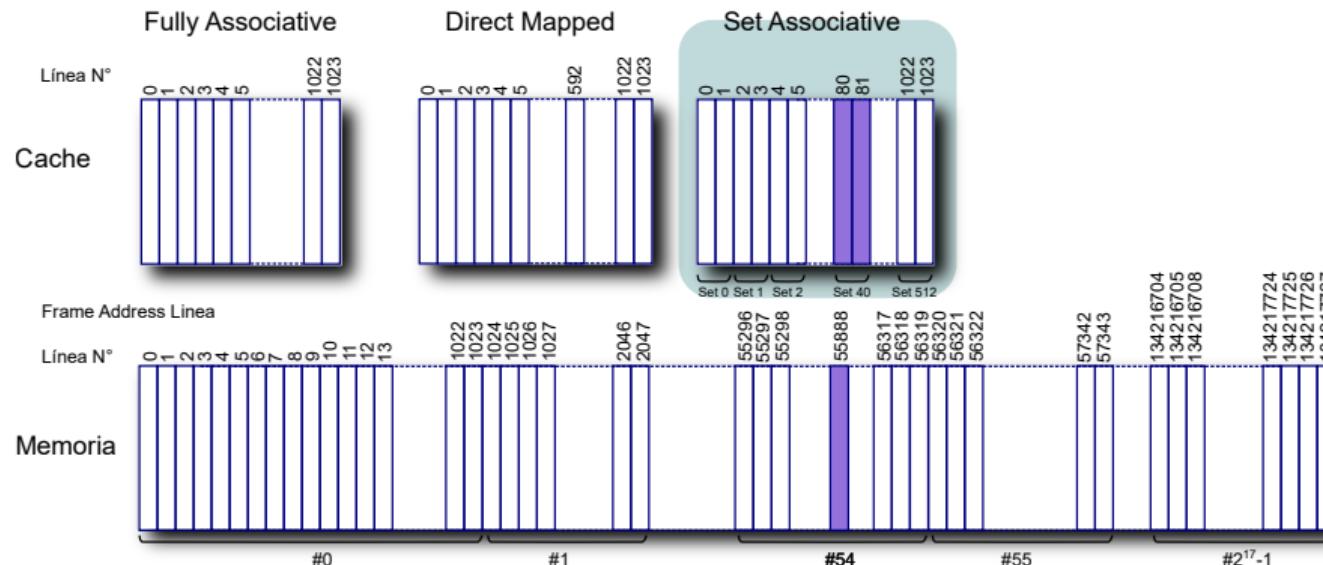
Coloca la línea en cualquiera disponible dentro del set.



# Line Placement: ¿Donde va una línea en el cache?

Coloca la línea en cualquiera disponible dentro del set.

**Ventajas:** Es la mejor relación de compromiso entre los otros criterios para determinar el Placement. La Búsqueda permite flexibilidad en el uso de diferentes algoritmos.

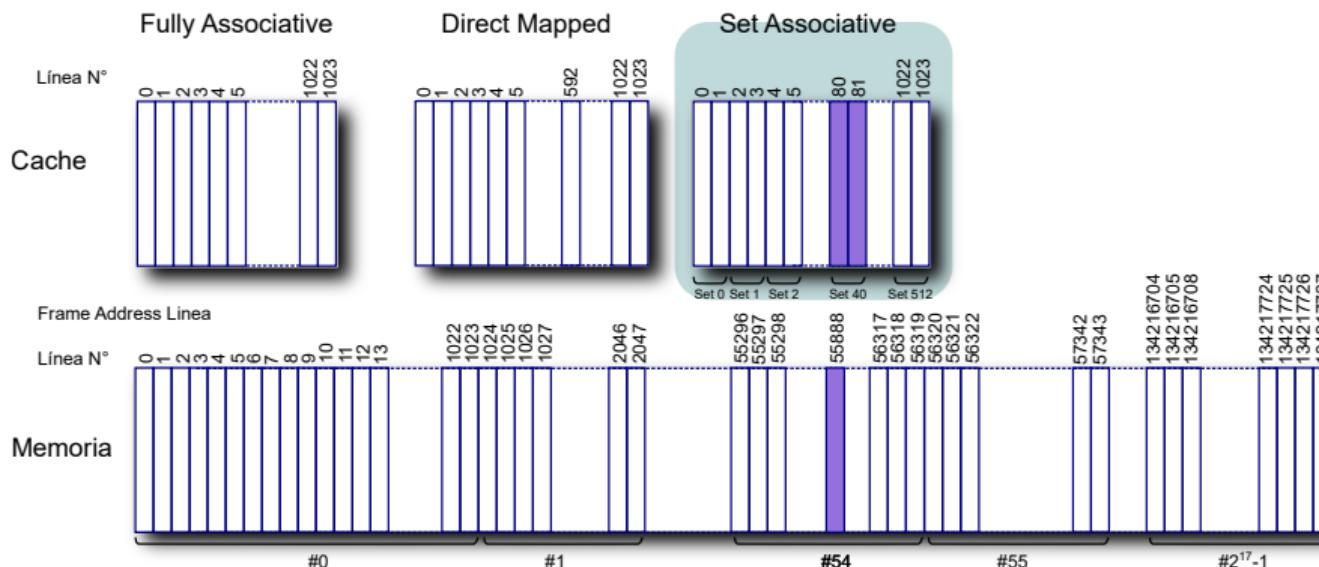


# Line Placement: ¿Donde va una línea en el cache?

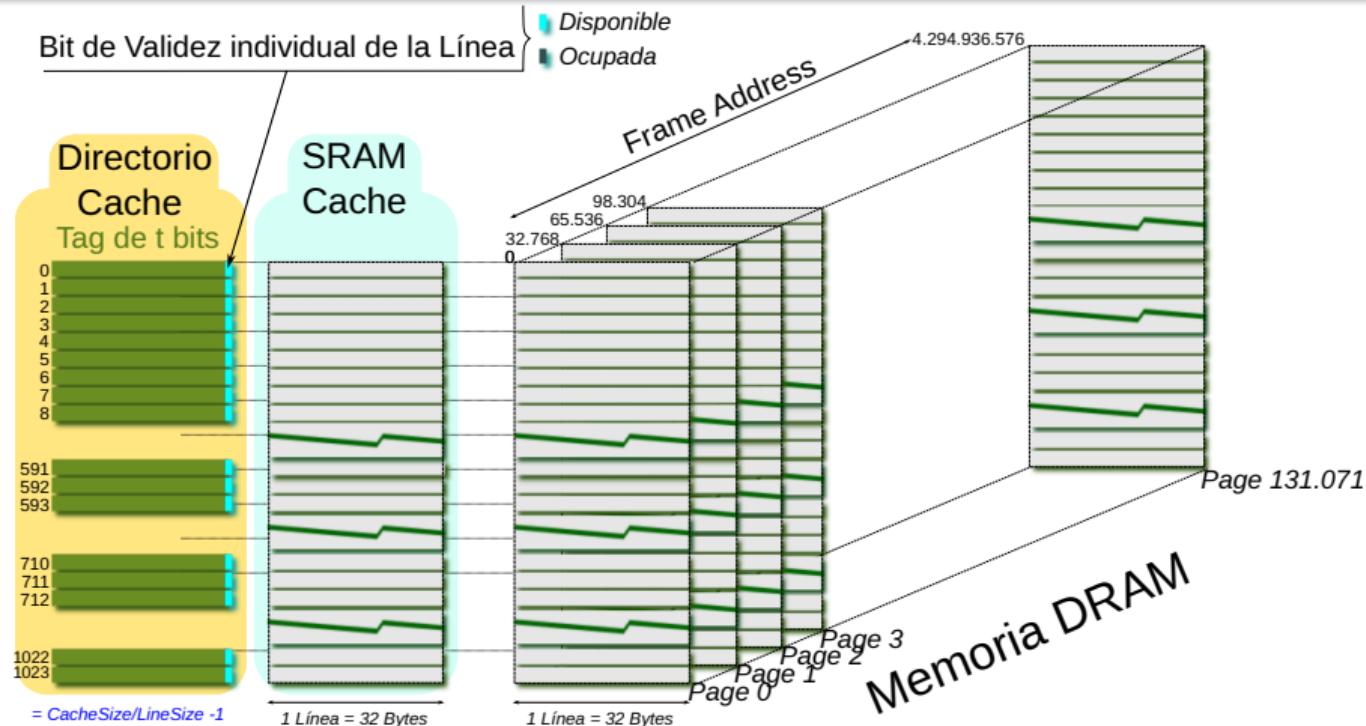
Coloca la línea en cualquiera disponible dentro del set.

**Ventajas:** Es la mejor relación de compromiso entre los otros criterios para determinar el Placement. La Búsqueda permite flexibilidad en el uso de diferentes algoritmos.

**Desventajas:** Su aprovechamiento de la capacidad del cache es sub óptimo. Puede llegar a Escenarios de **Conflict Miss**.

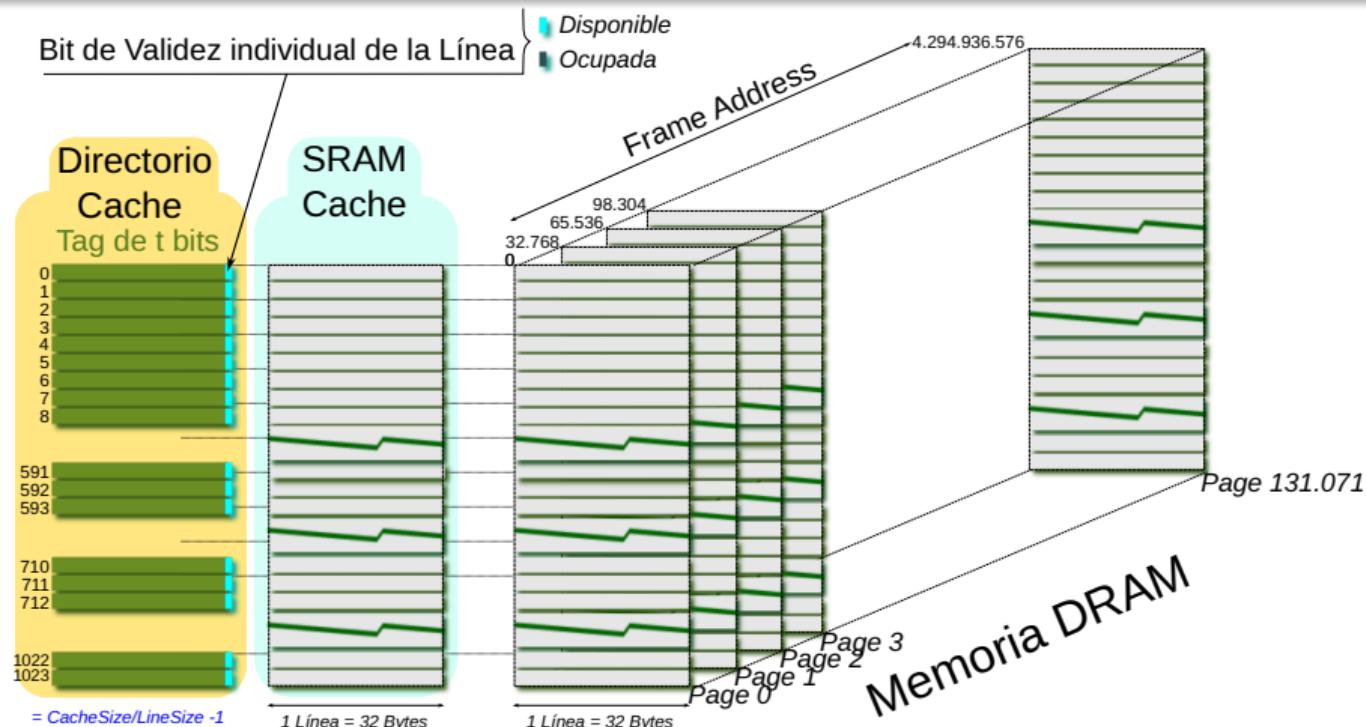


# Line Placement: Cache de Mapeo Directo



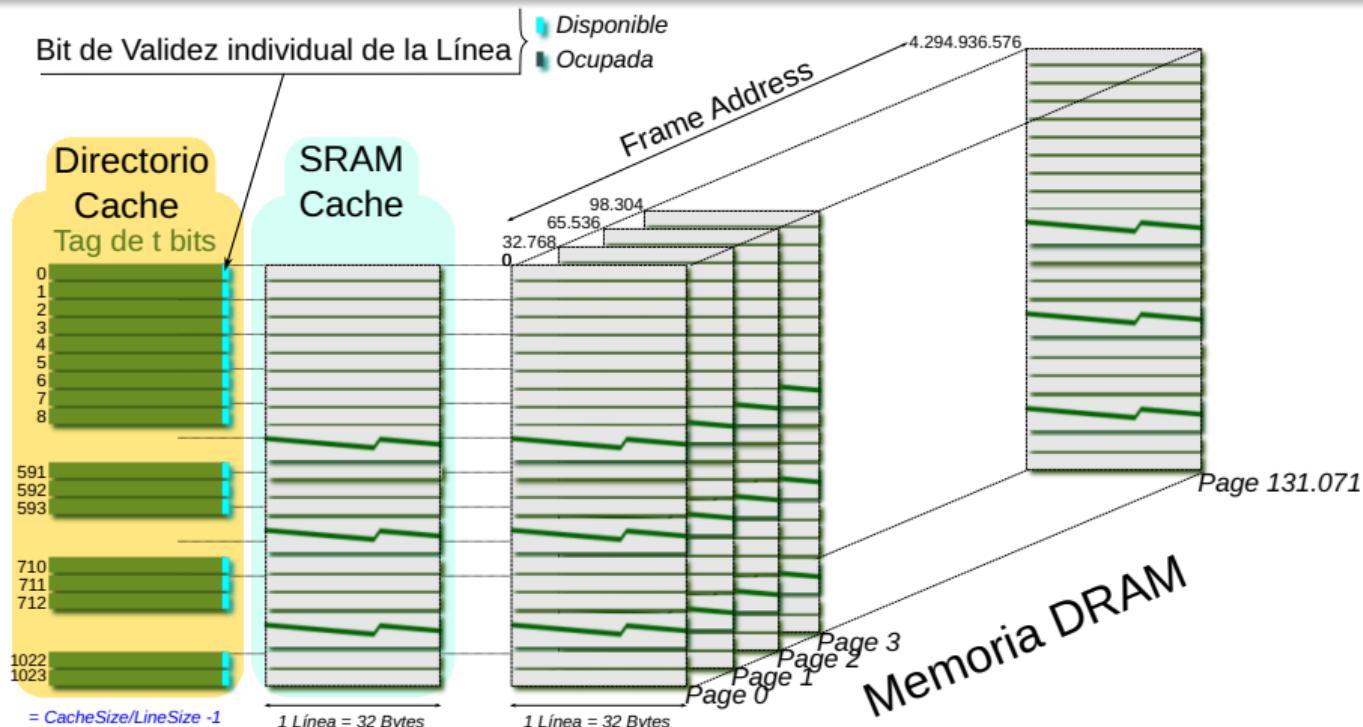
Cache de 32 KiB organizado en 1024 líneas de 32 B. La memoria Principal es vista como un conjunto de Páginas del mismo tamaño del cache, organizadas en líneas.

# Line Placement: Cache de Mapeo Directo



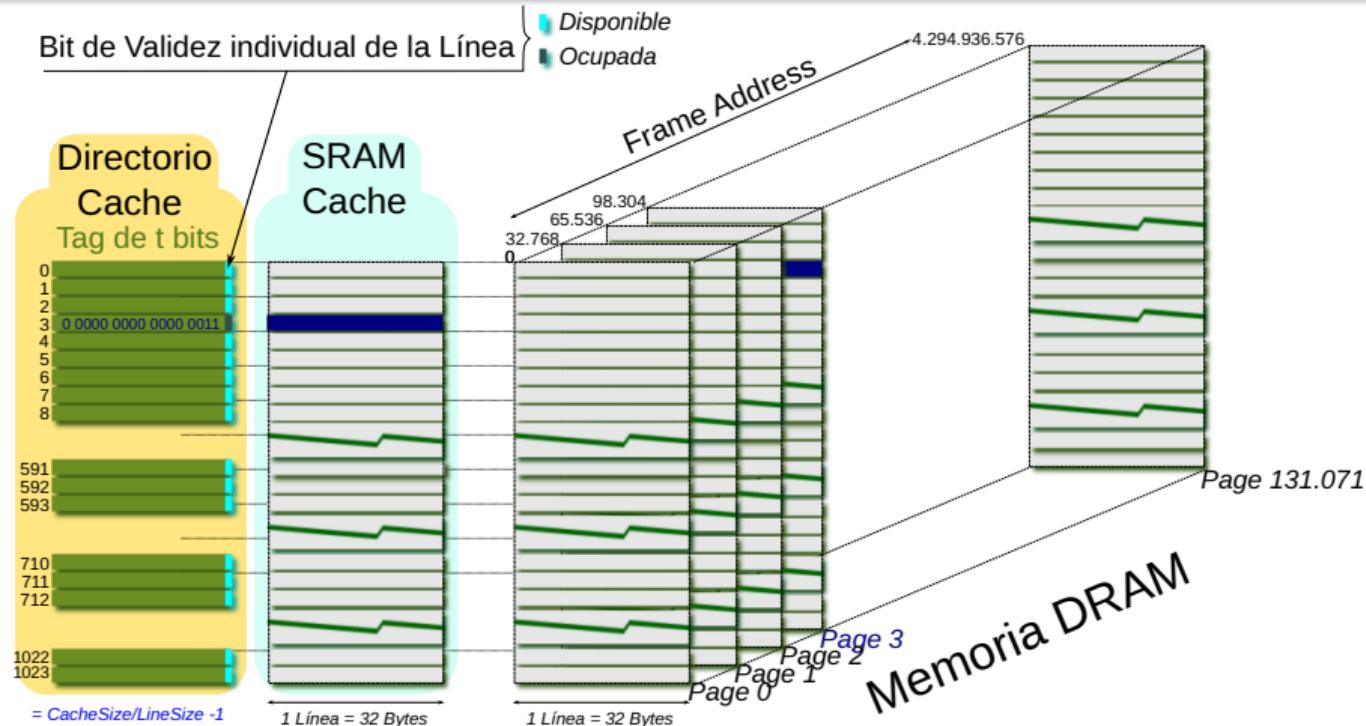
Al igual que en el ejemplo anterior tenemos una Memoria DRAM de 4 GiB, que por lo tanto el Controlador Cache “ve” 131072 páginas ( $2^{17}$ ) de 32 KiB, organizadas en líneas de 32 B

# Line Placement: Cache de Mapeo Directo



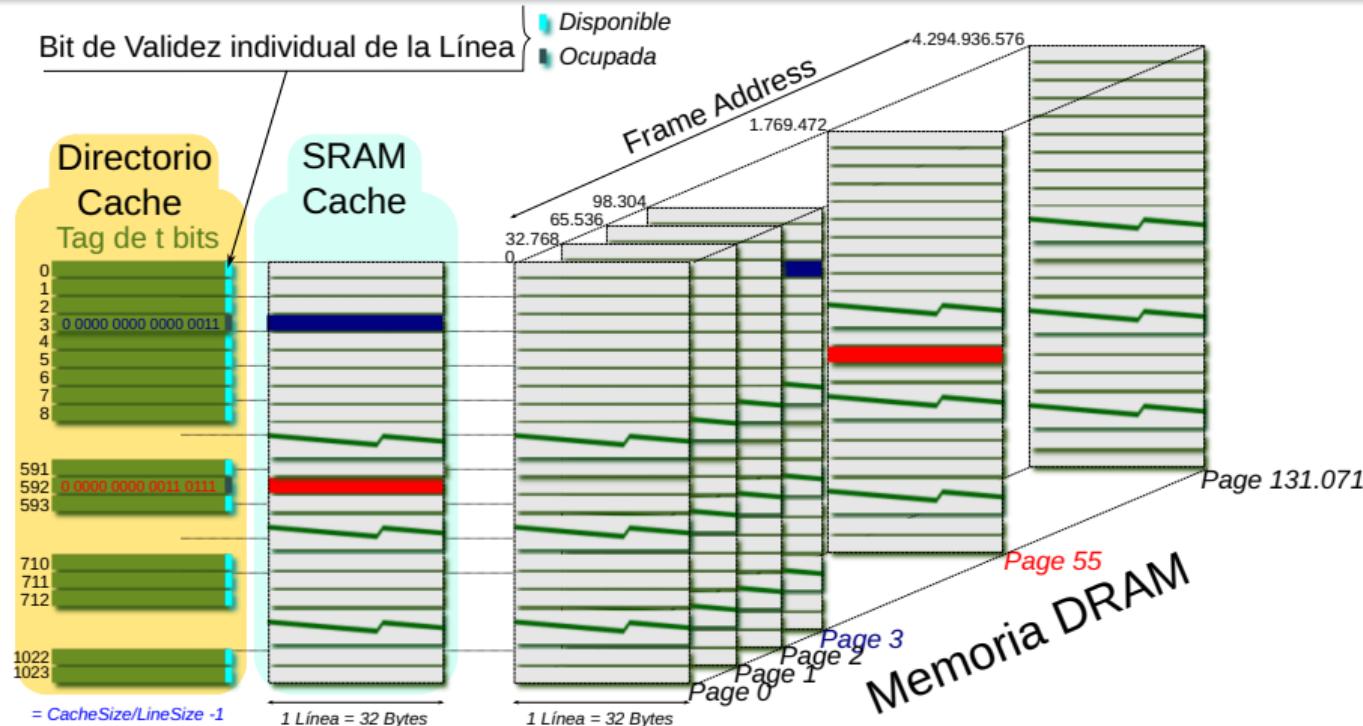
Los 4 GiB de memoria del sistema, son para el Controlador Cache una sucesión lineal de 134.217.728 líneas de 32 B c/u, agrupadas de a 1024 en 131.072 bloques o páginas de 32 KiB c/u.

# Line Placement: Cache de Mapeo Directo



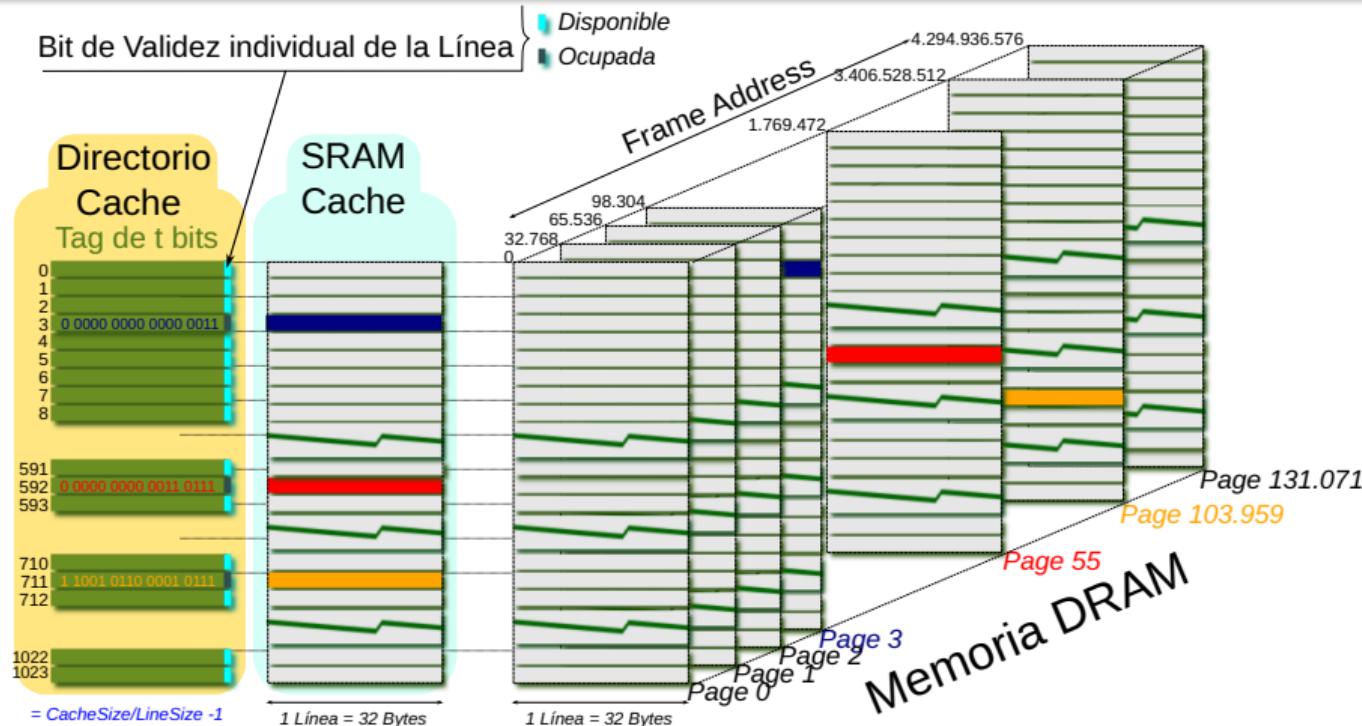
La CPU direcciona dentro del intervalo  $[3,145,824 - 3,145,855]$ , que es la línea 98.307, cuarta línea del cuarto bloque de memoria principal. Mapeo Directo la guarda en la 4to. línea del cache, y el tag será el nro. de bloque: 11b.

# Line Placement: Cache de Mapeo Directo



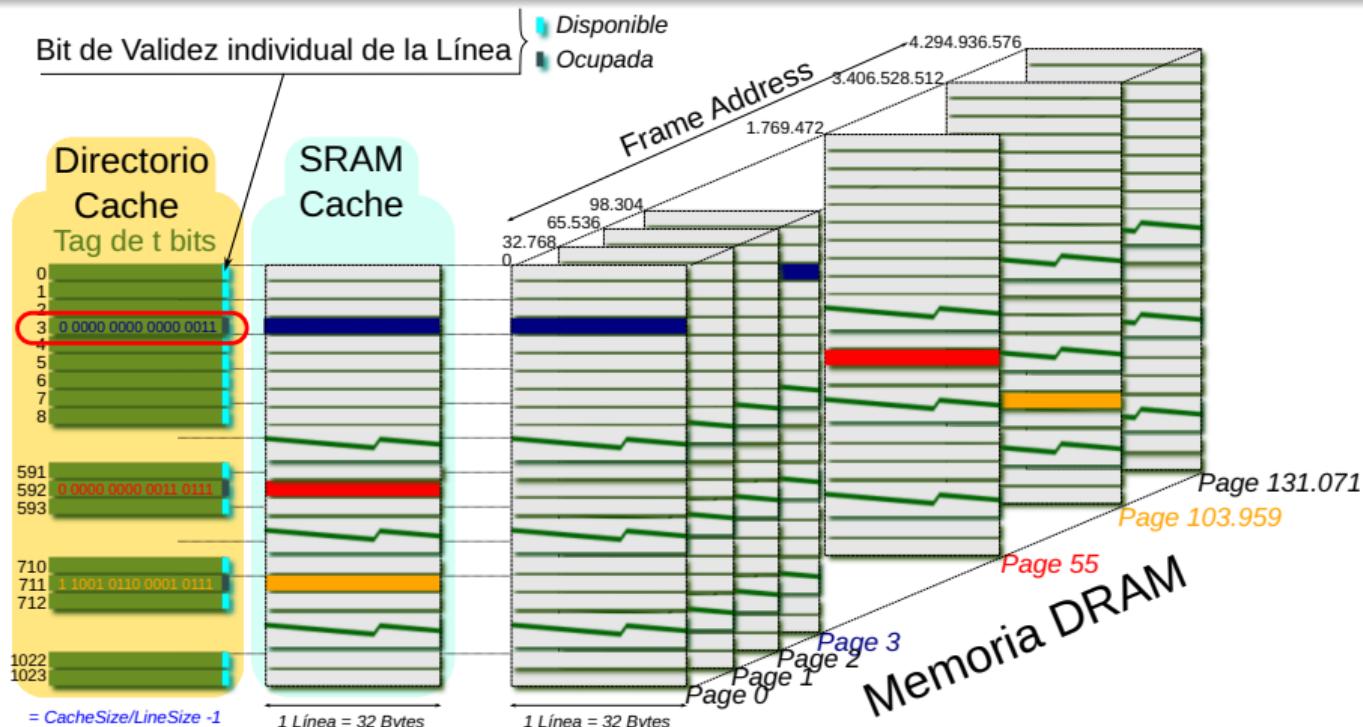
Del mismo modo para una dirección del intervalo [56,642,048 – 56,642,079], corresponde la línea 1.770.064 de la memoria principal. Es decir la línea 592, de la página 55. Y se guarda en la línea 592 del cache con tag 55 = 110111b

# Line Placement: Cache de Mapeo Directo



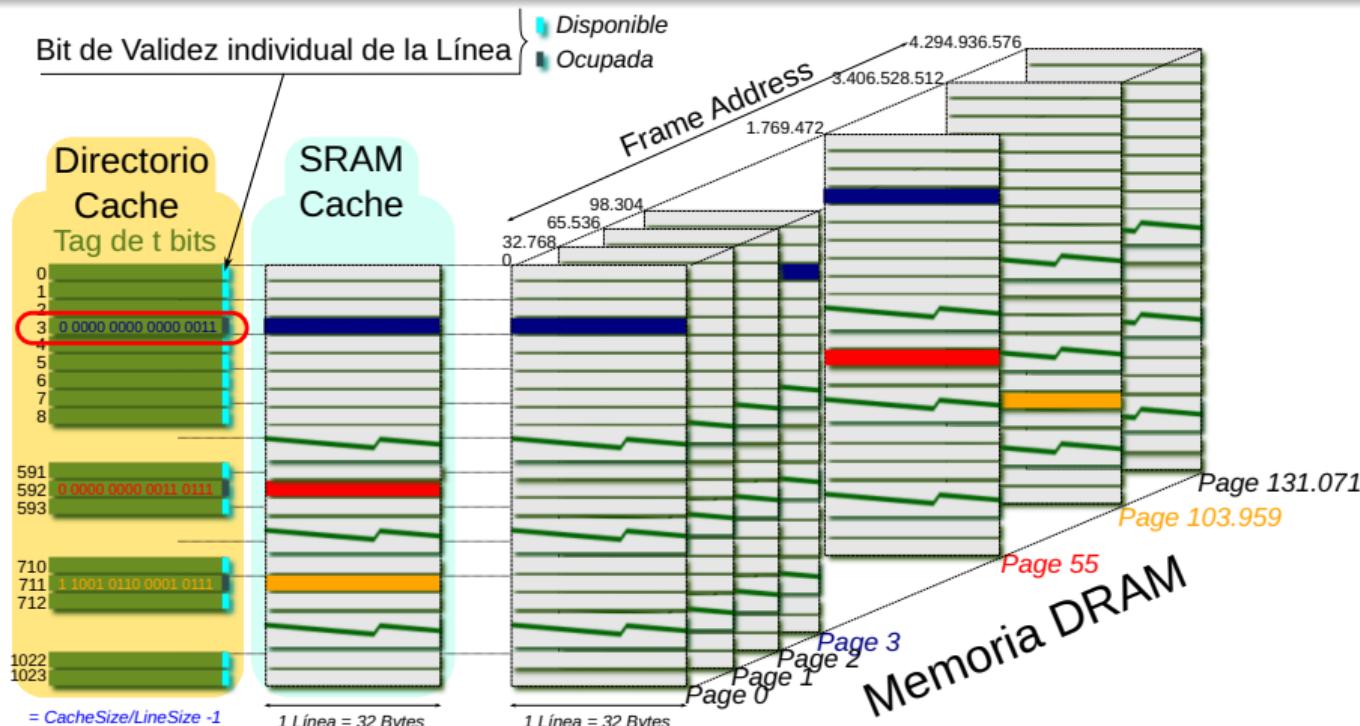
Y así para cualquier dirección. Todo va perfecto mientras no se repita ningún número de línea ya utilizado en el cache en una página diferente de la almacenada en éste.

# Line Placement: Cache de Mapeo Directo



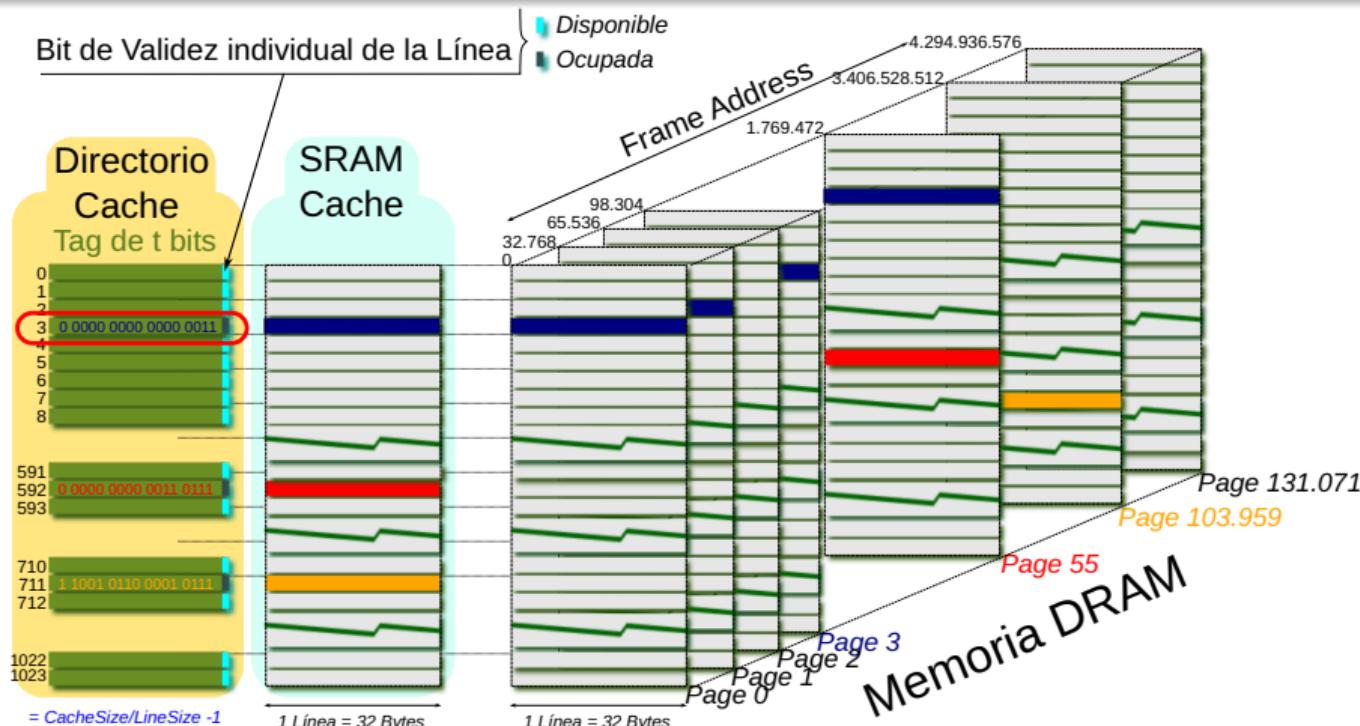
Si en éste estado de ocupación del cache la CPU requiriese por ejemplo la dirección 143 la línea 4 del cache debe ser reemplazada por la cuarta línea de la página 0. El tag se reemplaza por 00b en este caso.

# Line Placement: Cache de Mapeo Directo



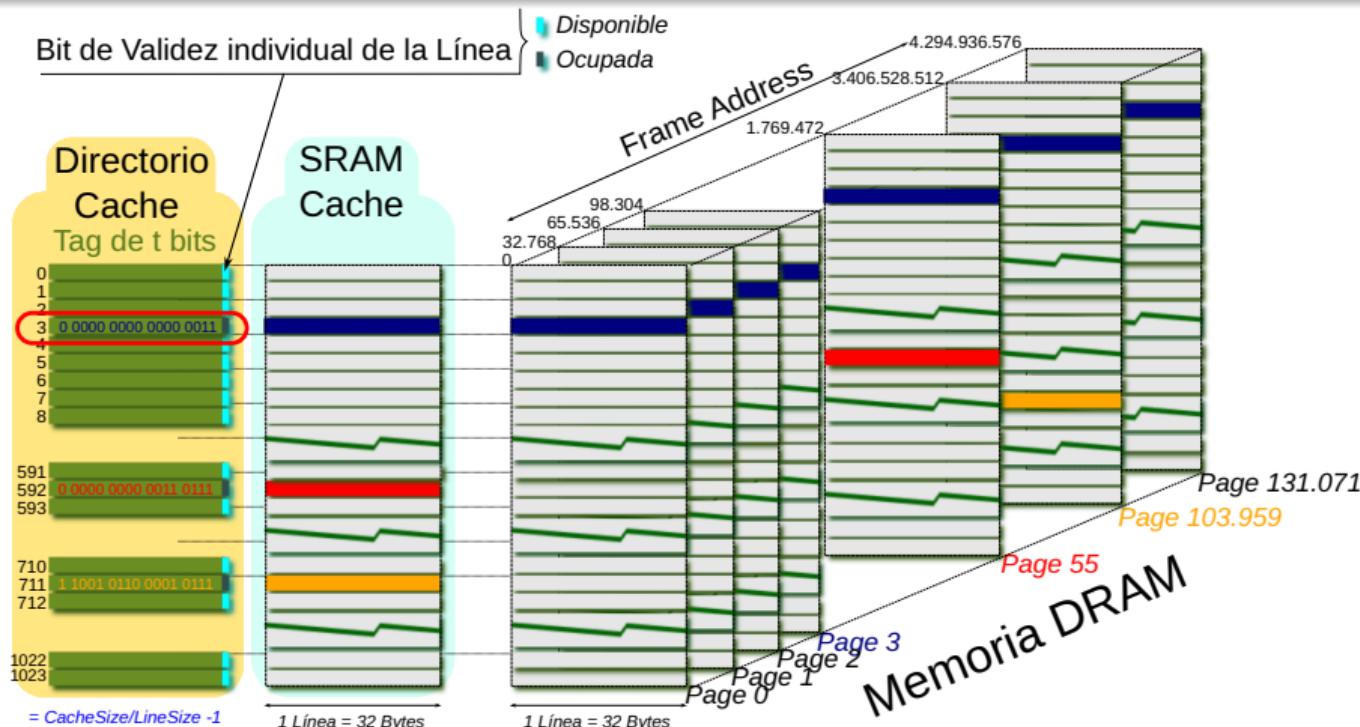
Este escenario se generará para cualquiera de las 131.071 líneas que ocupan la misma posición en las demás páginas de la memoria principal.

# Line Placement: Cache de Mapeo Directo



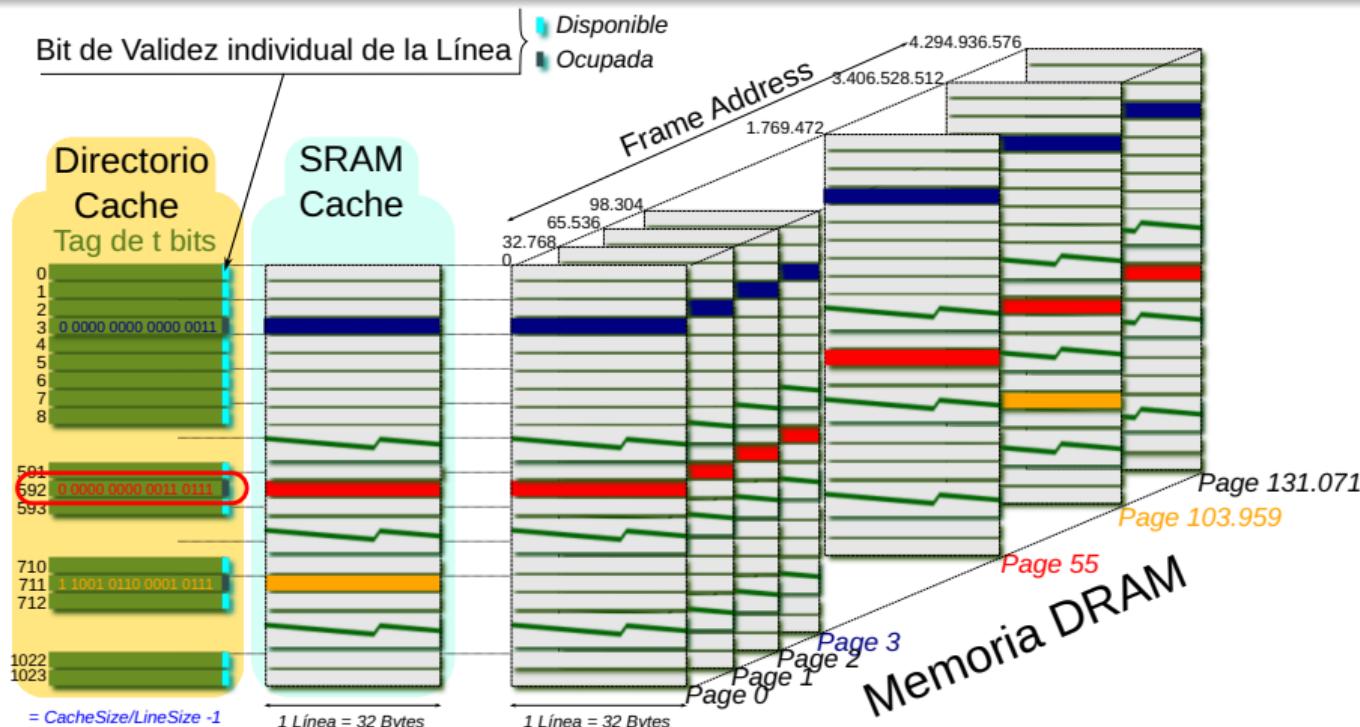
Este escenario se generará para cualquiera de las 131.071 líneas que ocupan la misma posición en las demás páginas de la memoria principal.

# Line Placement: Cache de Mapeo Directo



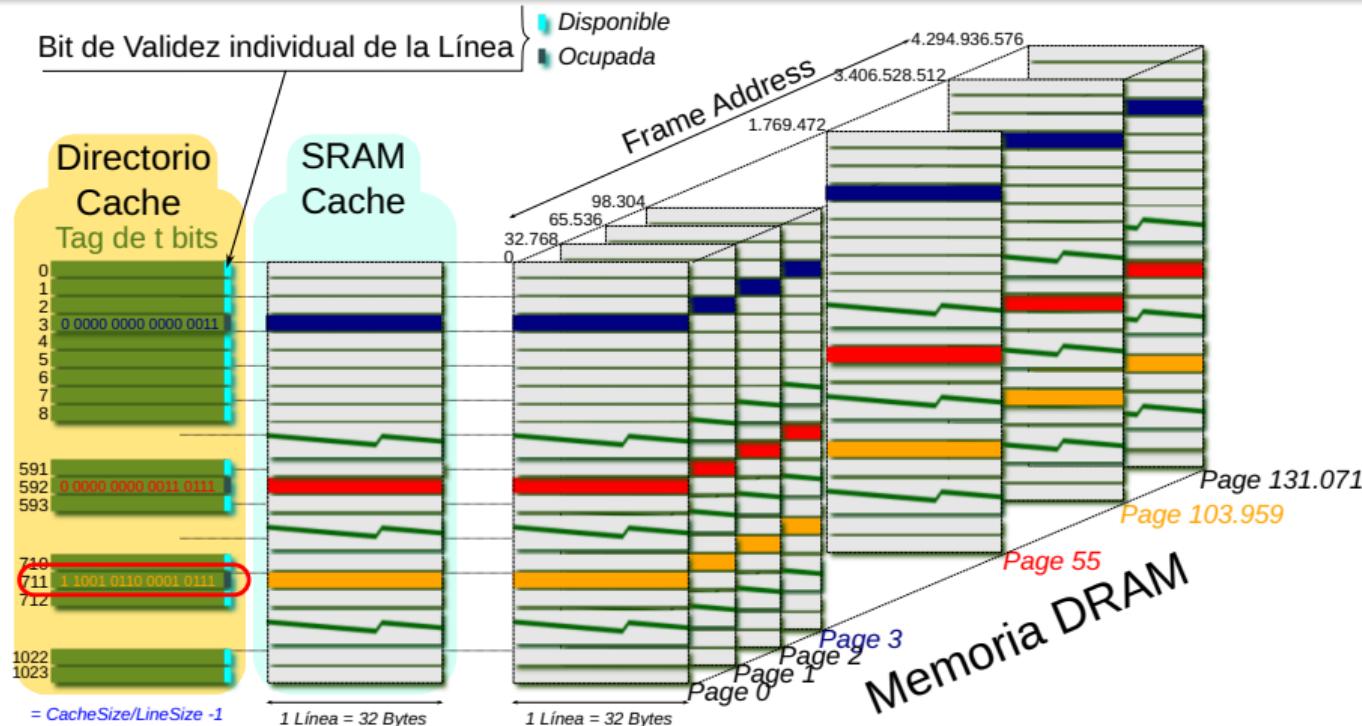
Este escenario se generará para cualquiera de las 131.071 líneas que ocupan la misma posición en las demás páginas de la memoria principal.

# Line Placement: Cache de Mapeo Directo



Este escenario se generará para cualquiera de las 131.071 líneas que ocupan la misma posición en las demás páginas de la memoria principal.

# Line Placement: Cache de Mapeo Directo



Si la CPU alterna con cierta frecuencia direcciones que traducen en un mismo número de línea, el **Conflict Miss** resultante degrada la performance.

# Line Placement: Cache Set-Asociativo

# Line Placement: Cache Set-Asociativo

- Trataremos de disminuir la probabilidad de **Conflict Miss**, con un costo de complejidad de hardware moderado y su consecuente costo energético.

# Line Placement: Cache Set-Asociativo

- Trataremos de disminuir la probabilidad de **Conflict Miss**, con un costo de complejidad de hardware moderado y su consecuente costo energético.
- El conflicto aparece por accesos recurrentes a direcciones de memoria que resultan en la misma línea en diferentes bloques o páginas de memoria principal.

## Line Placement: Cache Set-Asociativo

- Trataremos de disminuir la probabilidad de **Conflict Miss**, con un costo de complejidad de hardware moderado y su consecuente costo energético.
- El conflicto aparece por accesos recurrentes a direcciones de memoria que resultan en la misma línea en diferentes bloques o páginas de memoria principal.
- El Controlador Cache modifica la forma de organizar la Memoria Cache y la forma en que ésta mapea en la memoria principal.

# Line Placement: Cache Set-Asociativo

- Trataremos de disminuir la probabilidad de **Conflict Miss**, con un costo de complejidad de hardware moderado y su consecuente costo energético.
- El conflicto aparece por accesos recurrentes a direcciones de memoria que resultan en la misma línea en diferentes bloques o páginas de memoria principal.
- El Controlador Cache modifica la forma de organizar la Memoria Cache y la forma en que ésta mapea en la memoria principal.
- La Memoria Cache se organiza en bancos (o vías).

# Line Placement: Cache Set-Asociativo

- Trataremos de disminuir la probabilidad de **Conflict Miss**, con un costo de complejidad de hardware moderado y su consecuente costo energético.
- El conflicto aparece por accesos recurrentes a direcciones de memoria que resultan en la misma línea en diferentes bloques o páginas de memoria principal.
- El Controlador Cache modifica la forma de organizar la Memoria Cache y la forma en que ésta mapea en la memoria principal.
- La Memoria Cache se organiza en bancos (o vías).
- La Memoria Principal sigue siendo tratada como un conjunto de bloques o páginas, pero el tamaño de c/u ahora es igual al de cada vía.

# Line Placement: Cache Set-Asociativo

- Trataremos de disminuir la probabilidad de **Conflict Miss**, con un costo de complejidad de hardware moderado y su consecuente costo energético.
- El conflicto aparece por accesos recurrentes a direcciones de memoria que resultan en la misma línea en diferentes bloques o páginas de memoria principal.
- El Controlador Cache modifica la forma de organizar la Memoria Cache y la forma en que ésta mapea en la memoria principal.
- La Memoria Cache se organiza en bancos (o vías).
- La Memoria Principal sigue siendo tratada como un conjunto de bloques o páginas, pero el tamaño de c/u ahora es igual al de cada vía.
- Si la Memoria Cache de 32 KiB del ejemplo anterior, se organiza como set-asociativa de 8 vías, cada bloque de Memoria del sistema será de 4 KiB.

# Line Placement: Cache Set-Asociativo

- Si el tamaño de línea se mantiene (32 B), cada vía ahora contiene 128 líneas.

## Line Placement: Cache Set-Asociativo

- Si el tamaño de línea se mantiene (32 B), cada vía ahora contiene 128 líneas.
- El espacio de direccionamiento de 4 GiB, es tratado por el Controlador Cache como una sucesión lineal de 134.217.728 líneas de 32 B c/u, al igual que en el caso de Mapeo Directo ya que el tamaño de línea no se ha alterado.

## Line Placement: Cache Set-Asociativo

- Si el tamaño de línea se mantiene (32 B), cada vía ahora contiene 128 líneas.
- El espacio de direccionamiento de 4 GiB, es tratado por el Controlador Cache como una sucesión lineal de 134.217.728 líneas de 32 B c/u, al igual que en el caso de Mapeo Directo ya que el tamaño de línea no se ha alterado.
- El tamaño del bloque o página sí cambia: de 1024 a 128 líneas.

## Line Placement: Cache Set-Asociativo

- Si el tamaño de línea se mantiene (32 B), cada vía ahora contiene 128 líneas.
- El espacio de direccionamiento de 4 GiB, es tratado por el Controlador Cache como una sucesión lineal de 134.217.728 líneas de 32 B c/u, al igual que en el caso de Mapeo Directo ya que el tamaño de línea no se ha alterado.
- El tamaño del bloque o página sí cambia: de 1024 a 128 líneas.
- Si los bloques contiene menos líneas que en Mapeo Directo, y el espacio de direccionamiento se mantiene en 4 GiB la cantidad de bloques o páginas en que se organiza la Memoria del sistema es lógico que aumente: de 131.072 ( $2^{17}$ ) a 1.048.576 ( $2^{20}$ ).

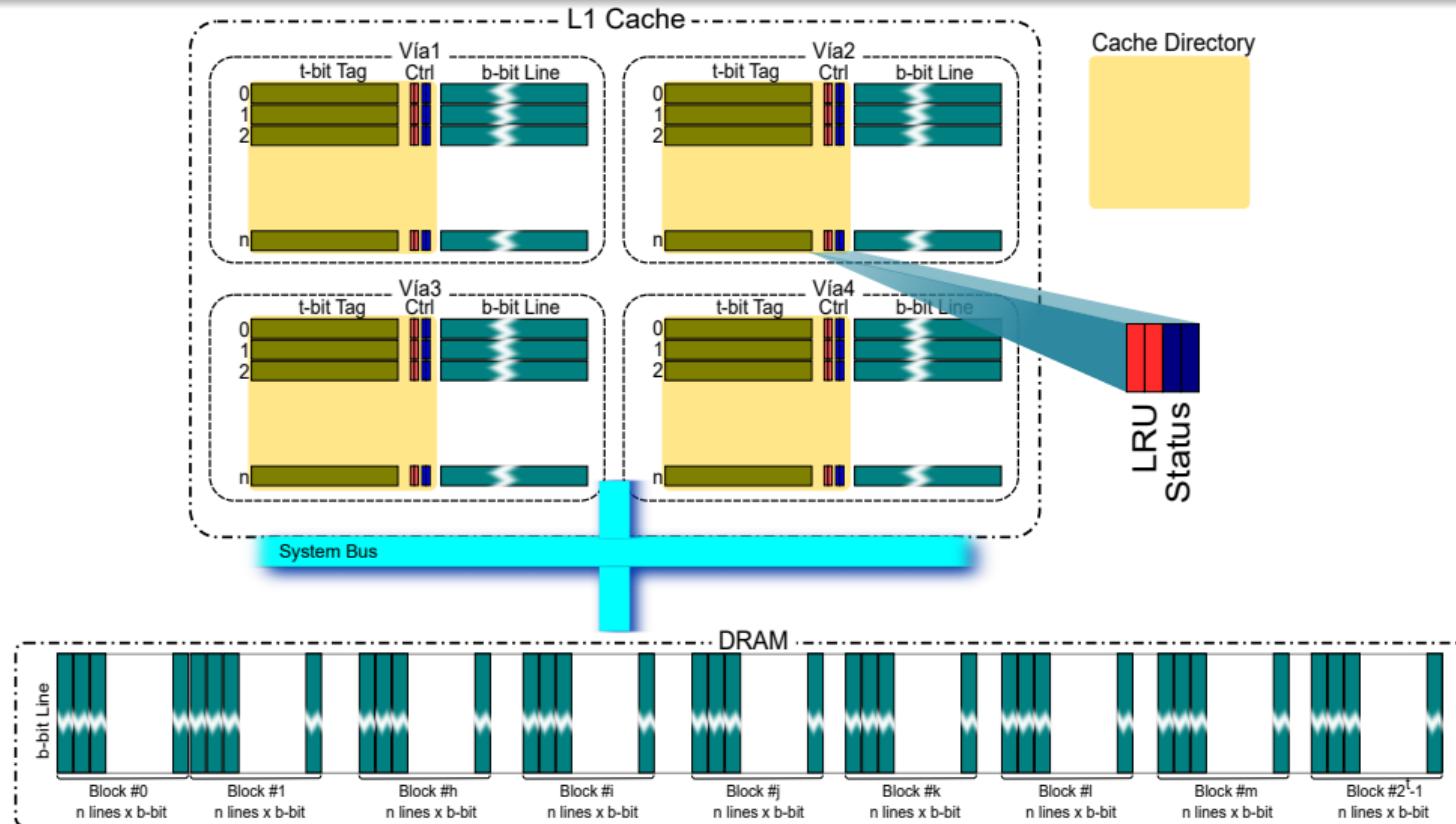
## Line Placement: Cache Set-Asociativo

- Si el tamaño de línea se mantiene (32 B), cada vía ahora contiene 128 líneas.
- El espacio de direccionamiento de 4 GiB, es tratado por el Controlador Cache como una sucesión lineal de 134.217.728 líneas de 32 B c/u, al igual que en el caso de Mapeo Directo ya que el tamaño de línea no se ha alterado.
- El tamaño del bloque o página sí cambia: de 1024 a 128 líneas.
- Si los bloques contiene menos líneas que en Mapeo Directo, y el espacio de direccionamiento se mantiene en 4 GiB la cantidad de bloques o páginas en que se organiza la Memoria del sistema es lógico que aumente: de 131.072 ( $2^{17}$ ) a 1.048.576 ( $2^{20}$ ).
- Esto modifica le tamaño del Tag de 17 a 20 bits para este caso.

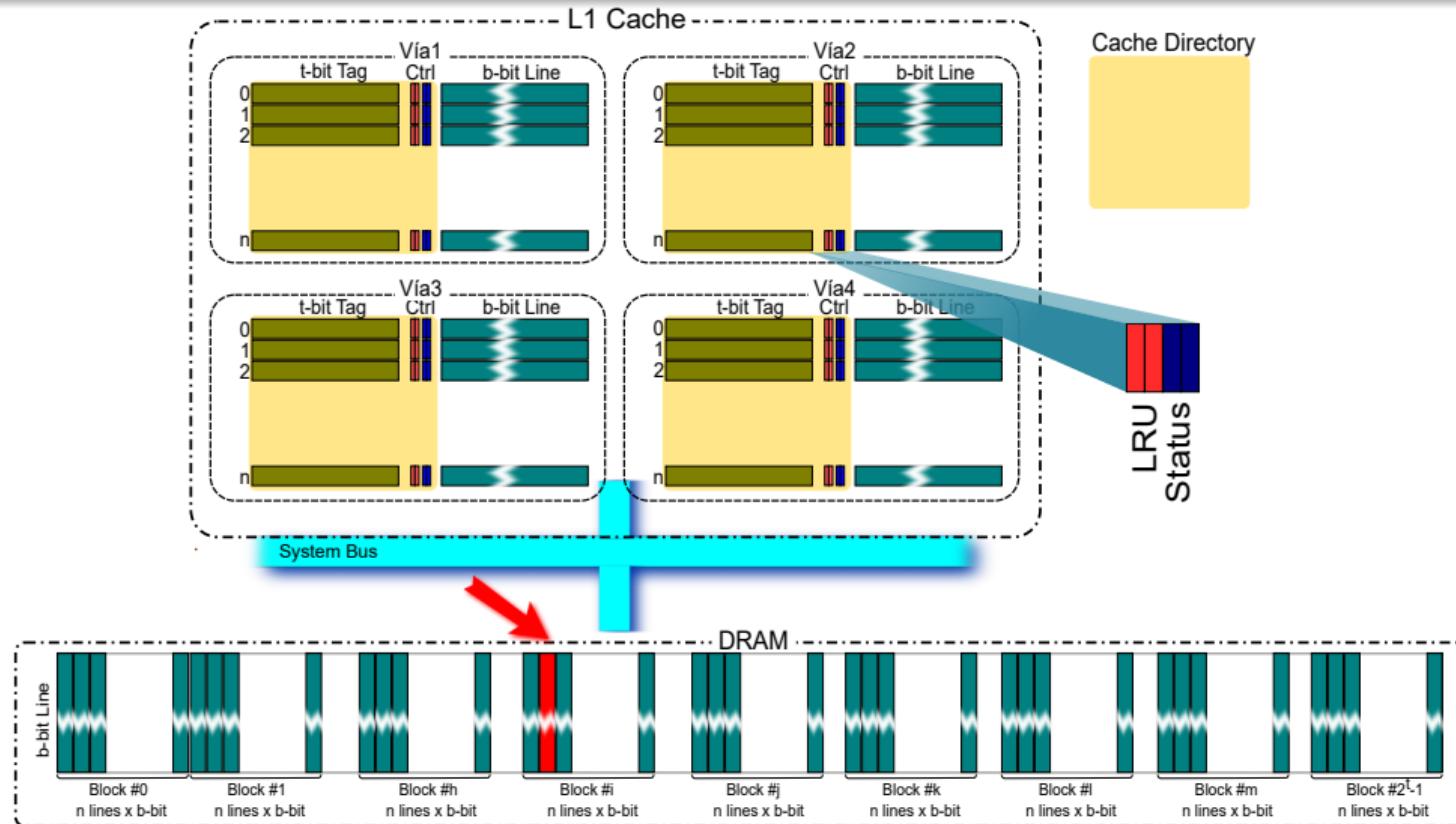
## Line Placement: Cache Set-Asociativo

- Si el tamaño de línea se mantiene (32 B), cada vía ahora contiene 128 líneas.
- El espacio de direccionamiento de 4 GiB, es tratado por el Controlador Cache como una sucesión lineal de 134.217.728 líneas de 32 B c/u, al igual que en el caso de Mapeo Directo ya que el tamaño de línea no se ha alterado.
- El tamaño del bloque o página sí cambia: de 1024 a 128 líneas.
- Si los bloques contiene menos líneas que en Mapeo Directo, y el espacio de direccionamiento se mantiene en 4 GiB la cantidad de bloques o páginas en que se organiza la Memoria del sistema es lógico que aumente: de 131.072 ( $2^{17}$ ) a 1.048.576 ( $2^{20}$ ).
- Esto modifica le tamaño del Tag de 17 a 20 bits para este caso.
- También requiere agregar bits de control para manejar el desalojo de un ítem ante un **Conflict Miss**, identificando al ítem cuya probabilidad de reutilización sea menor (criterio de vecindad).

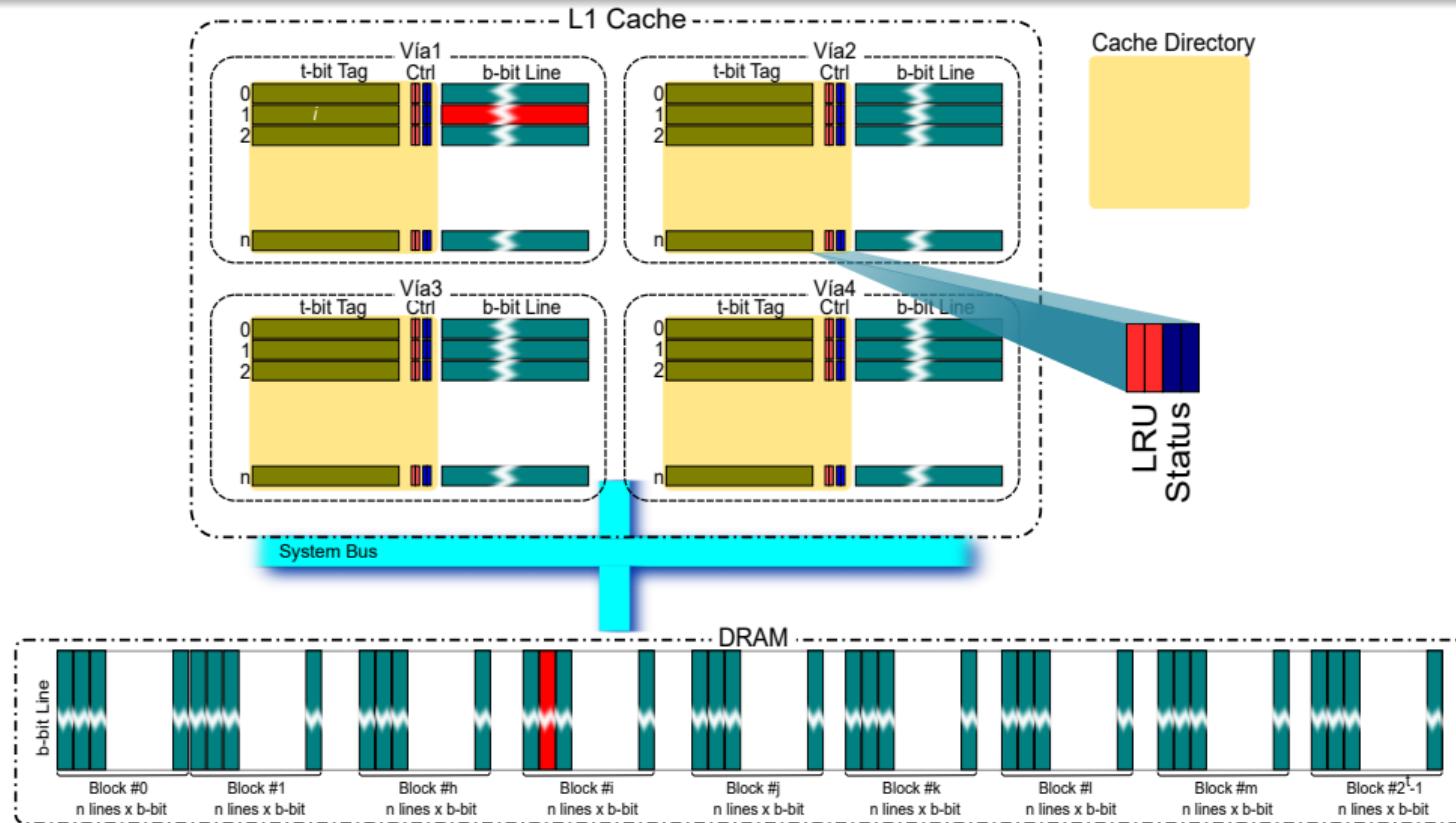
# Line Placement: Cache Set-Asociativo de 4 Vías



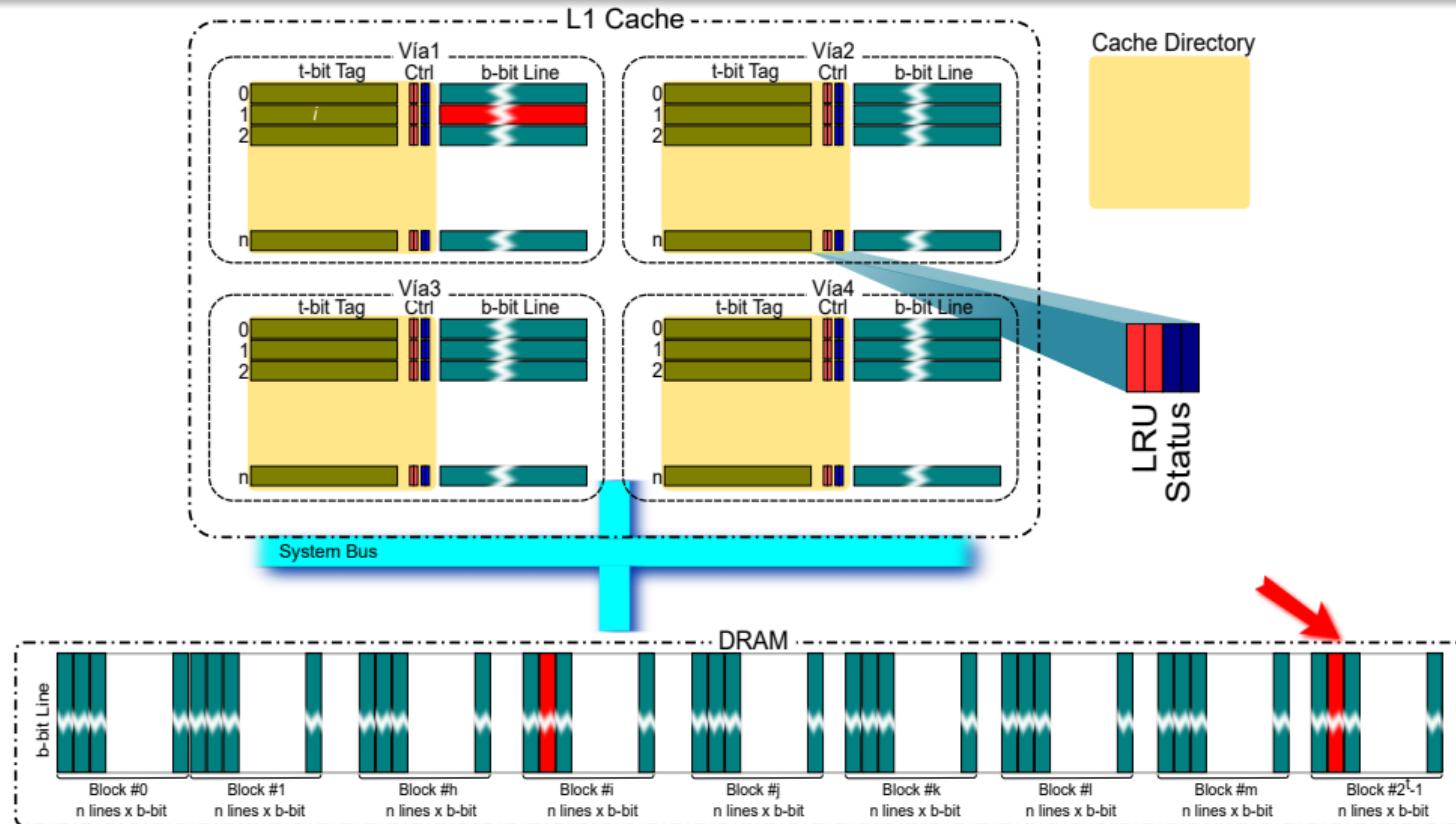
# Line Placement: Cache Set-Asociativo de 4 Vías



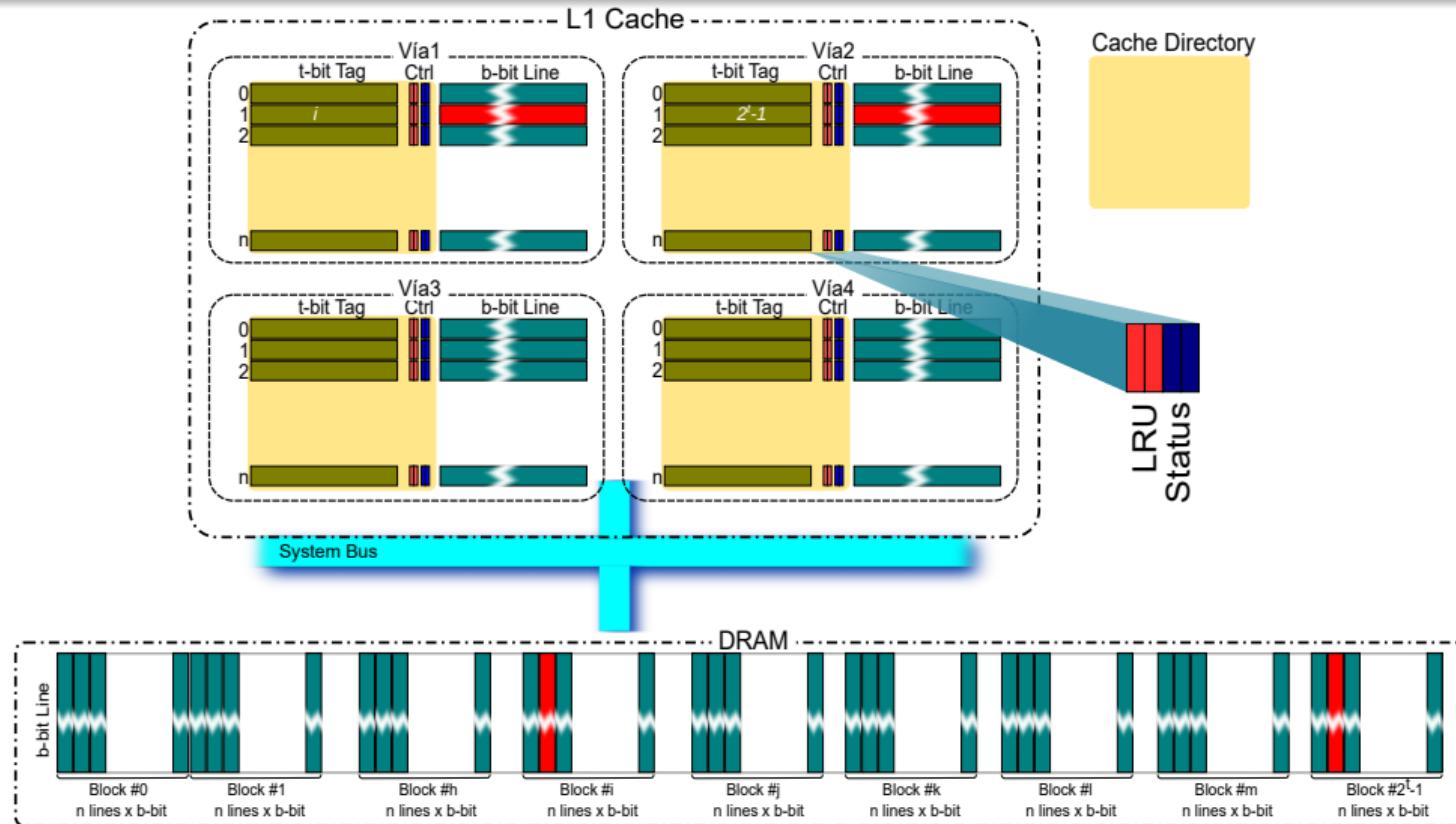
# Line Placement: Cache Set-Asociativo de 4 Vías



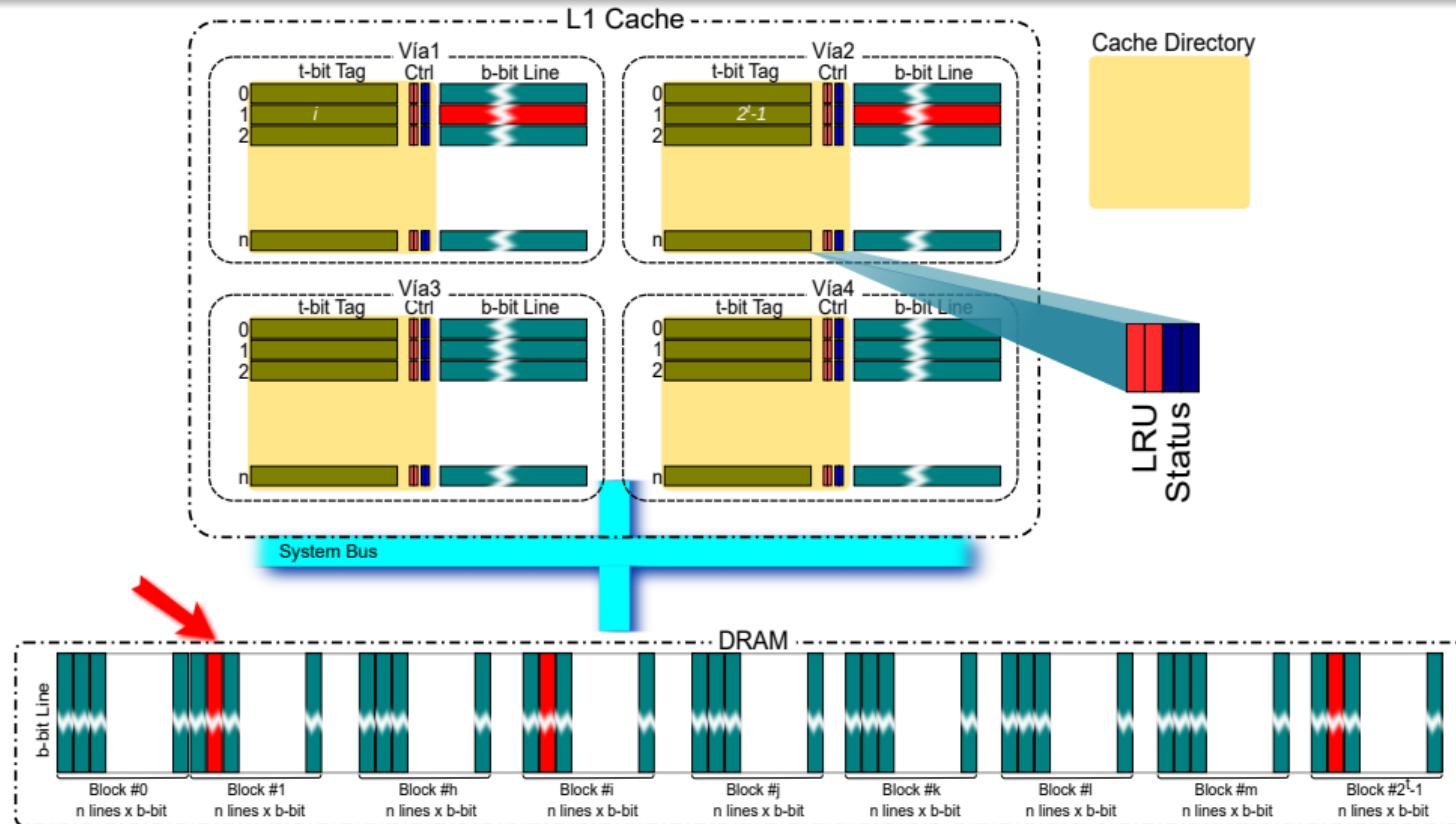
# Line Placement: Cache Set-Asociativo de 4 Vías



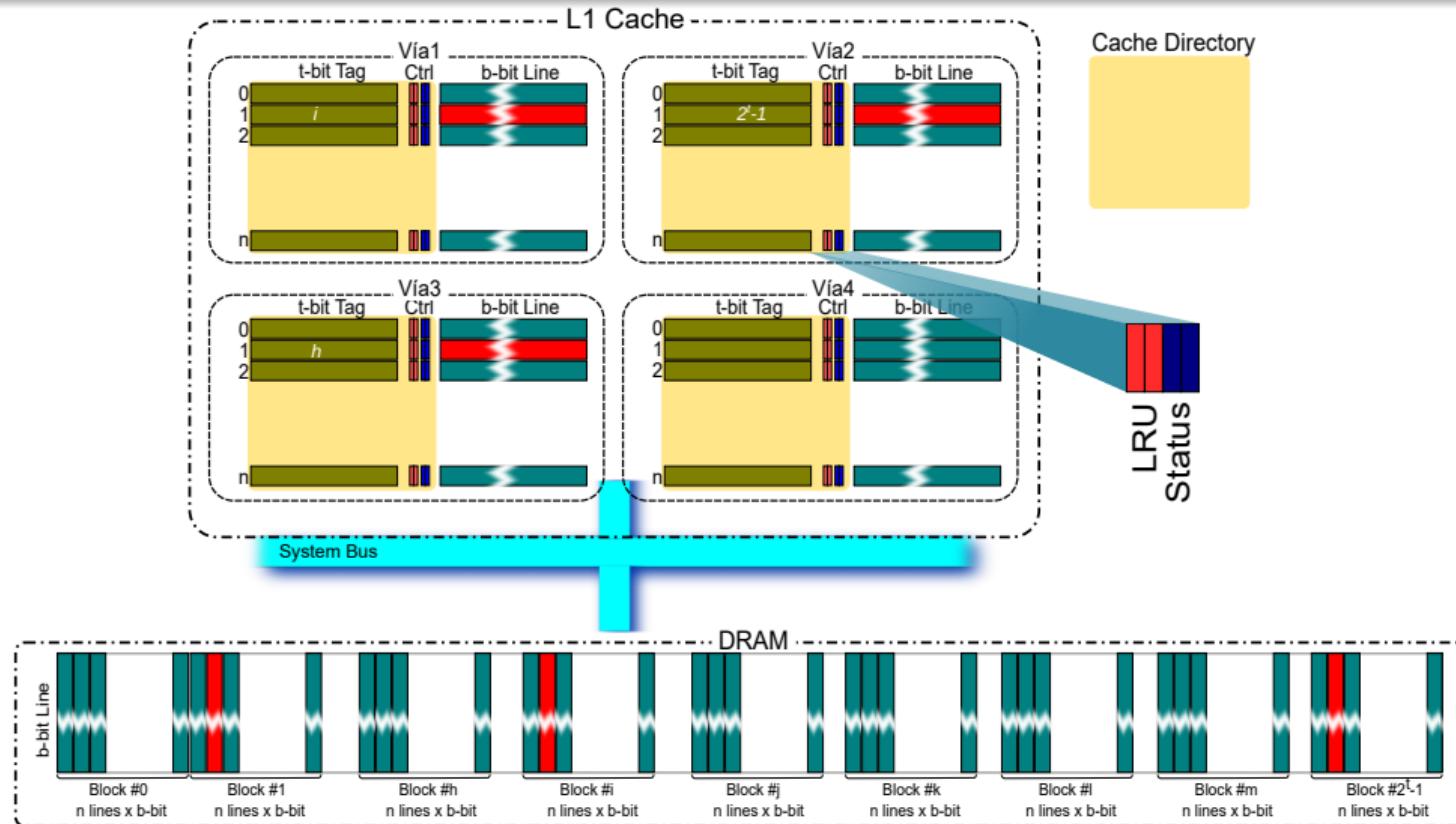
# Line Placement: Cache Set-Asociativo de 4 Vías



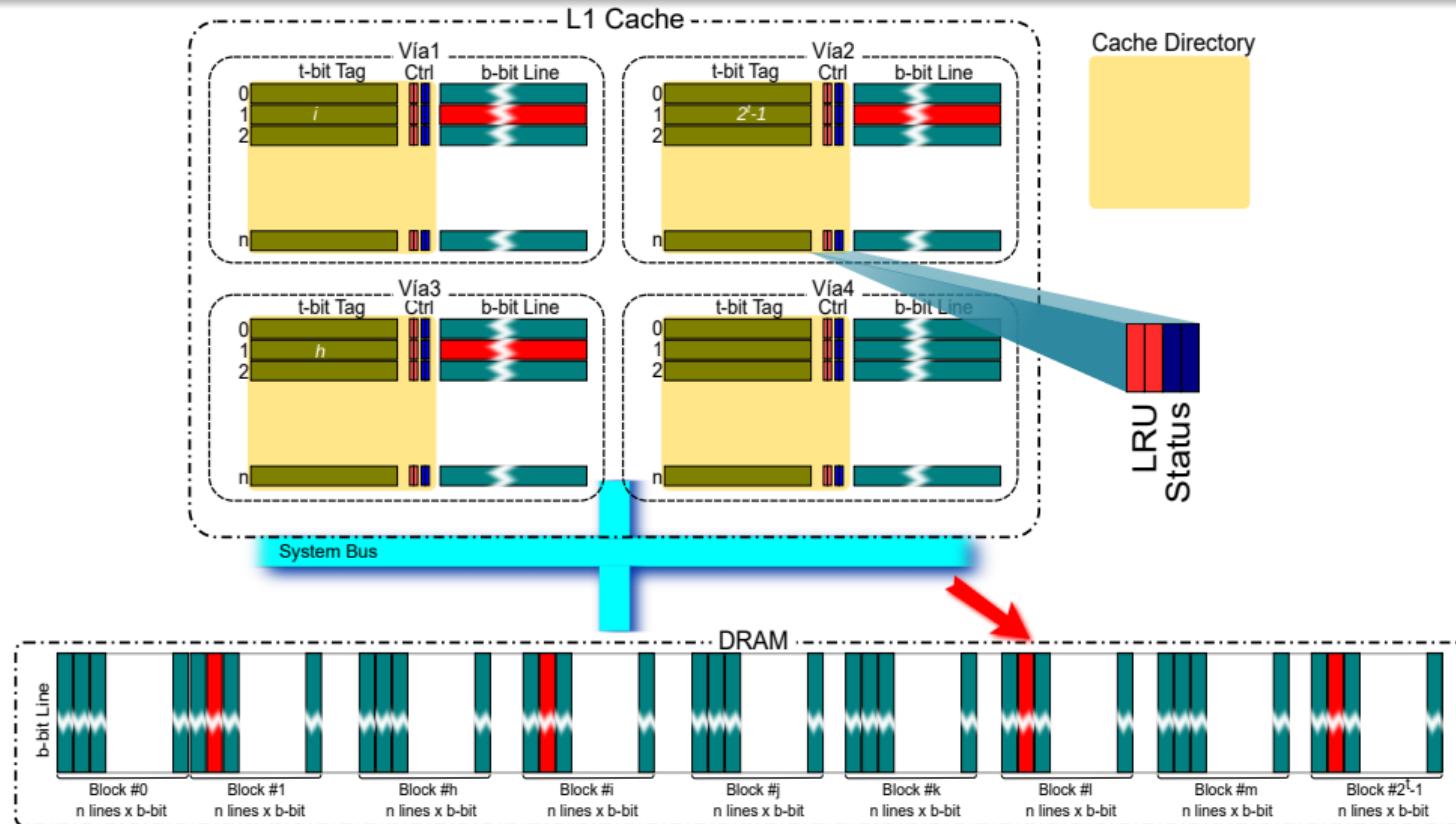
# Line Placement: Cache Set-Asociativo de 4 Vías



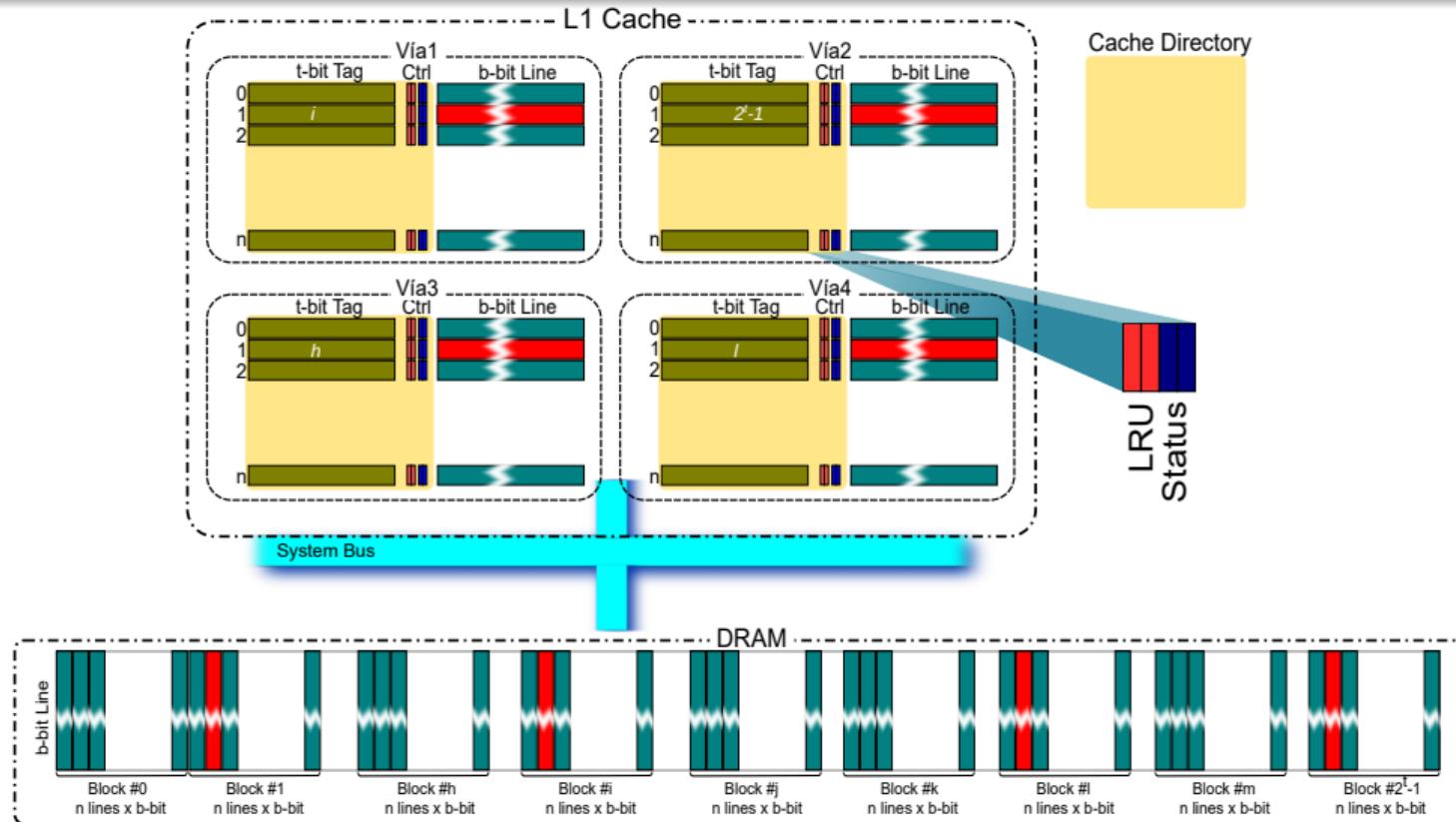
# Line Placement: Cache Set-Asociativo de 4 Vías



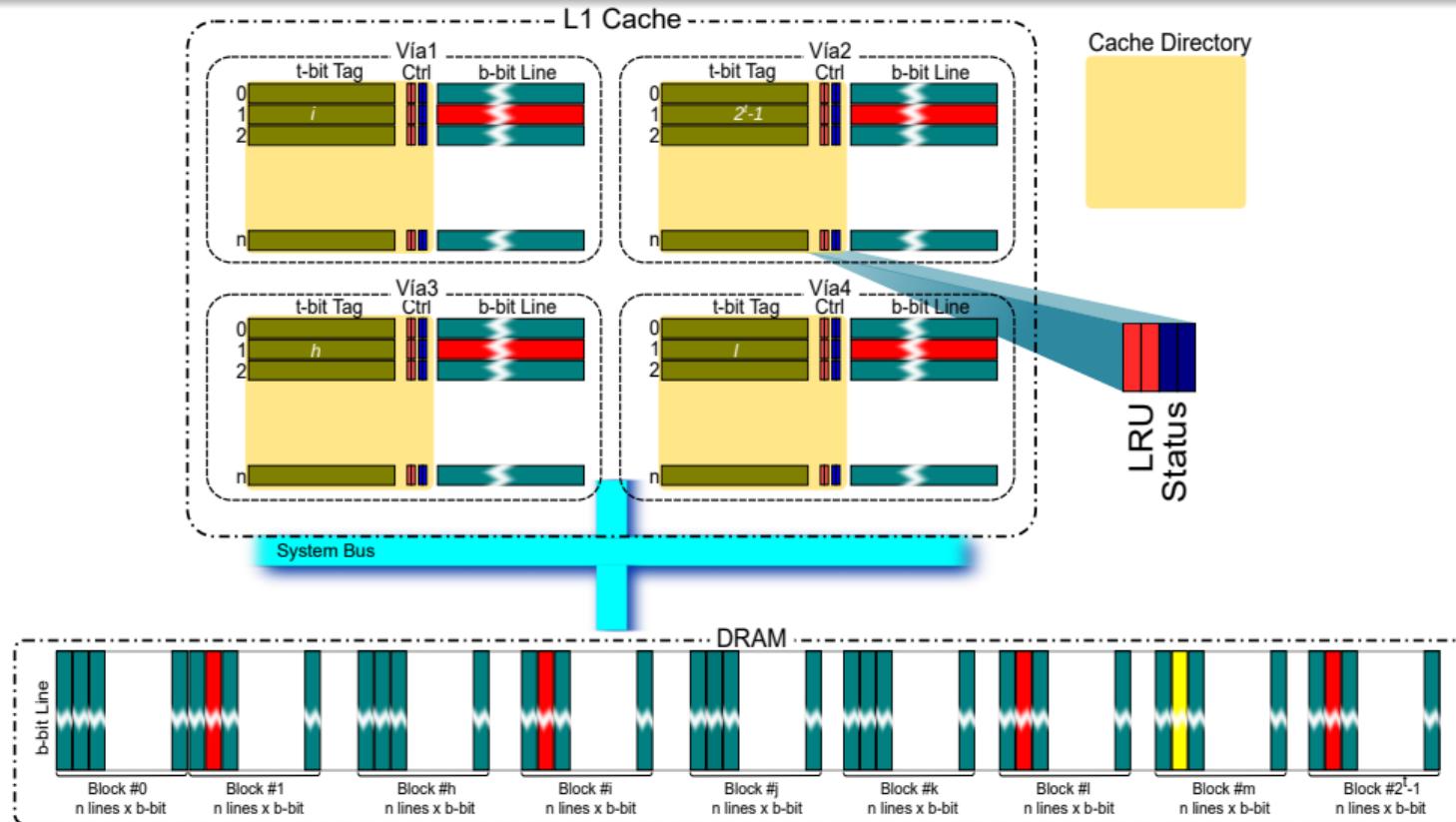
# Line Placement: Cache Set-Asociativo de 4 Vías



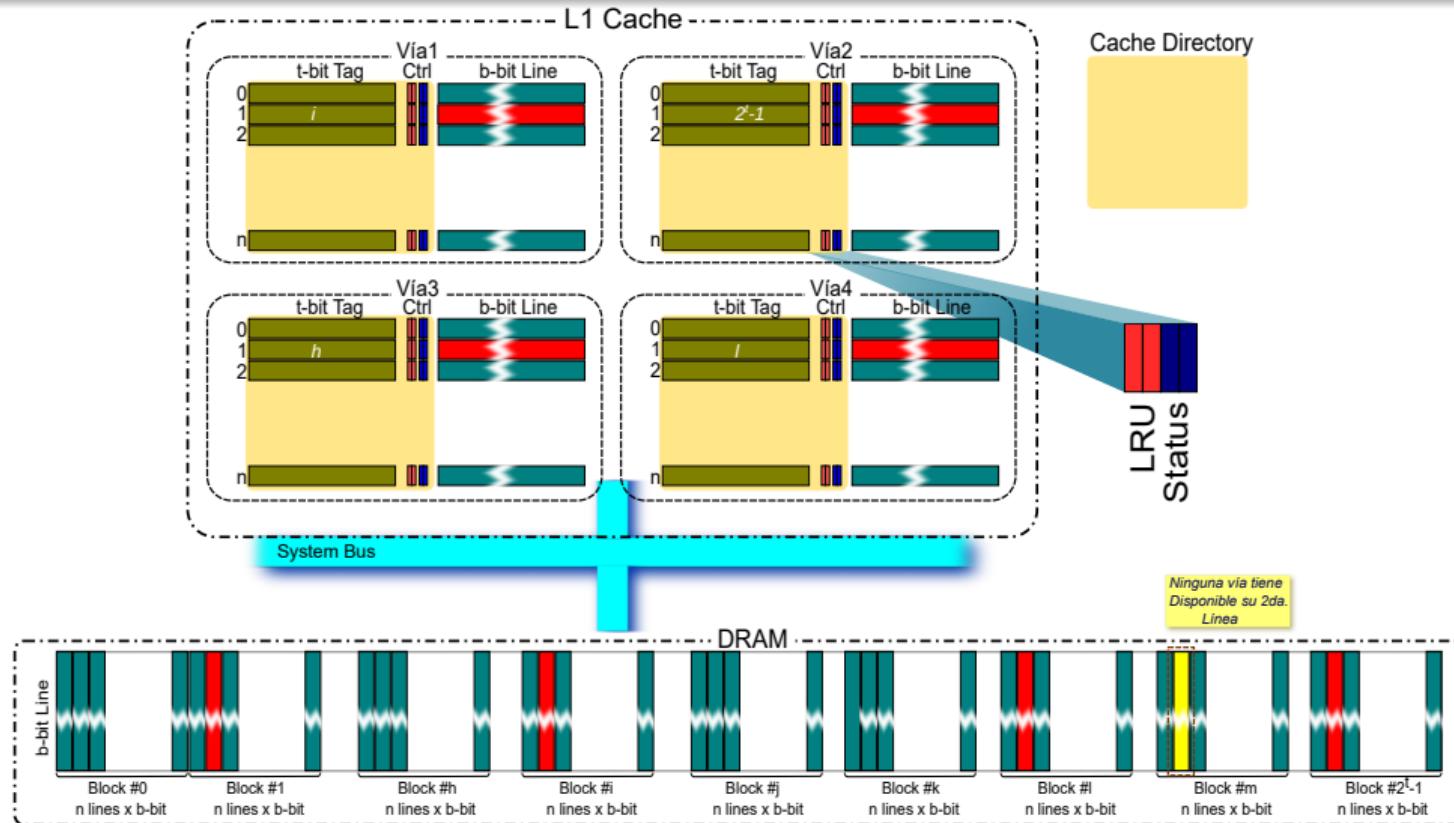
# Line Placement: Cache Set-Asociativo de 4 Vías



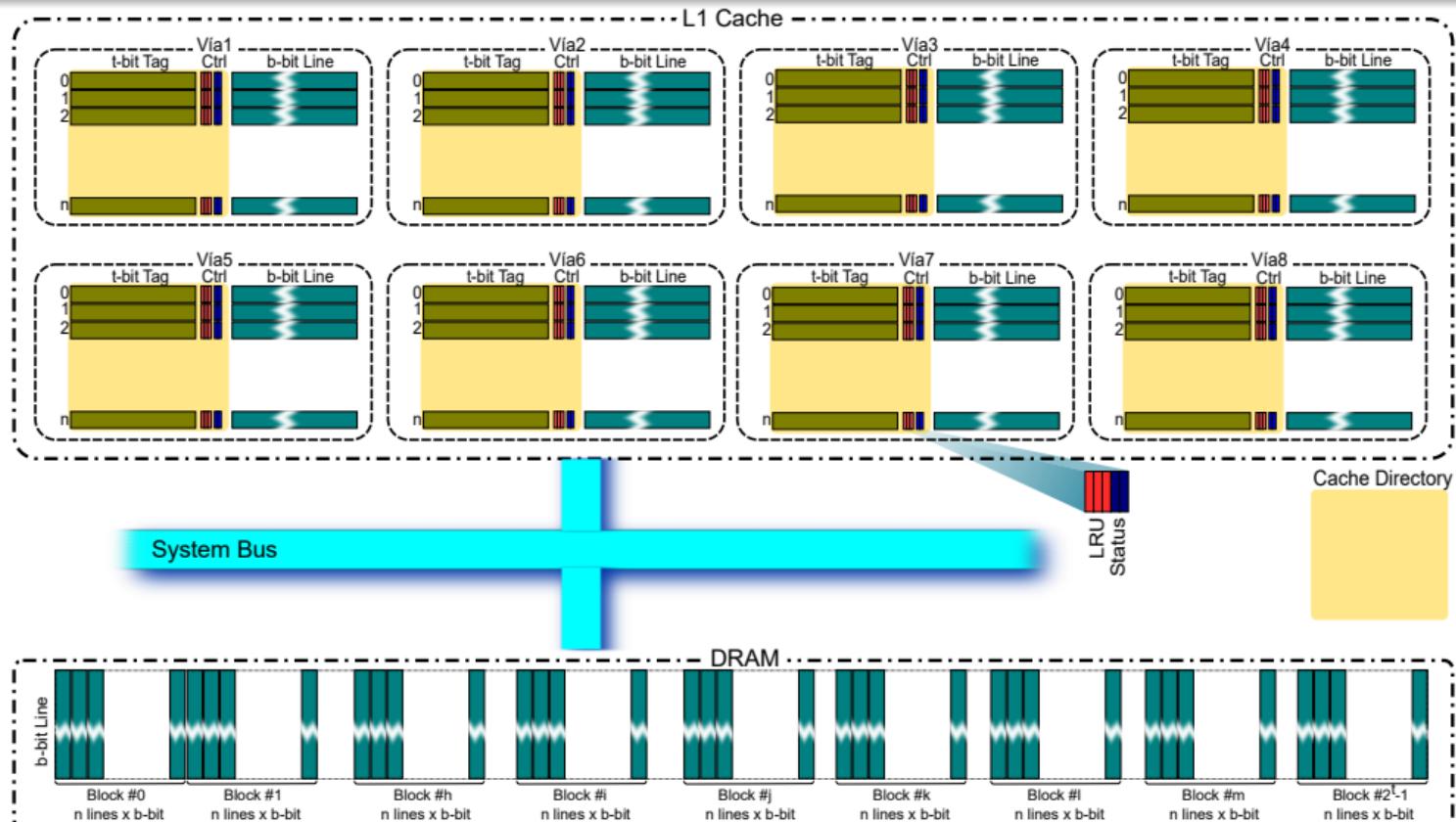
# Line Placement: Cache Set-Asociativo de 4 Vías



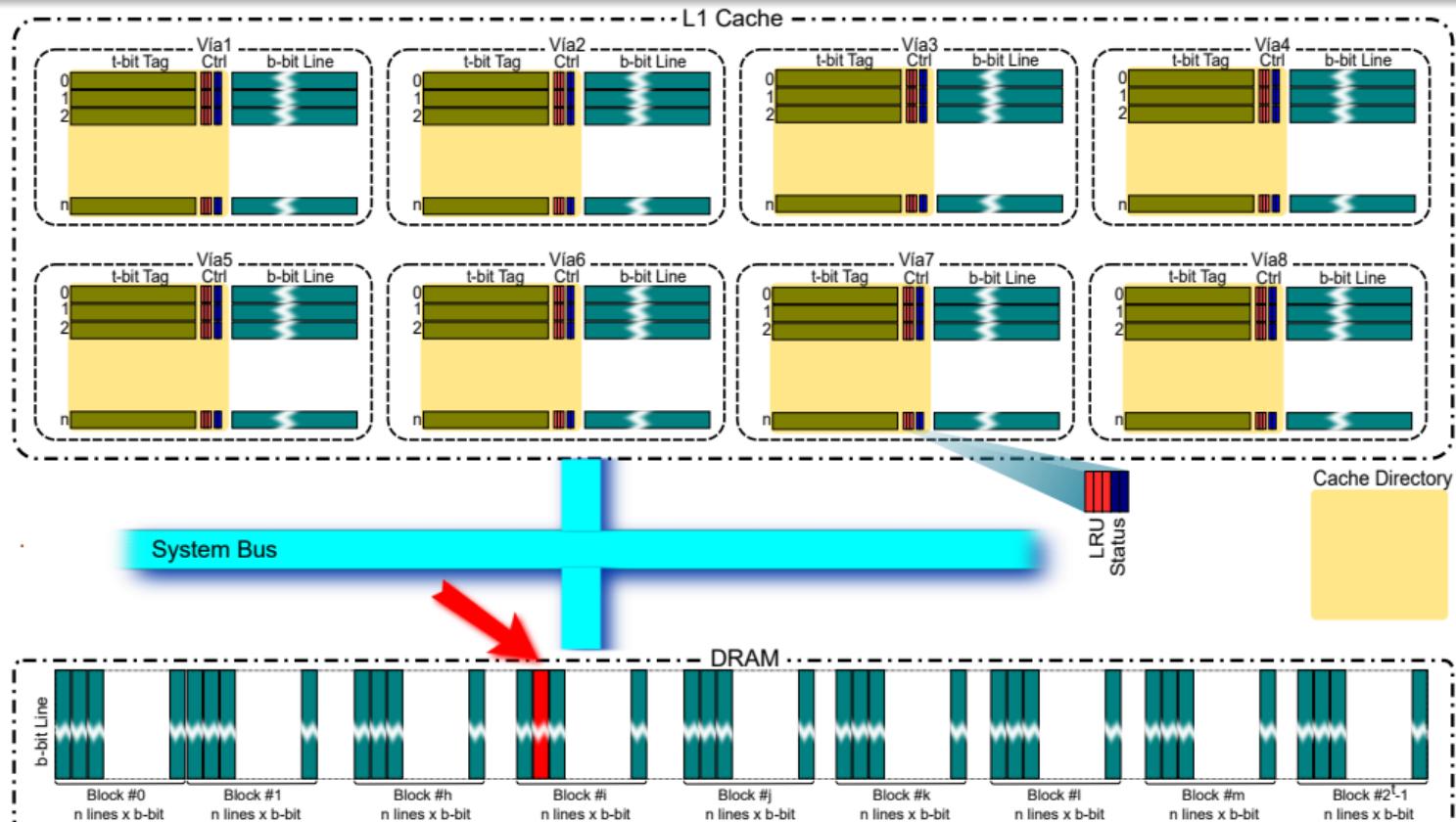
# Line Placement: Cache Set-Asociativo de 4 Vías



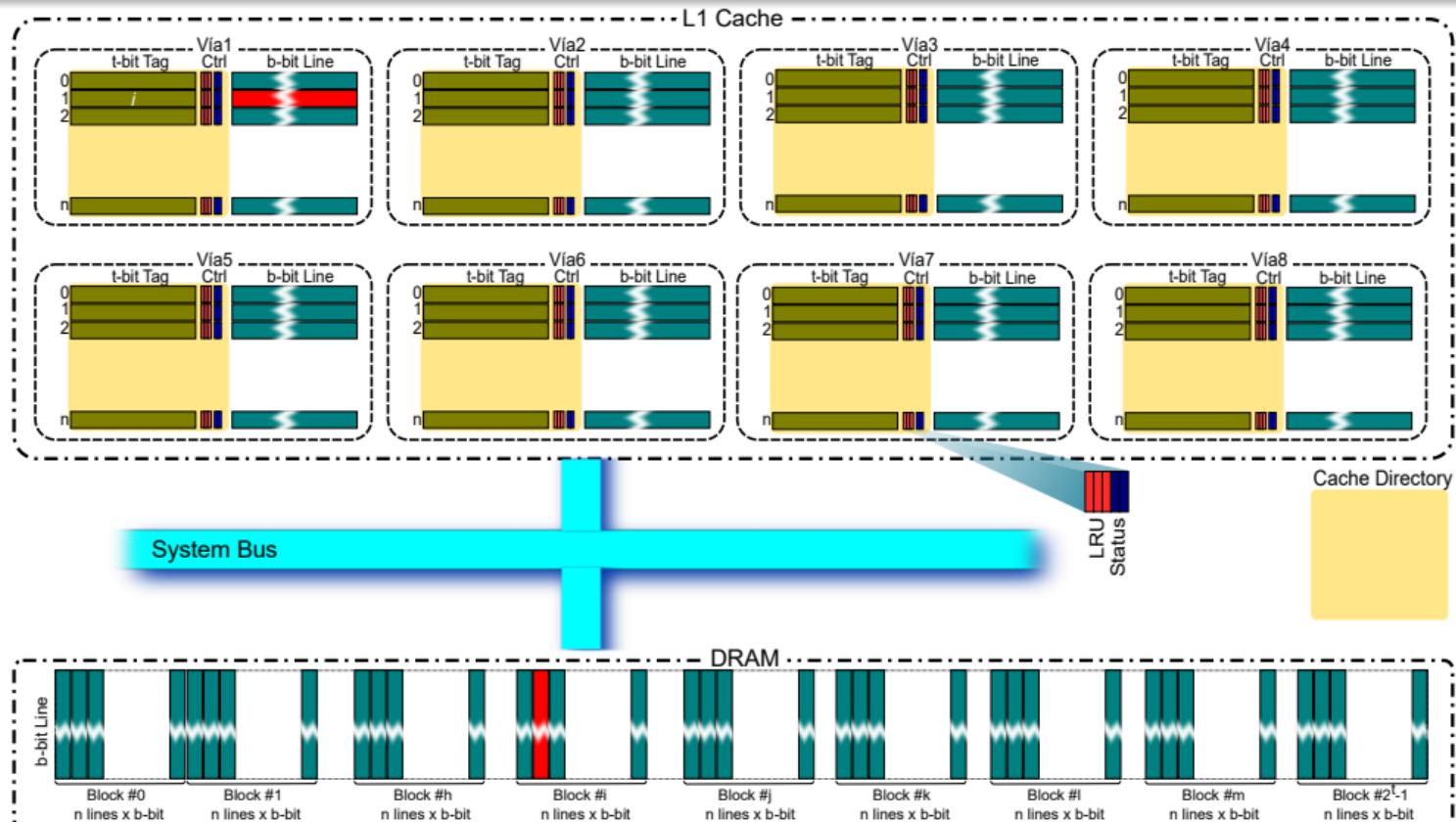
# Line Placement: Cache Set-Asociativo de 8 Vías



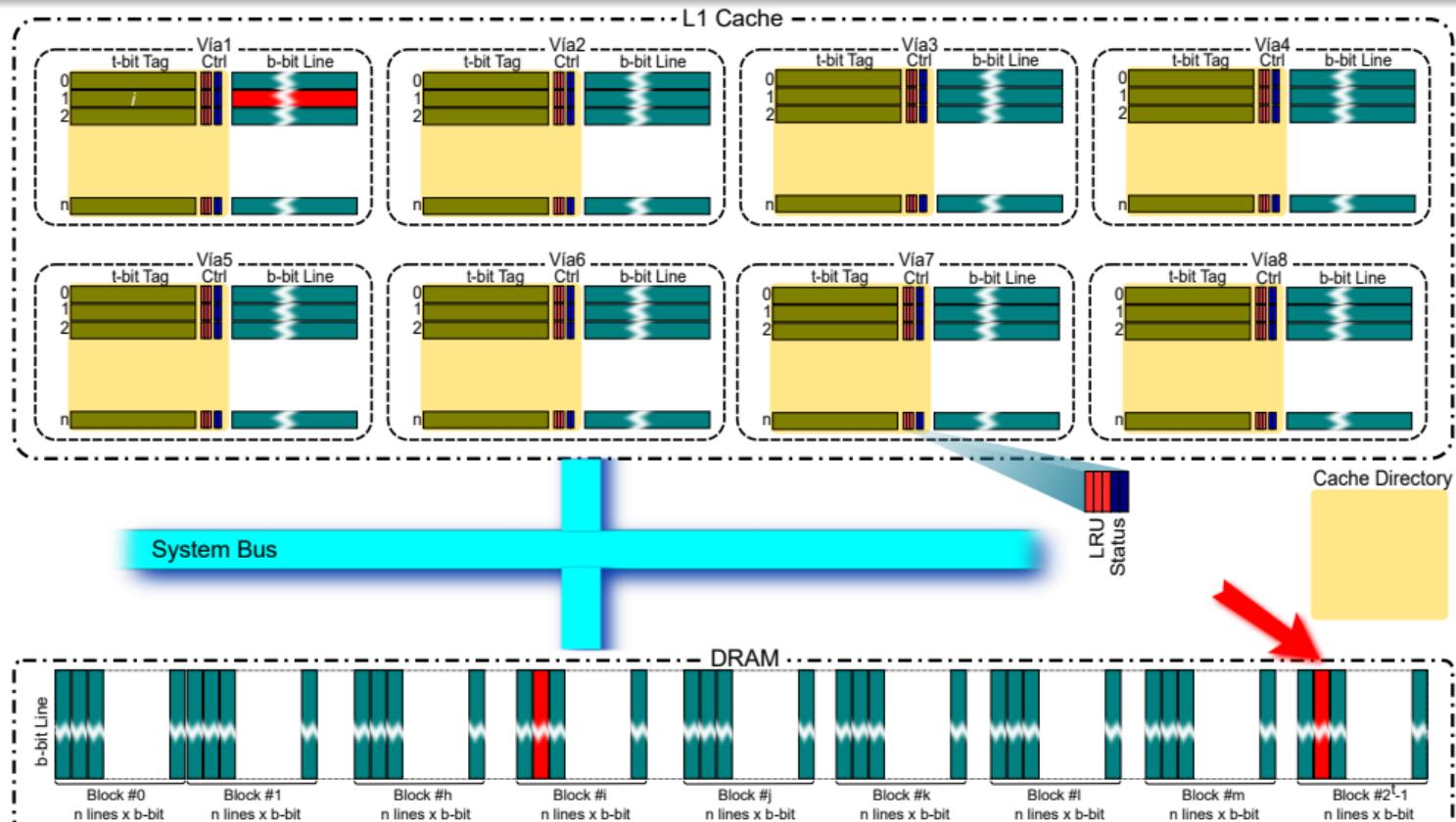
# Line Placement: Cache Set-Asociativo de 8 Vías



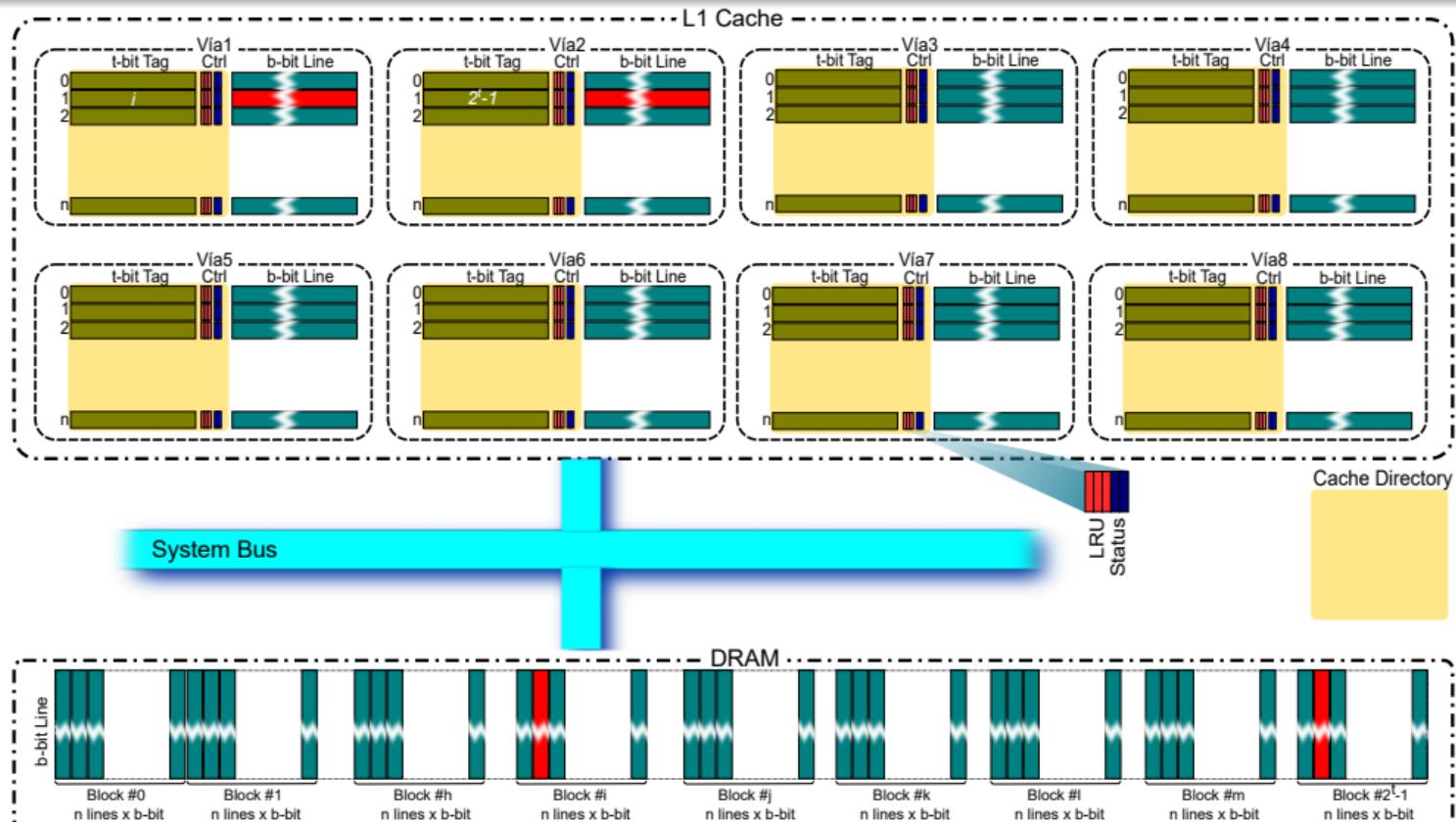
# Line Placement: Cache Set-Asociativo de 8 Vías



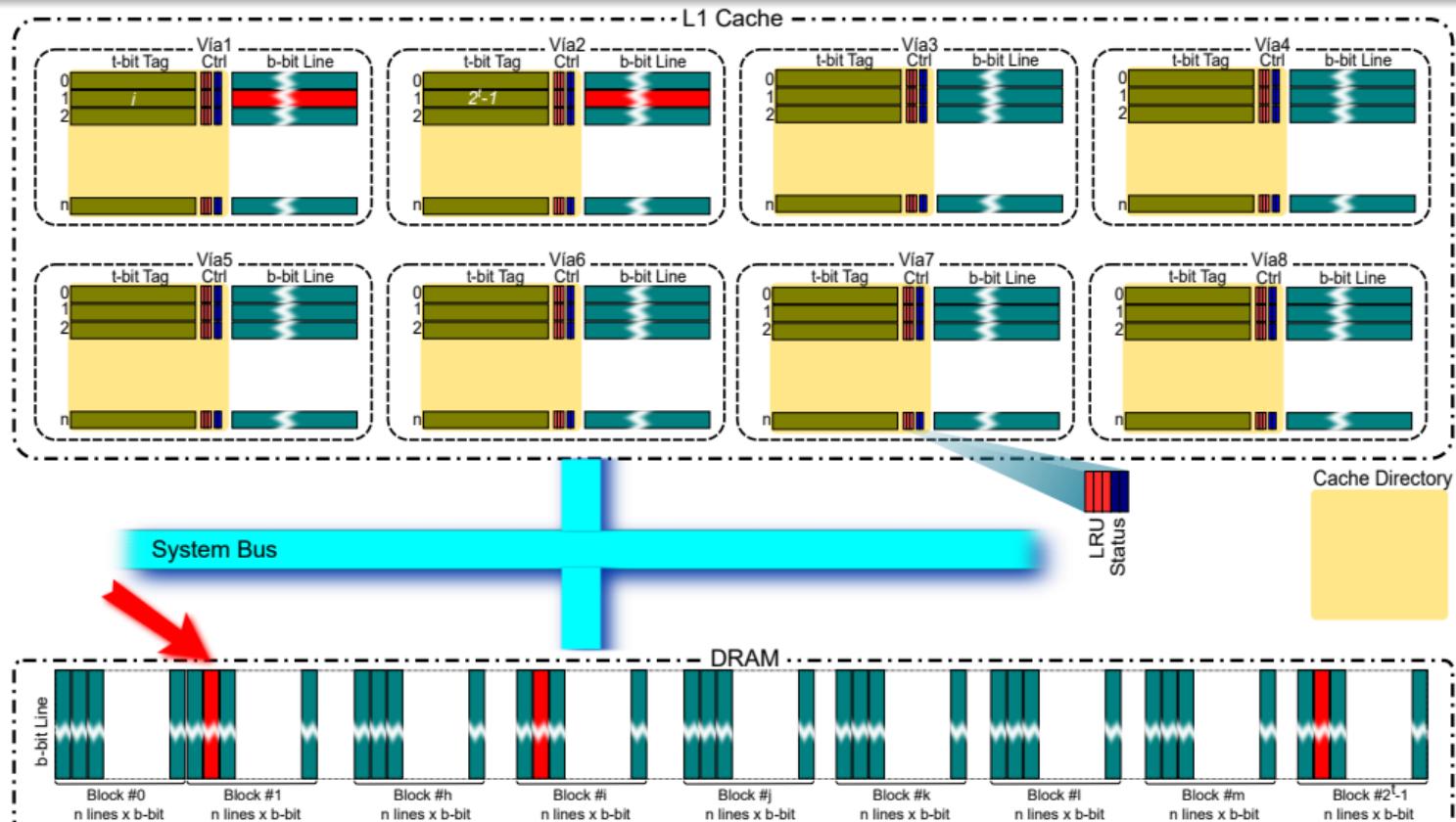
# Line Placement: Cache Set-Asociativo de 8 Vías



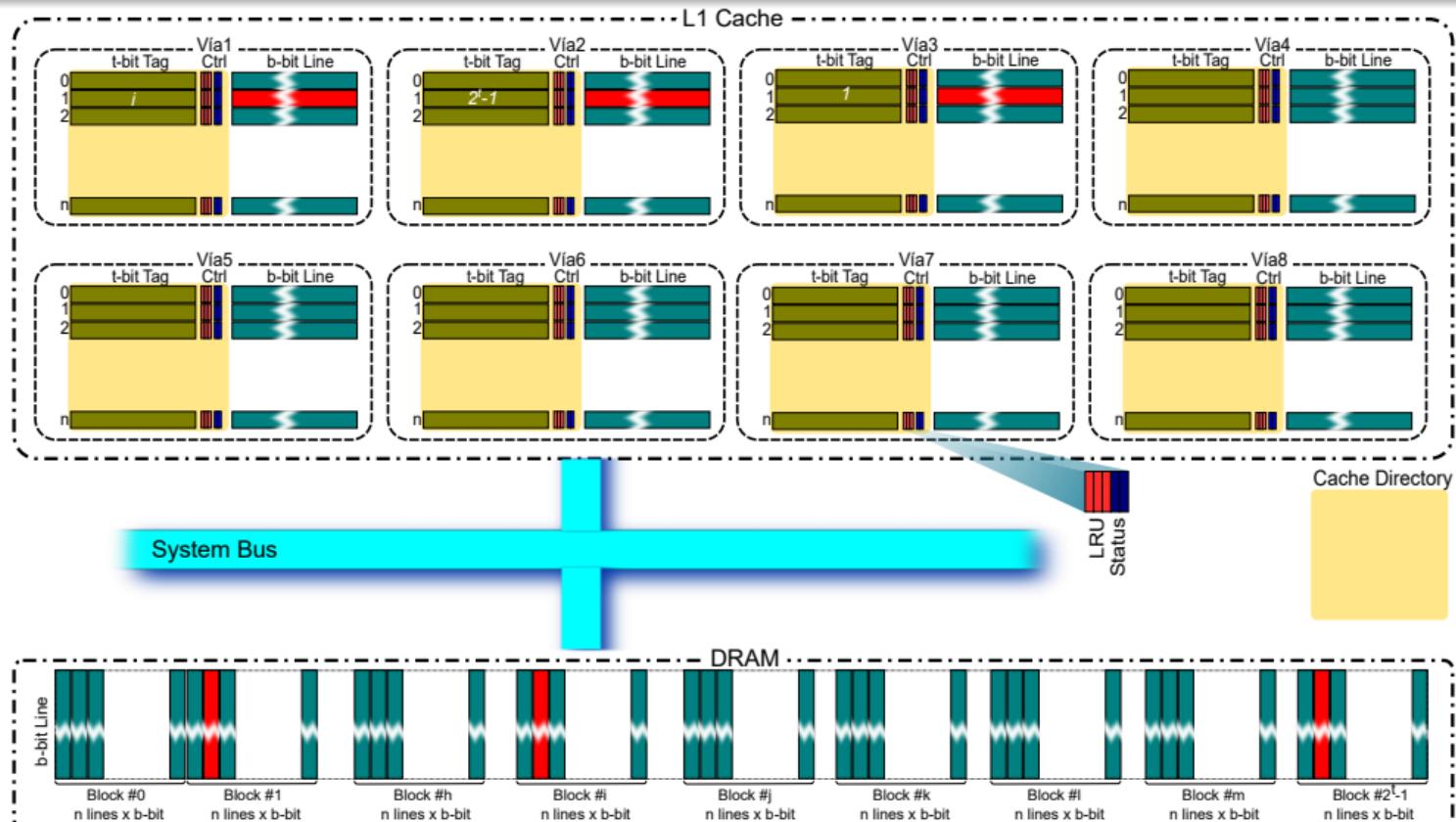
# Line Placement: Cache Set-Asociativo de 8 Vías



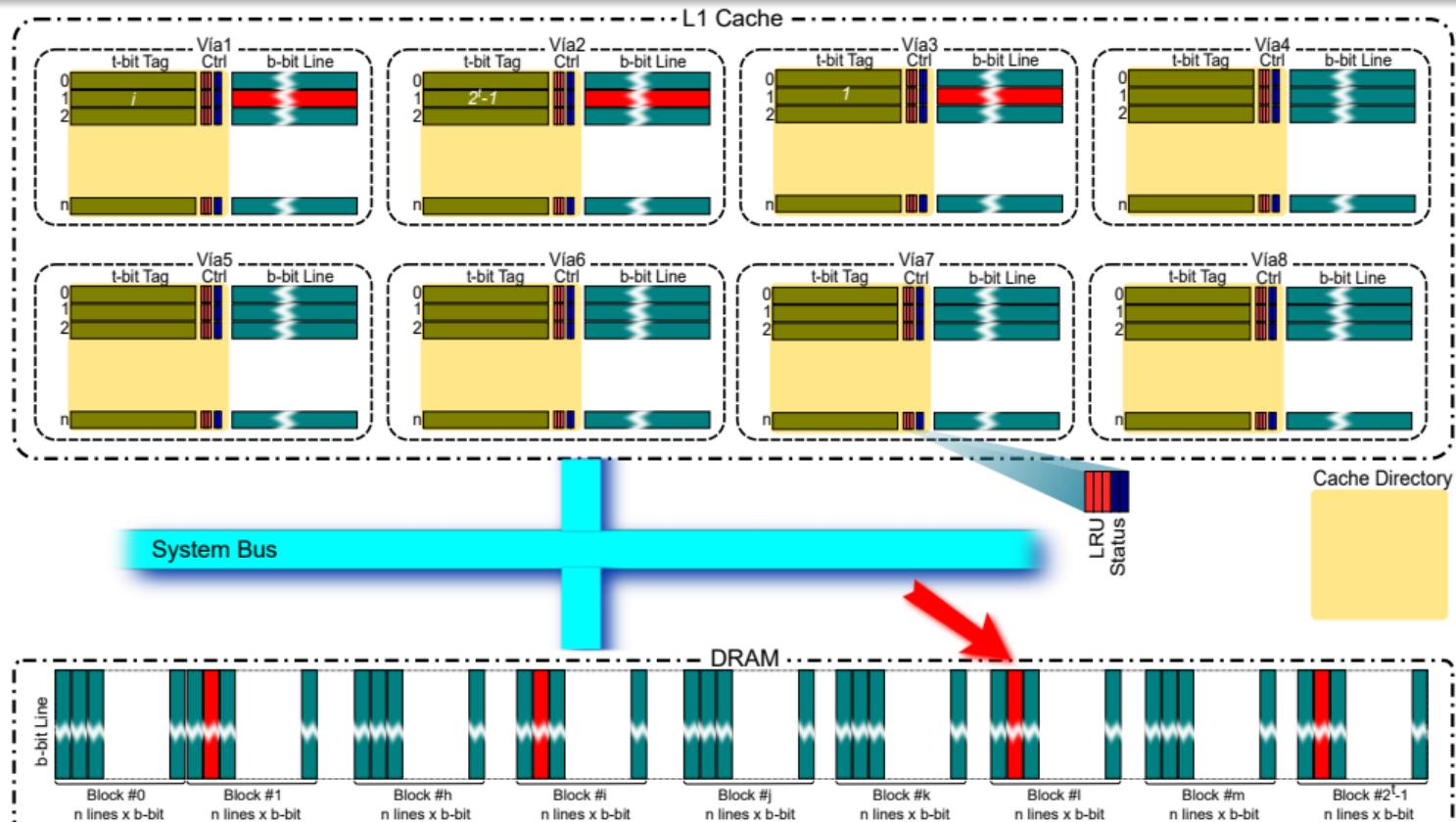
# Line Placement: Cache Set-Asociativo de 8 Vías



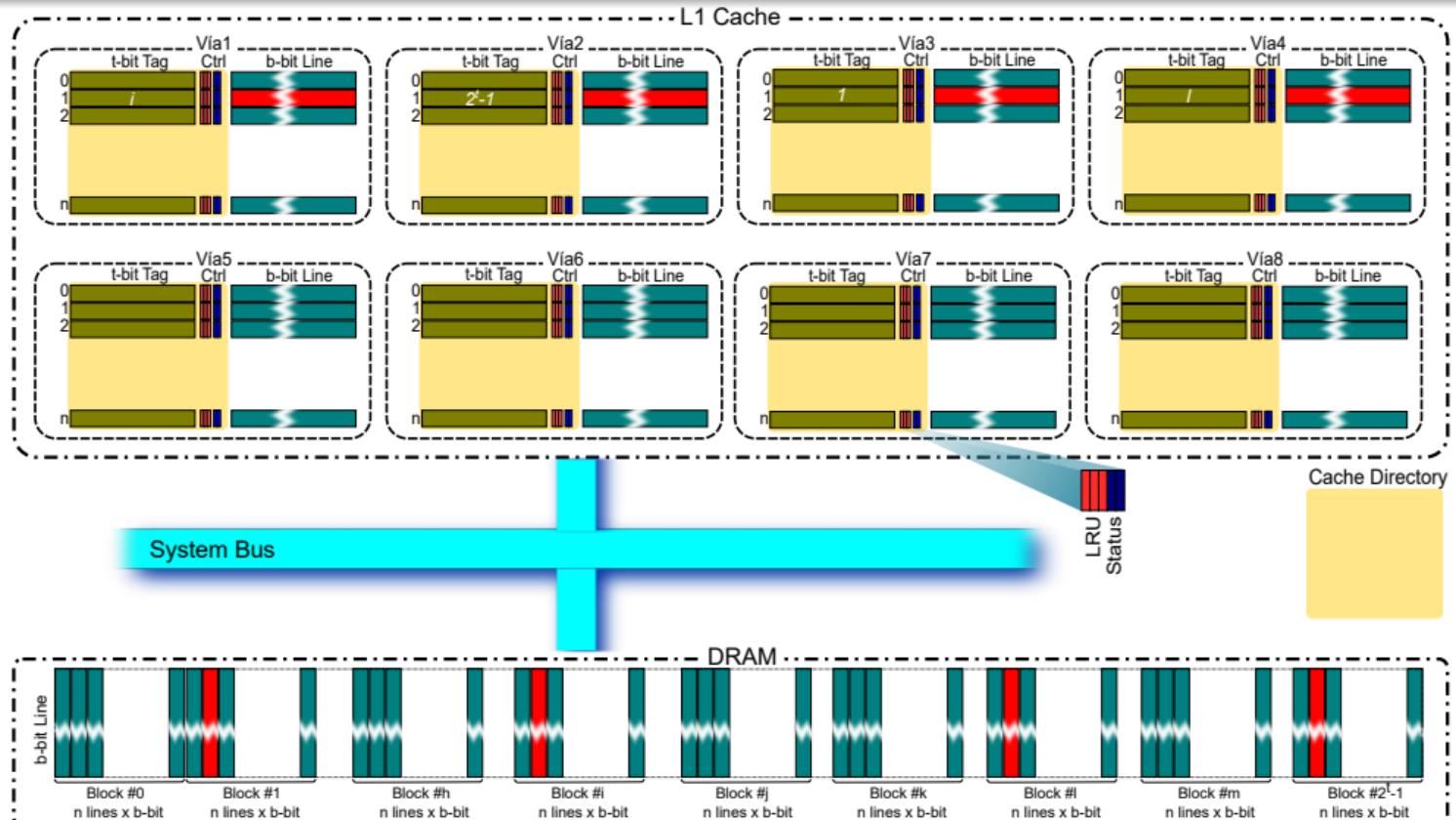
# Line Placement: Cache Set-Asociativo de 8 Vías



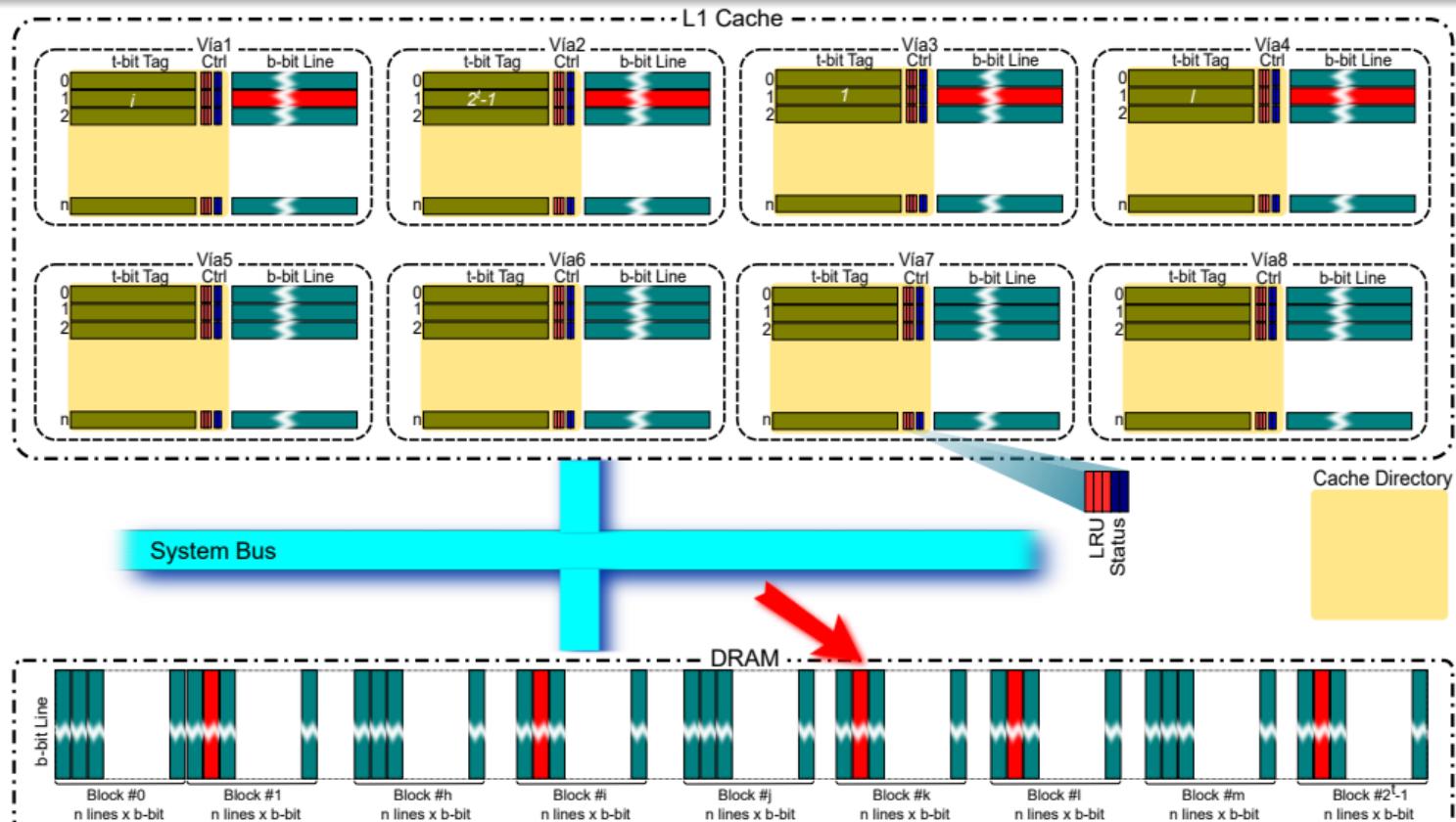
# Line Placement: Cache Set-Asociativo de 8 Vías



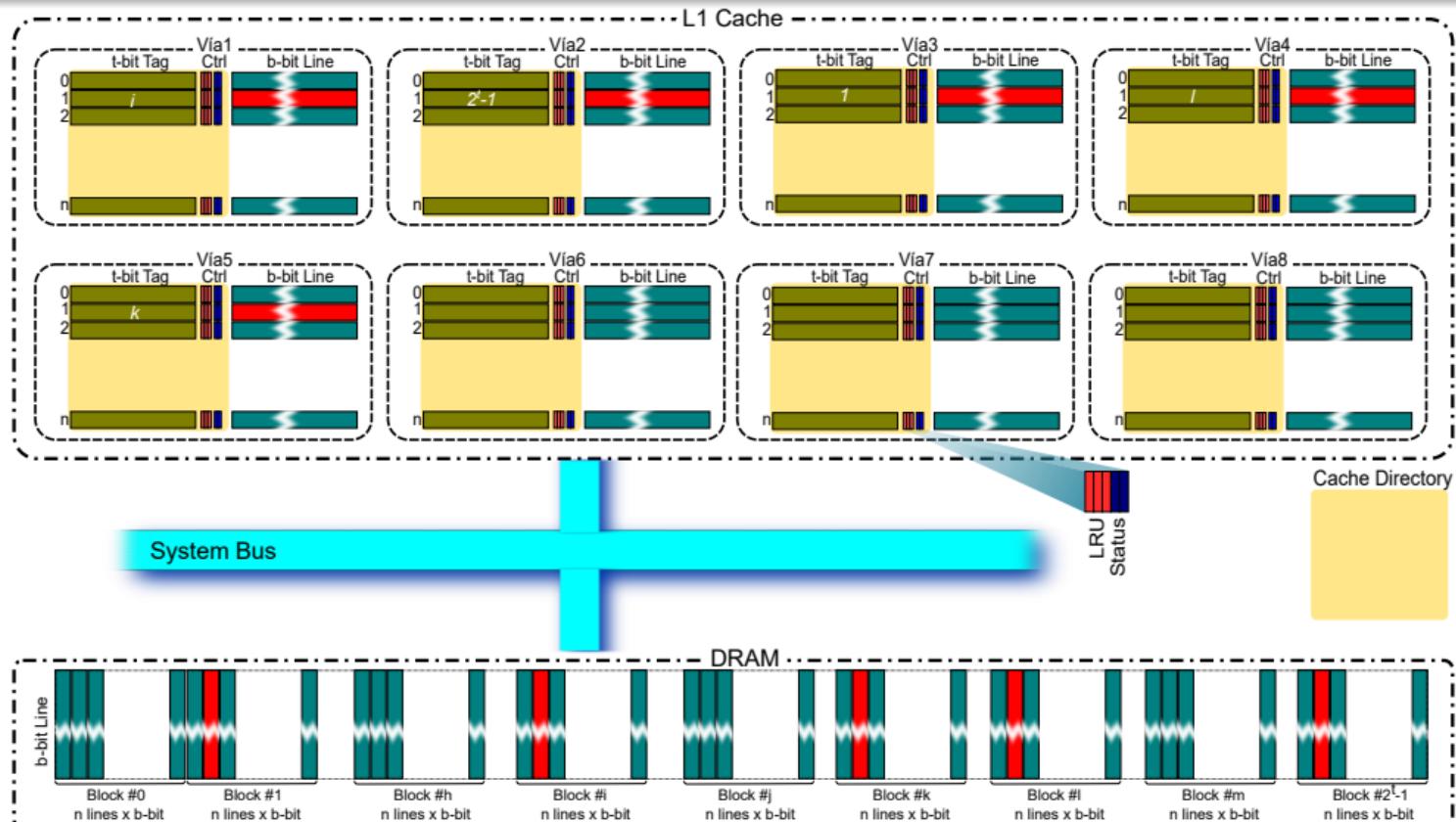
# Line Placement: Cache Set-Asociativo de 8 Vías



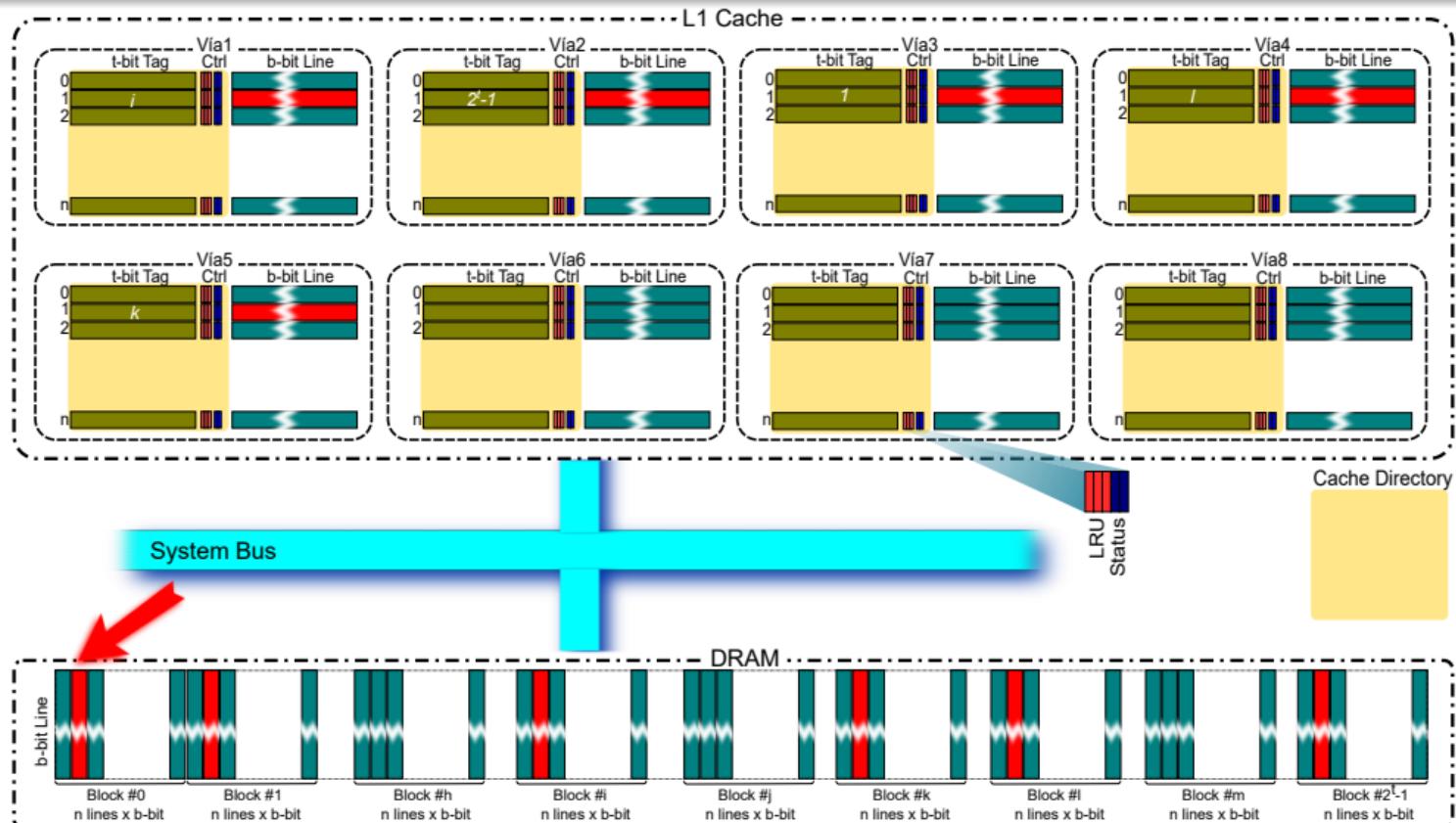
# Line Placement: Cache Set-Asociativo de 8 Vías



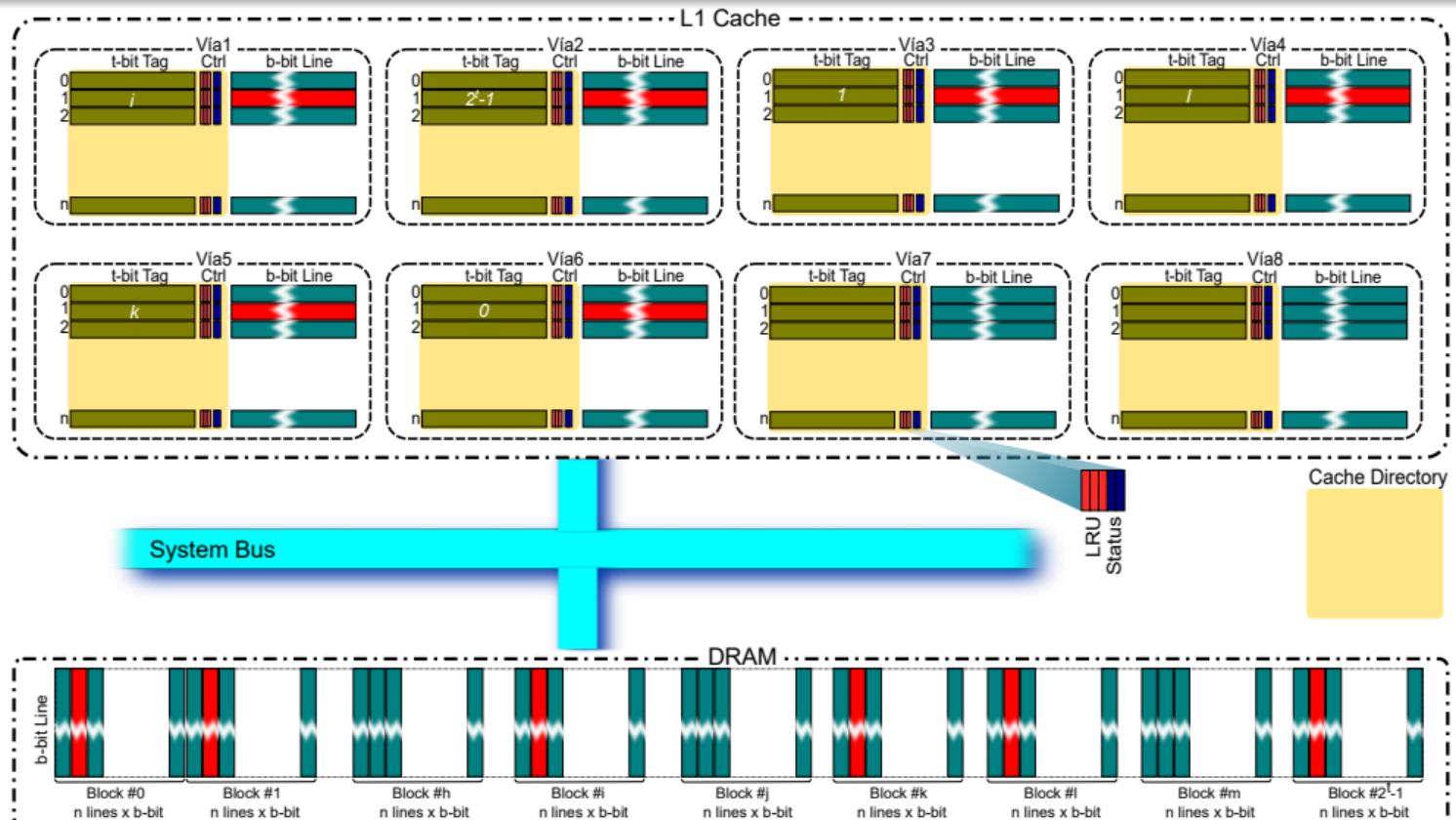
# Line Placement: Cache Set-Asociativo de 8 Vías



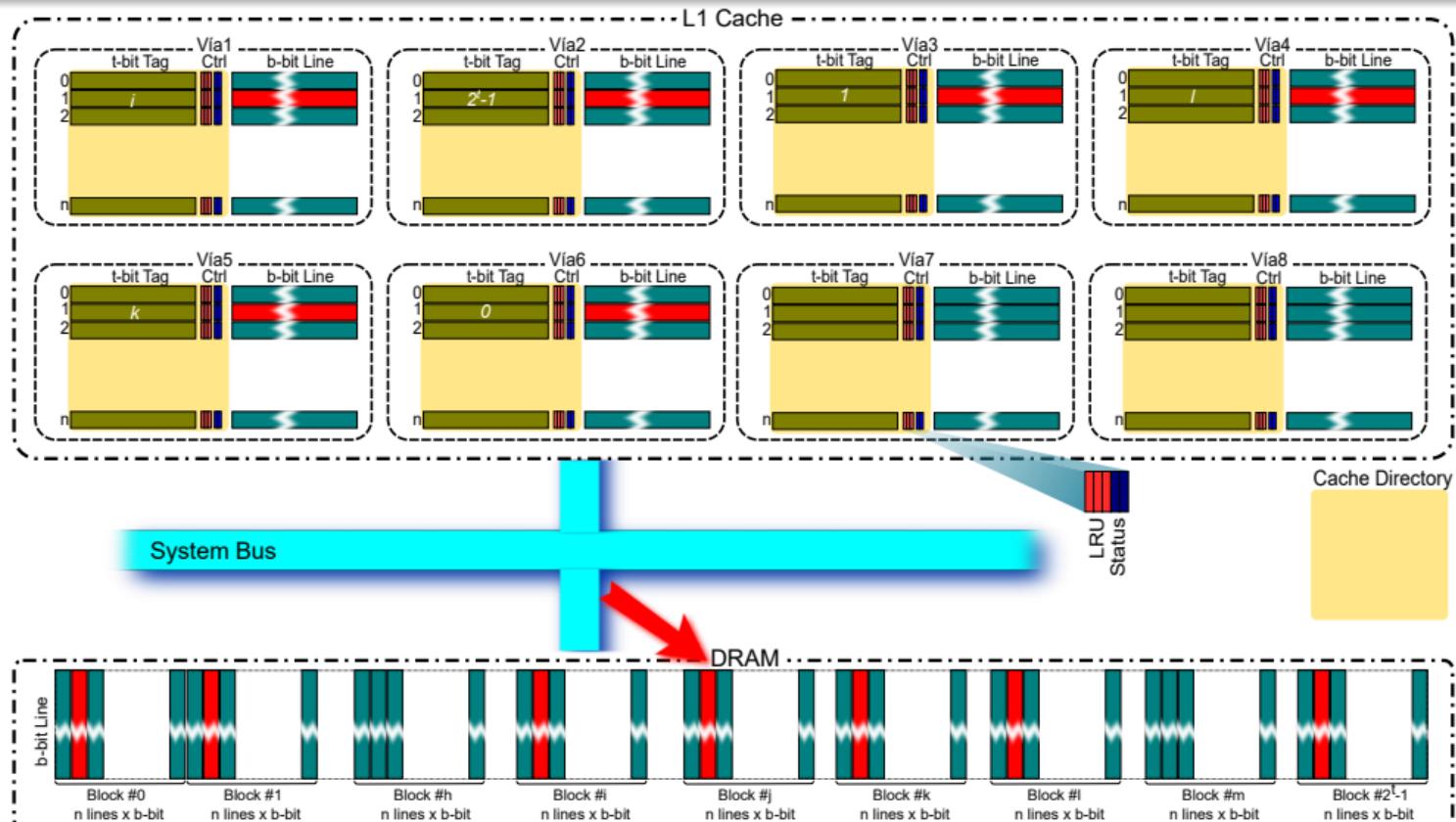
# Line Placement: Cache Set-Asociativo de 8 Vías



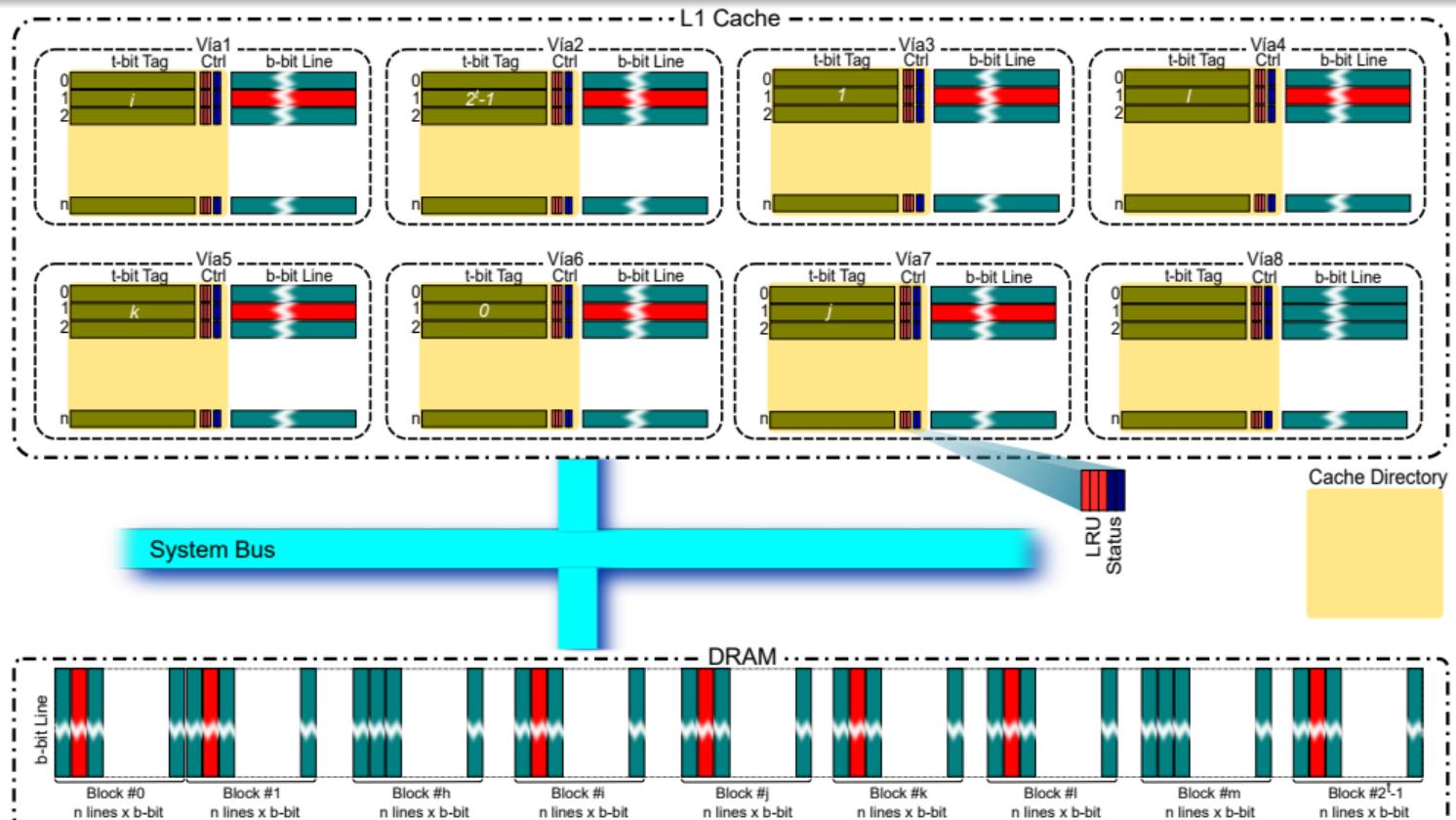
# Line Placement: Cache Set-Asociativo de 8 Vías



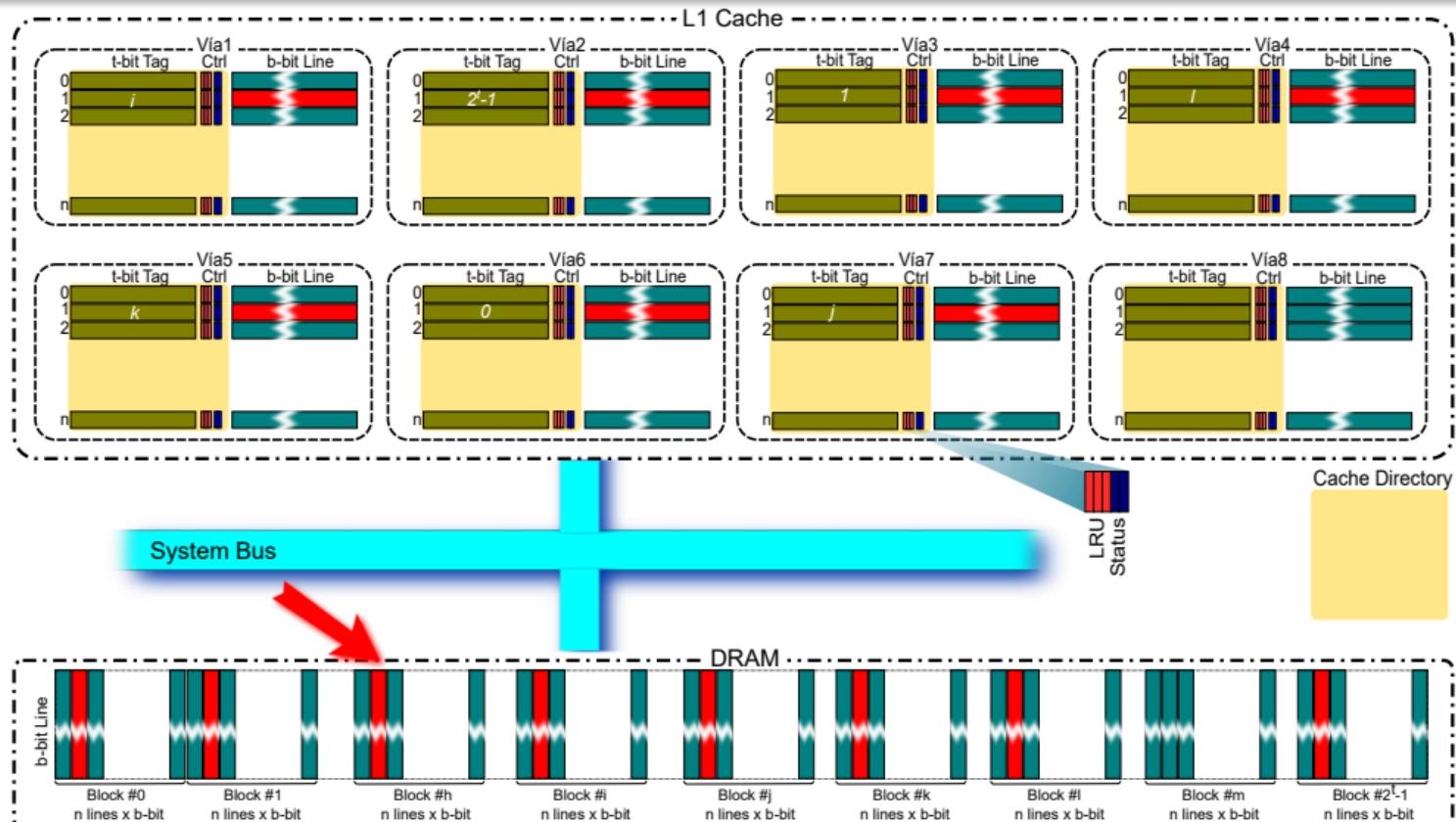
# Line Placement: Cache Set-Asociativo de 8 Vías



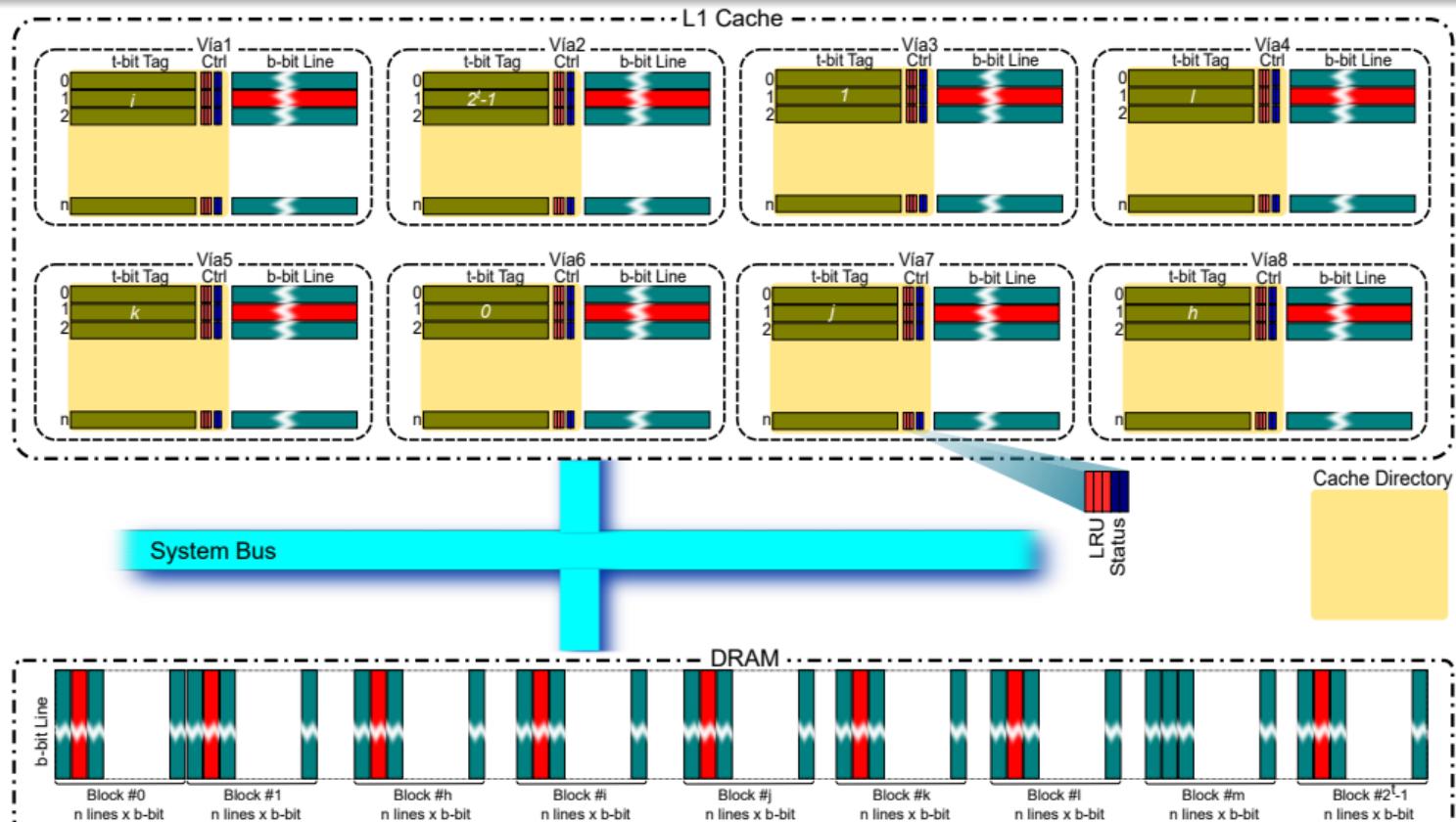
# Line Placement: Cache Set-Asociativo de 8 Vías



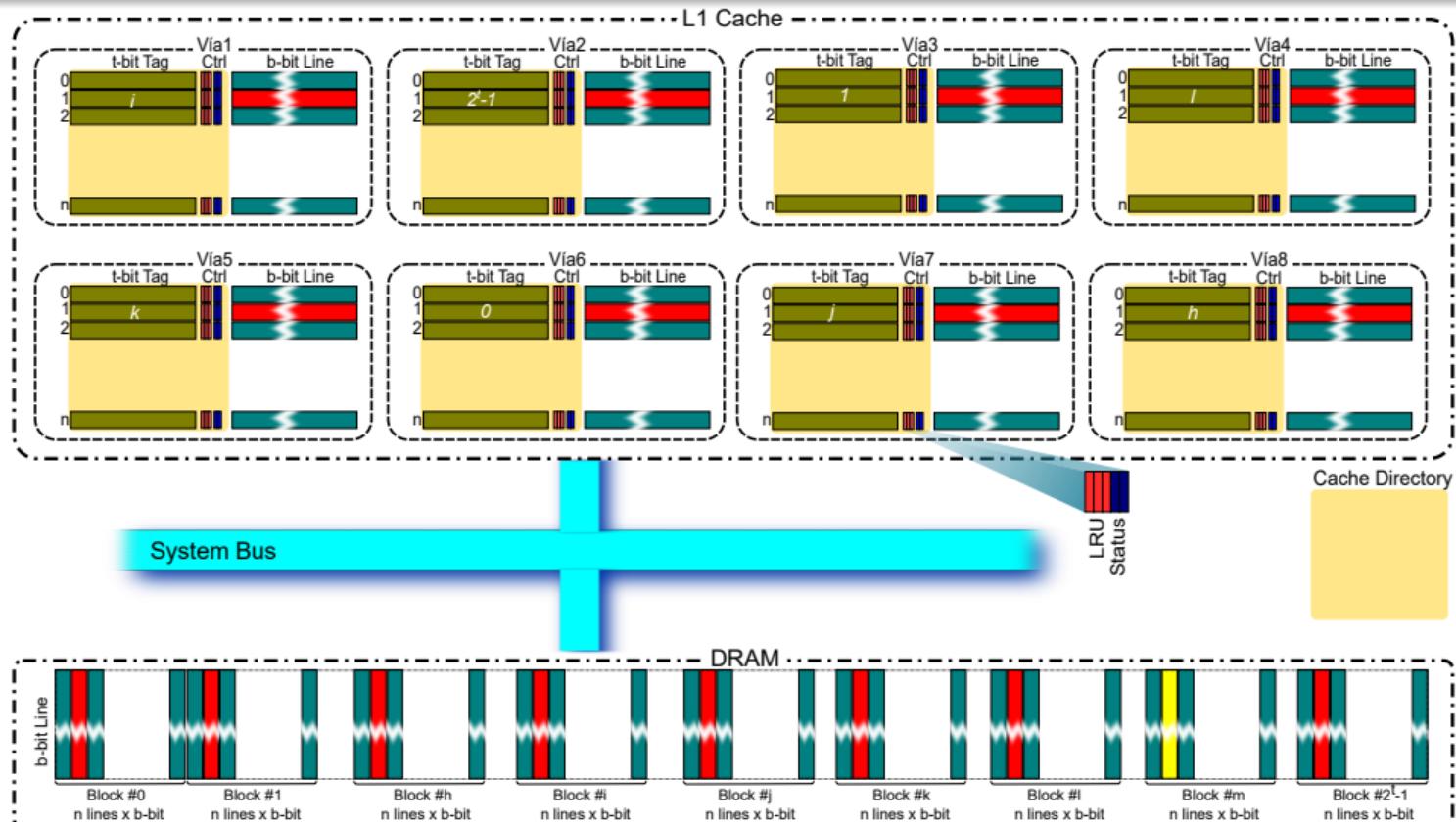
# Line Placement: Cache Set-Asociativo de 8 Vías



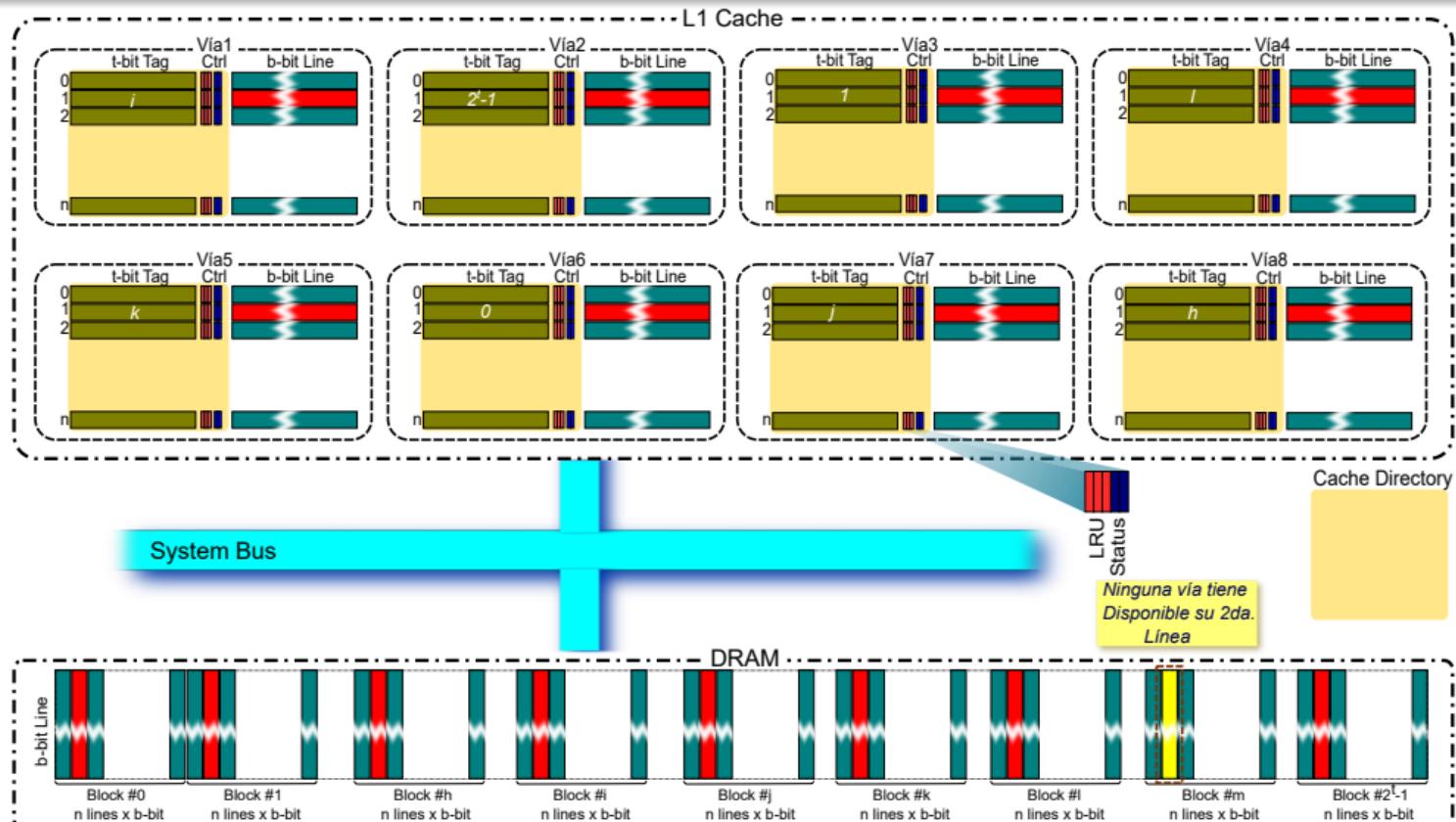
# Line Placement: Cache Set-Asociativo de 8 Vías



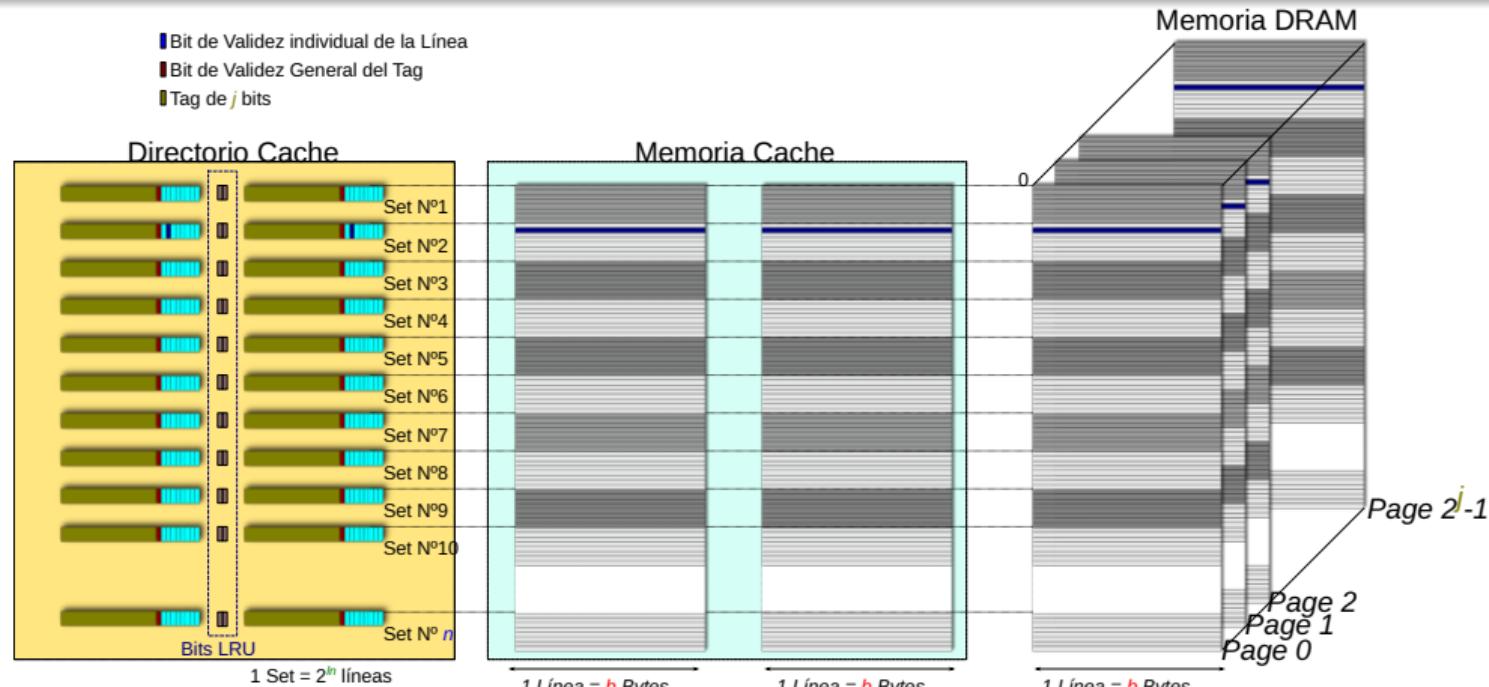
# Line Placement: Cache Set-Asociativo de 8 Vías



# Line Placement: Cache Set-Asociativo de 8 Vías



# Line Placement: Cache Asociativo de 2 Vías



Los primeros sistemas Set-Asociativos fueron de dos vías. Por entonces los buses eran muy angostos y por lo tanto cada set tenía un bloque de varias líneas (típicamente 8 líneas por set). El ejemplo es el primer cache de Intel para el procesador 80386 DX. El criterio sigue siendo el mismo.

# Line Identification

- El Controlador Cache lee la dirección física que la CPU pone en su Address Bus.

# Line Identification

- El Controlador Cache lee la dirección física que la CPU pone en su Address Bus.
- Los bits mas significativos de la dirección física indican el **Tag**.

# Line Identification

- El Controlador Cache lee la dirección física que la CPU pone en su Address Bus.
- Los bits mas significativos de la dirección física indican el **Tag**.
- En el Directorio Cache se guarda un **Tag** asociado a la línea.

# Line Identification

- El Controlador Cache lee la dirección física que la CPU pone en su Address Bus.
- Los bits mas significativos de la dirección física indican el **Tag**.
- En el Directorio Cache se guarda un **Tag** asociado a la línea.
- Ese **Tag** corresponde al número de página que contiene la línea presente en alguna de las vías del Cache.

# Line Identification

- El Controlador Cache lee la dirección física que la CPU pone en su Address Bus.
- Los bits mas significativos de la dirección física indican el **Tag**.
- En el Directorio Cache se guarda un **Tag** asociado a la línea.
- Ese **Tag** corresponde al número de página que contiene la línea presente en alguna de las vías del Cache.
- El Controlador debe determinar si el **Tag** está asociado a alguna de las líneas del set.

# Line Identification

- El Controlador Cache lee la dirección física que la CPU pone en su Address Bus.
- Los bits mas significativos de la dirección física indican el **Tag**.
- En el Directorio Cache se guarda un **Tag** asociado a la línea.
- Ese **Tag** corresponde al número de página que contiene la línea presente en alguna de las vías del Cache.
- El Controlador debe determinar si el **Tag** está asociado a alguna de las líneas del set.
- Los bits menos significativos de la Dirección Física, indican el offset del dato realmente direccionado por la CPU dentro de la línea.

# Line Identification

- El Controlador Cache lee la dirección física que la CPU pone en su Address Bus.
- Los bits mas significativos de la dirección física indican el **Tag**.
- En el Directorio Cache se guarda un **Tag** asociado a la línea.
- Ese **Tag** corresponde al número de página que contiene la línea presente en alguna de las vías del Cache.
- El Controlador debe determinar si el **Tag** está asociado a alguna de las líneas del set.
- Los bits menos significativos de la Dirección Física, indican el offset del dato realmente direccionado por la CPU dentro de la línea.
- Los bits intermedios indican el índice a la línea dentro del cache (o de la página de memoria principal).

# Line Identification

- El Controlador Cache lee la dirección física que la CPU pone en su Address Bus.
- Los bits mas significativos de la dirección física indican el **Tag**.
- En el Directorio Cache se guarda un **Tag** asociado a la línea.
- Ese **Tag** corresponde al número de página que contiene la línea presente en alguna de las vías del Cache.
- El Controlador debe determinar si el **Tag** está asociado a alguna de las líneas del set.
- Los bits menos significativos de la Dirección Física, indican el offset del dato realmente direccionado por la CPU dentro de la línea.
- Los bits intermedios indican el índice a la línea dentro del cache (o de la página de memoria principal).

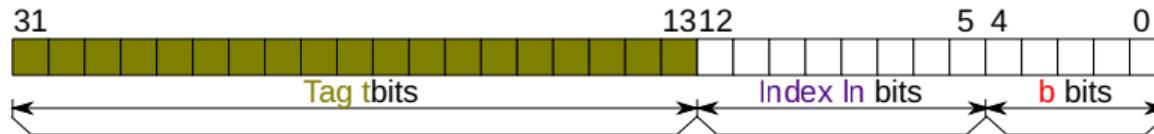
n-1

0

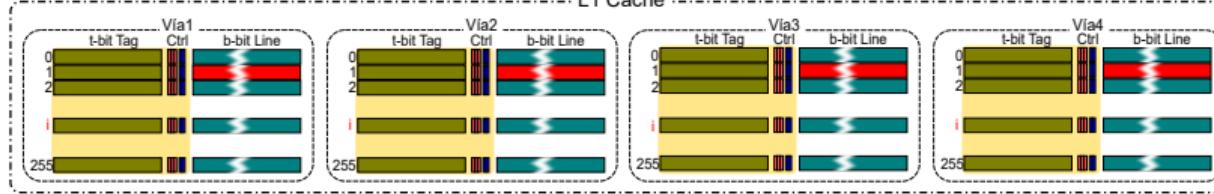
Line Address		Line Index	Line Offset
Tag			

# Line Identification

Dirección Física



L1 Cache



A partir de la dirección física obtenida de la CPU, el Controlador Cache obtiene el **Tag**, y el **Index**, de  $t$  bit y  $ln$  bit respectivamente.

$$t = 19 \text{ bits}$$

$$ln = 8 \text{ bits}$$

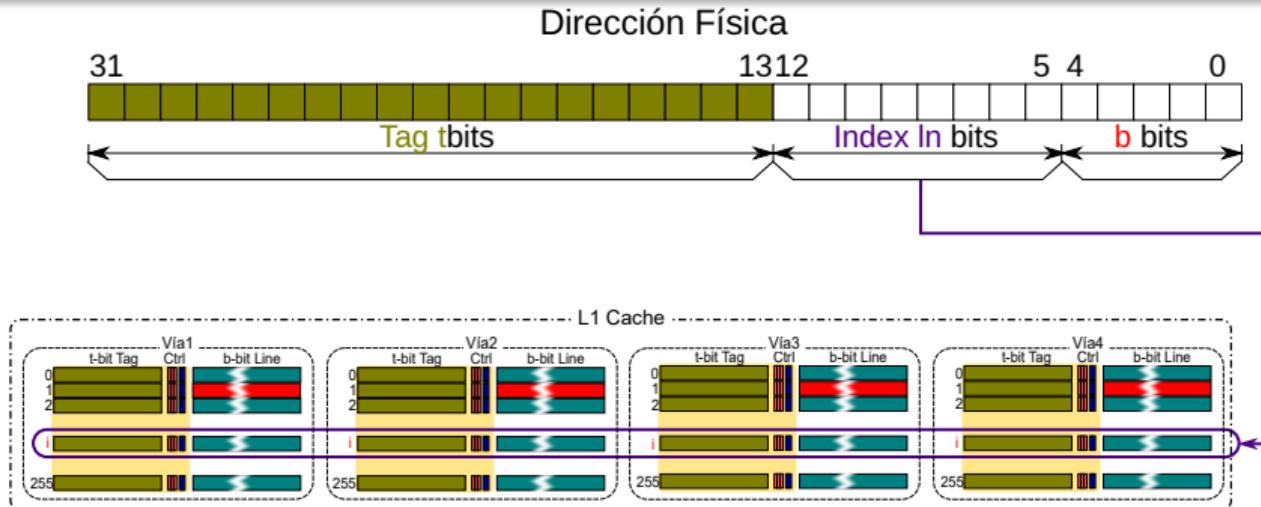
$$b = 5 \text{ bits}$$

$$\text{line size} = 32 \text{ B}$$

$$\text{Cache size} = 32 \text{ KiB}$$

$$\text{Address Space} = 4 \text{ GiB}$$

# Line Identification



Los bits de índice señalan el set. En este caso al ser un Cache Set-Asociativo de 4 vías el set N°  $i$  se componen de las 4  $i$ -ésimas líneas de cada vía.

En este ejemplo **Index**, contiene el valor  $i$

$$t = 19 \text{ bits}$$

$$ln = 8 \text{ bits}$$

$$b = 5 \text{ bits}$$

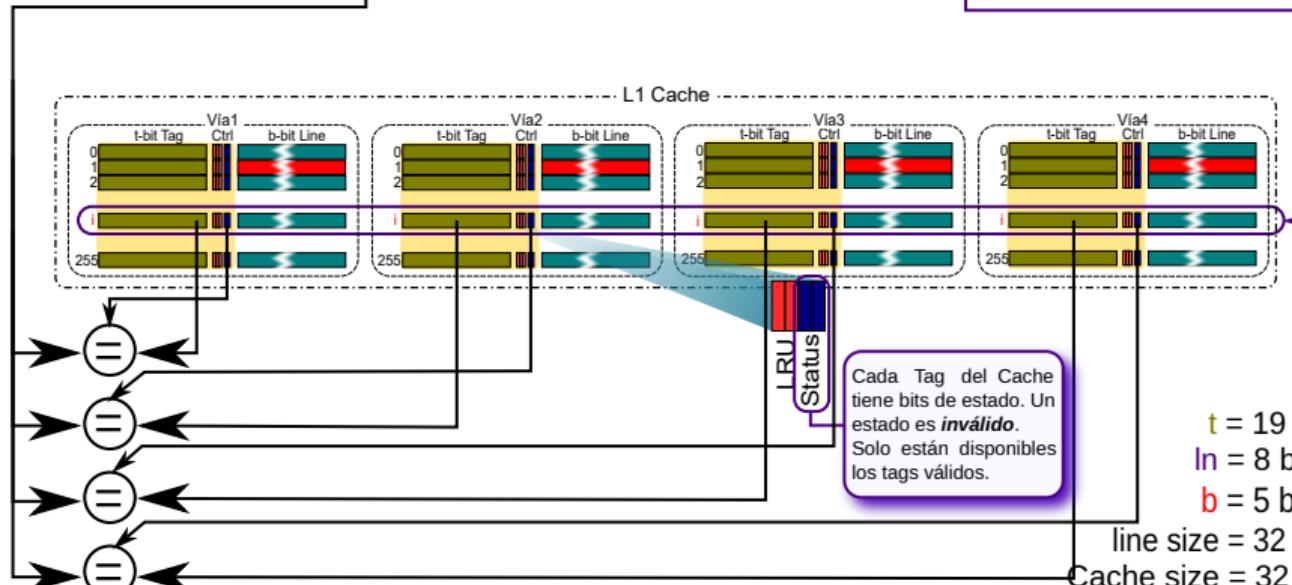
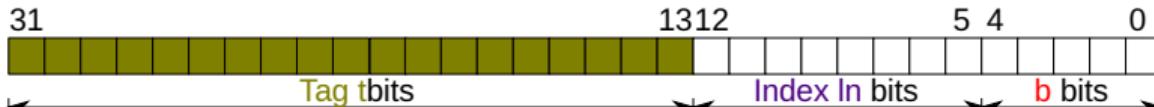
$$\text{line size} = 32 \text{ B}$$

$$\text{Cache size} = 32 \text{ KiB}$$

$$\text{Address Space} = 4 \text{ GiB}$$

# Line Identification

Dirección Física



Cada Tag del Cache tiene bits de estado. Un estado es **inválido**. Solo están disponibles los tags válidos.

Compara en paralelo el **Tag** con cada **Tag** válido del i-ésimo set de 4 líneas . Address Space = 4 GiB

# Line Replacement

- Por lo general las caches son del tipo Set-asociativas. La discusión en la actualidad es cuantas vías.

# Line Replacement

- Por lo general las caches son del tipo Set-asociativas. La discusión en la actualidad es cuantas vías.
- En este punto, la cuestión a resolver es cual de las líneas de las n vías desalojar cuando se genera un **Conflict Miss**

# Line Replacement

- Por lo general las caches son del tipo Set-asociativas. La discusión en la actualidad es cuantas vías.
- En este punto, la cuestión a resolver es cual de las líneas de las n vías desalojar cuando se genera un **Conflict Miss**
- Surgen así tres criterios posibles:

# Line Replacement

- Por lo general las caches son del tipo Set-asociativas. La discusión en la actualidad es cuantas vías.
- En este punto, la cuestión a resolver es cual de las líneas de las n vías desalojar cuando se genera un **Conflict Miss**
- Surgen así tres criterios posibles:
  - **Random:** Reparte la asignación de manera uniforme, seleccionando de manera aleatoria (aunque las implementaciones de hardware suelen ser seudo aleatorias) al candidato a ser desalojado.

# Line Replacement

- Por lo general las caches son del tipo Set-asociativas. La discusión en la actualidad es cuantas vías.
- En este punto, la cuestión a resolver es cual de las líneas de las n vías desalojar cuando se genera un **Conflict Miss**
- Surgen así tres criterios posibles:

**Random:** Reparte la asignación de manera uniforme, seleccionando de manera aleatoria (aunque las implementaciones de hardware suelen ser seudo aleatorias) al candidato a ser desalojado.

**LRU:** Least Recently Used. Si de acuerdo con la localidad, es probable reutilizar líneas recientemente accedidas, entonces la mejor candidata para desalojar es la línea usada menos recientemente. Costoso en términos de hardware.

# Line Replacement

- Por lo general las caches son del tipo Set-asociativas. La discusión en la actualidad es cuantas vías.
- En este punto, la cuestión a resolver es cual de las líneas de las n vías desalojar cuando se genera un **Conflict Miss**
- Surgen así tres criterios posibles:

**Random:** Reparte la asignación de manera uniforme, seleccionando de manera aleatoria (aunque las implementaciones de hardware suelen ser seudo aleatorias) al candidato a ser desalojado.

**LRU:** Least Recently Used. Si de acuerdo con la localidad, es probable reutilizar líneas recientemente accedidas, entonces la mejor candidata para desalojar es la línea usada menos recientemente. Costoso en términos de hardware.

**FIFO:** First In First Out. Es una simplificación de **LRU**, que se limita a marcar la línea ingresada hace mas tiempo.

# Line Replacement

- Una virtud del Random es que es fácil de construir en el hardware.

# Line Replacement

- Una virtud del Random es que es fácil de construir en el hardware.
- A medida que aumenta la cantidad de líneas, LRU se vuelve cada vez más costoso y se termina aproximando (pseudo LRU).

# Line Replacement

- Una virtud del Random es que es fácil de construir en el hardware.
- A medida que aumenta la cantidad de líneas, LRU se vuelve cada vez más costoso y se termina aproximando (pseudo LRU).
- Se puede aproximar con un campo de bits para cada set del cache. Cada bit corresponde a una vía. Cuando se accede a un set, se activa el bit correspondiente a la vía que contiene la línea deseada.

# Line Replacement

- Una virtud del Random es que es fácil de construir en el hardware.
- A medida que aumenta la cantidad de líneas, LRU se vuelve cada vez más costoso y se termina aproximando (pseudo LRU).
- Se puede aproximar con un campo de bits para cada set del cache. Cada bit corresponde a una vía. Cuando se accede a un set, se activa el bit correspondiente a la vía que contiene la línea deseada.
- Si como consecuencia de setear un bit todos los bits de un set quedan activados, se resetean con la excepción del bit activado más recientemente.

# Line Replacement

- Una virtud del Random es que es fácil de construir en el hardware.
- A medida que aumenta la cantidad de líneas, LRU se vuelve cada vez más costoso y se termina aproximando (pseudo LRU).
- Se puede aproximar con un campo de bits para cada set del cache. Cada bit corresponde a una vía. Cuando se accede a un set, se activa el bit correspondiente a la vía que contiene la línea deseada.
- Si como consecuencia de setear un bit todos los bits de un set quedan activados, se resetean con la excepción del bit activado más recientemente.
- Cuando se debe reemplazar una línea, el procesador elige la de la vía cuyo bit está reseteado. Si hay más de una opción disponible... selecciona random.

# Line Replacement

- Una virtud del Random es que es fácil de construir en el hardware.
- A medida que aumenta la cantidad de líneas, LRU se vuelve cada vez más costoso y se termina aproximando (pseudo LRU).
- Se puede aproximar con un campo de bits para cada set del cache. Cada bit corresponde a una vía. Cuando se accede a un set, se activa el bit correspondiente a la vía que contiene la línea deseada.
- Si como consecuencia de setear un bit todos los bits de un set quedan activados, se resetean con la excepción del bit activado más recientemente.
- Cuando se debe reemplazar una línea, el procesador elige la de la vía cuyo bit está reseteado. Si hay más de una opción disponible... selecciona random.
- Esto se aproxima a LRU, porque no se habrá accedido a la línea que se reemplaza desde la última vez que se accedió a todas las líneas del set.

# Line Replacement

Size	Two-Way			Four-Way			Eight-Way		
	LRU	Random	FIFO	LRU	Random	FIFO	LRU	Random	FIFO
16 KiB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KiB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KiB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

# Line Replacement

Size	Two-Way			Four-Way			Eight-Way		
	LRU	Random	FIFO	LRU	Random	FIFO	LRU	Random	FIFO
16 KiB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KiB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KiB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

- En el libro “Computer Architecture a Quantitative approach”, Henessy y Patterson muestran esta tabla de Cache Misses cada 1000 instrucciones que comparan reemplazos LRU, Random y FIFO para varios tamaños y asociatividades.

# Line Replacement

Size	Two-Way			Four-Way			Eight-Way		
	LRU	Random	FIFO	LRU	Random	FIFO	LRU	Random	FIFO
16 KiB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KiB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KiB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

- En el libro “Computer Architecture a Quantitative approach”, Hennessy y Patterson muestran esta tabla de Cache Misses cada 1000 instrucciones que comparan reemplazos LRU, Random y FIFO para varios tamaños y asociatividades.
- Los valores corresponden a un procesador Alpha con tamaño de línea de 64 B.

# Line Replacement

Size	Two-Way			Four-Way			Eight-Way		
	LRU	Random	FIFO	LRU	Random	FIFO	LRU	Random	FIFO
16 KiB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KiB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KiB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

- En el libro “Computer Architecture a Quantitative approach”, Hennessy y Patterson muestran esta tabla de Cache Misses cada 1000 instrucciones que comparan reemplazos LRU, Random y FIFO para varios tamaños y asociatividades.
- Los valores corresponden a un procesador Alpha con tamaño de línea de 64 B.
- Se utilizaron 10 puntos de referencia SPEC2000. Cinco son de SPECint2000 (gap, gcc, gzip, mcf y perl) y cinco son de SPECfp2000 (applu, art, equake, lucas y swim).

# Line Replacement

Size	Two-Way			Four-Way			Eight-Way		
	LRU	Random	FIFO	LRU	Random	FIFO	LRU	Random	FIFO
16 KiB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KiB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KiB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

- En el libro “Computer Architecture a Quantitative approach”, Hennessy y Patterson muestran esta tabla de Cache Misses cada 1000 instrucciones que comparan reemplazos LRU, Random y FIFO para varios tamaños y asociatividades.
- Los valores corresponden a un procesador Alpha con tamaño de línea de 64 B.
- Se utilizaron 10 puntos de referencia SPEC2000. Cinco son de SPECint2000 (gap, gcc, gzip, mcf y perl) y cinco son de SPECfp2000 (applu, art, equake, lucas y swim).
- Los resultados permiten ver que la diferencia entre LRU y Random tiende a bajar cuando aumenta el tamaño del cache.

# Line Replacement

Size	Two-Way			Four-Way			Eight-Way		
	LRU	Random	FIFO	LRU	Random	FIFO	LRU	Random	FIFO
16 KiB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KiB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KiB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

- En el libro “Computer Architecture a Quantitative approach”, Hennessy y Patterson muestran esta tabla de Cache Misses cada 1000 instrucciones que comparan reemplazos LRU, Random y FIFO para varios tamaños y asociatividades.
- Los valores corresponden a un procesador Alpha con tamaño de línea de 64 B.
- Se utilizaron 10 puntos de referencia SPEC2000. Cinco son de SPECint2000 (gap, gcc, gzip, mcf y perl) y cinco son de SPECfp2000 (applu, art, equake, lucas y swim).
- Los resultados permiten ver que la diferencia entre LRU y Random tiende a bajar cuando aumenta el tamaño del cache.
- LRU se impone mas claramente cuando se utilizan caches más pequeños.

# Line Replacement

Size	Two-Way			Four-Way			Eight-Way		
	LRU	Random	FIFO	LRU	Random	FIFO	LRU	Random	FIFO
16 KiB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KiB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KiB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

- En el libro “Computer Architecture a Quantitative approach”, Hennessy y Patterson muestran esta tabla de Cache Misses cada 1000 instrucciones que comparan reemplazos LRU, Random y FIFO para varios tamaños y asociatividades.
- Los valores corresponden a un procesador Alpha con tamaño de línea de 64 B.
- Se utilizaron 10 puntos de referencia SPEC2000. Cinco son de SPECint2000 (gap, gcc, gzip, mcf y perl) y cinco son de SPECfp2000 (applu, art, equake, lucas y swim).
- Los resultados permiten ver que la diferencia entre LRU y Random tiende a bajar cuando aumenta el tamaño del cache.
- LRU se impone mas claramente cuando se utilizan caches más pequeños.
- FIFO generalmente supera a Random en Cache más pequeños.

# Write Policies

- Las mayoría de los accesos a memoria son lecturas.

# Write Policies

- Las mayoría de los accesos a memoria son lecturas.
- Una CPU fetchea instrucciones de manera permanente. El opcode fetch es en si un acceso de lectura.

# Write Policies

- Las mayoría de los accesos a memoria son lecturas.
- Una CPU fetchea instrucciones de manera permanente. El opcode fetch es en si un acceso de lectura.
- De éstas instrucciones, aproximadamente entre el 25 % y el 30 % ejecutan lecturas de datos de memoria (Load), y entre el 10 % y el 15 % ejecutan escrituras de datos de memoria (Store).

# Write Policies

- Las mayoría de los accesos a memoria son lecturas.
- Una CPU fetchea instrucciones de manera permanente. El opcode fetch es en si un acceso de lectura.
- De éstas instrucciones, aproximadamente entre el 25 % y el 30 % ejecutan lecturas de datos de memoria (Load), y entre el 10 % y el 15 % ejecutan escrituras de datos de memoria (Store).
- Significa que en el mejor de los casos el porcentaje de escrituras de memoria respecto del total de accesos viene dado por:

$$\frac{\%stores}{\%fetches + \%Stores + \%Loads} = \frac{15}{100+25+15} = 10,7 \%$$

# Write Policies

- Las mayoría de los accesos a memoria son lecturas.
- Una CPU fetchea instrucciones de manera permanente. El opcode fetch es en si un acceso de lectura.
- De éstas instrucciones, aproximadamente entre el 25 % y el 30 % ejecutan lecturas de datos de memoria (Load), y entre el 10 % y el 15 % ejecutan escrituras de datos de memoria (Store).
- Significa que en el mejor de los casos el porcentaje de escrituras de memoria respecto del total de accesos viene dado por:

$$\frac{\%stores}{\%fetches + \%Stores + \%Loads} = \frac{15}{100+25+15} = 10,7 \%$$

- El caso mas frecuente (Lectura) es relativamente sencillo de realizar en el menor tiempo posible.

## Write Policies

- Las mayoría de los accesos a memoria son lecturas.
- Una CPU fetchea instrucciones de manera permanente. El opcode fetch es en si un acceso de lectura.
- De éstas instrucciones, aproximadamente entre el 25 % y el 30 % ejecutan lecturas de datos de memoria (Load), y entre el 10 % y el 15 % ejecutan escrituras de datos de memoria (Store).
- Significa que en el mejor de los casos el porcentaje de escrituras de memoria respecto del total de accesos viene dado por:

$$\frac{\%stores}{\%fetches + \%Stores + \%Loads} = \frac{15}{100+25+15} = 10,7 \%$$

- El caso mas frecuente (Lectura) es relativamente sencillo de realizar en el menor tiempo posible.
- En el mismo clock leemos el dato del cache y verificamos el tag para saber si es un hit. Si lo es, el dato está disponible para ser procesado.

# Write Policies

- No es posible escribir sin antes saber si el acceso es un hit.

# Write Policies

- No es posible escribir sin antes saber si el acceso es un hit.
- Al hecho de no poder paralelizar la escritura con la comprobación del tag, se agregan demoras adicionales

# Write Policies

- No es posible escribir sin antes saber si el acceso es un hit.
- Al hecho de no poder paralelizar la escritura con la comprobación del tag, se agregan demoras adicionales
- Las escrituras deben modificar únicamente la parte de la línea que corresponde. El procesador establece de manera precisa a partir de qué dirección escribir, y el tamaño del dato a escribir, que puede variar entre 1 y 8 B.

## Write Policies

- No es posible escribir sin antes saber si el acceso es un hit.
- Al hecho de no poder paralelizar la escritura con la comprobación del tag, se agregan demoras adicionales
- Las escrituras deben modificar únicamente la parte de la línea que corresponde. El procesador establece de manera precisa a partir de qué dirección escribir, y el tamaño del dato a escribir, que puede variar entre 1 y 8 B.
- Las lecturas en contraste llevan a la CPU la línea entera, sin demoras.

# Write Policies

- No es posible escribir sin antes saber si el acceso es un hit.
- Al hecho de no poder paralelizar la escritura con la comprobación del tag, se agregan demoras adicionales
- Las escrituras deben modificar únicamente la parte de la línea que corresponde. El procesador establece de manera precisa a partir de qué dirección escribir, y el tamaño del dato a escribir, que puede variar entre 1 y 8 B.
- Las lecturas en contraste llevan a la CPU la línea entera, sin demoras.
- Las políticas de escritura a adoptar definen el diseño de un Sistema Cache.

# Write Policies

Para escribir en un cache, hay dos opciones bien diferentes:

# Write Policies

Para escribir en un cache, hay dos opciones bien diferentes:

**Write through:** El dato se escribe en el Cache y en el nivel inferior de la jerarquía.

# Write Policies

Para escribir en un cache, hay dos opciones bien diferentes:

**Write through:** El dato se escribe en el Cache y en el nivel inferior de la jerarquía.

**Write Back:** A.k.a. Copy Back. El dato se escribe solo en el cache, y se actualiza en el resto de la jerarquía solo cuando es desalojado.

# Write Policies

Para escribir en un cache, hay dos opciones bien diferentes:

**Write through:** El dato se escribe en el Cache y en el nivel inferior de la jerarquía.

**Write Back:** A.k.a. Copy Back. El dato se escribe solo en el cache, y se actualiza en el resto de la jerarquía solo cuando es desalojado.

Para minimizar la cantidad de write back's se utiliza por cada línea un bit que indique si está dirty (modificada) o clean (no modificada).

Solo se realiza el write back al momento del desalojo para líneas dirty, ya que son las que se han modificado respecto del/los nivel/es inferior/es.

# Write Policies

Para escribir en un cache, hay dos opciones bien diferentes:

**Write through:** El dato se escribe en el Cache y en el nivel inferior de la jerarquía.

**Write Back:** A.k.a. Copy Back. El dato se escribe solo en el cache, y se actualiza en el resto de la jerarquía solo cuando es desalojado.

Para minimizar la cantidad de write back's se utiliza por cada línea un bit que indique si está dirty (modificada) o clean (no modificada).

Solo se realiza el write back al momento del desalojo para líneas dirty, ya que son las que se han modificado respecto del/los nivel/es inferior/es.

Ambas políticas tienen ventajas y desventajas

# Write Policies

## Ventajas de Write Back

Las escrituras ocurren a la velocidad del cache, es decir, la máxima. Si en una misma línea ocurren múltiples escrituras, solo una escritura se enviará hacia el nivel inferior de la jerarquía. En comparación con Write Through, demandará menos ancho de banda en el Bus. Esto encaja muy bien en sistemas Multiprocesador, en los que el Bus es un recurso compartido. Además minimizar los Write Back implica mínimo consumo de energía, muy atractivo para los sistemas Embedded.

# Write Policies

## Ventajas de Write Back

Las escrituras ocurren a la velocidad del cache, es decir, la máxima. Si en una misma línea ocurren múltiples escrituras, solo una escritura se enviará hacia el nivel inferior de la jerarquía. En comparación con Write Through, demandará menos ancho de banda en el Bus. Esto encaja muy bien en sistemas Multiprocesador, en los que el Bus es un recurso compartido. Además minimizar los Write Back implica mínimo consumo de energía, muy atractivo para los sistemas Embedded.

## Ventajas de Write Through

Es mucho mas simple su implementación. El siguiente nivel jerárquico menor siempre tiene una copia coherente del dato. Esto también favorece a los sistemas Multiprocesador, pero también a los subsistemas de E/S, como los DMA de disco por ejemplo. Se puede encontrar Write Through en el nivel alto de un Cache Multinivel, ya que el tiempo hasta el segundo nivel de Cache es mucho menor que hasta la DRAM.

# Write Policies

- Se llama *write stall* a la espera del procesador una escritura que debe replicarse en niveles inferiores de la memoria.

# Write Policies

- Se llama *write stall* a la espera del procesador una escritura que debe replicarse en niveles inferiores de la memoria.
- Una opción que mejora la penalización de performance debida a *write stall*, es incluir un Write Buffer en el Controlador Cache.

# Write Policies

- Se llama *write stall* a la espera del procesador una escritura que debe replicarse en niveles inferiores de la memoria.
- Una opción que mejora la penalización de performance debida a *write stall*, es incluir un Write Buffer en el Controlador Cache.
- Las escrituras se llevan a cabo en el cache y al mismo tiempo se almacena en el Write Buffer.

# Write Policies

- Se llama *write stall* a la espera del procesador una escritura que debe replicarse en niveles inferiores de la memoria.
- Una opción que mejora la penalización de performance debida a *write stall*, es incluir un Write Buffer en el Controlador Cache.
- Las escrituras se llevan a cabo en el cache y al mismo tiempo se almacena en el Write Buffer.
- En paralelo se actualiza la copia en el nivel inferior de la jerarquía, y el procesador ejecuta otras instrucciones, y lee datos del Cache.

# Write Policies

- Se llama *write stall* a la espera del procesador una escritura que debe replicarse en niveles inferiores de la memoria.
- Una opción que mejora la penalización de performance debida a *write stall*, es incluir un Write Buffer en el Controlador Cache.
- Las escrituras se llevan a cabo en el cache y al mismo tiempo se almacena en el Write Buffer.
- En paralelo se actualiza la copia en el nivel inferior de la jerarquía, y el procesador ejecuta otras instrucciones, y lee datos del Cache.
- No obstante si se genera un Read Miss mientras el controlador cache está actualizando el valor, el Miss penalty es mayor ya que deberá esperar que termine la actualización.

# Write Policies

- Se llama *write stall* a la espera del procesador una escritura que debe replicarse en niveles inferiores de la memoria.
- Una opción que mejora la penalización de performance debida a *write stall*, es incluir un Write Buffer en el Controlador Cache.
- Las escrituras se llevan a cabo en el cache y al mismo tiempo se almacena en el Write Buffer.
- En paralelo se actualiza la copia en el nivel inferior de la jerarquía, y el procesador ejecuta otras instrucciones, y lee datos del Cache.
- No obstante si se genera un Read Miss mientras el controlador cache está actualizando el valor, el Miss penalty es mayor ya que deberá esperar que termine la actualización.
- Por otra parte el Write Buffer puede recibir escrituras a mayor frecuencia que la que le toma llevarlas a cabo, de modo que el Write Buffer no elimina los *Write Stall*. Tan solo reduce su cantidad.

# Write Policies

- Una posibilidad cuando se requiere una escritura, es que el dato a escribir no esté en el cache. Es decir, Write Miss.

# Write Policies

- Una posibilidad cuando se requiere una escritura, es que el dato a escribir no esté en el cache. Es decir, Write Miss.
- Hay dos formas de tratar este problema:

# Write Policies

- Una posibilidad cuando se requiere una escritura, es que el dato a escribir no esté en el cache. Es decir, Write Miss.
- Hay dos formas de tratar este problema:  
**Write Allocate:** Actúa como un Read Miss. Se lee la línea en el cache y luego se escribe.

# Write Policies

- Una posibilidad cuando se requiere una escritura, es que el dato a escribir no esté en el cache. Es decir, Write Miss.
- Hay dos formas de tratar este problema:

Write Allocate: Actúa como un Read Miss. Se lee la línea en el cache y luego se escribe.

No-Write Allocate: La operación no afecta el cache. El dato se escribe en el nivel inferior inmediato. Recién con una lectura se trae el dato al cache (Allocate).

# Write Policies

- Una posibilidad cuando se requiere una escritura, es que el dato a escribir no esté en el cache. Es decir, Write Miss.
- Hay dos formas de tratar este problema:

Write Allocate: Actúa como un Read Miss. Se lee la línea en el cache y luego se escribe.

No-Write Allocate: La operación no afecta el cache. El dato se escribe en el nivel inferior inmediato. Recién con una lectura se trae el dato al cache (Allocate).

- Las direcciones apuntadas por R6 y R8 no están en el cache

```
1      str  r1, [r6]
2      str  r3, [r6]
3      ldr  r0, [r8]
4      str  r2, [r8]
5      str  r9, [r6]
```

# Write Policies

- Una posibilidad cuando se requiere una escritura, es que el dato a escribir no esté en el cache. Es decir, Write Miss.
- Hay dos formas de tratar este problema:

Write Allocate: Actúa como un Read Miss. Se lee la línea en el cache y luego se escribe.

No-Write Allocate: La operación no afecta el cache. El dato se escribe en el nivel inferior inmediato. Recién con una lectura se trae el dato al cache (Allocate).

- Las direcciones apuntadas por R6 y R8 no están en el cache
- No-Write Allocate: Las tres escrituras a [R6] son Miss. La lectura a [R8] es Miss y la escritura Posterior Hit. Total 4 Misses y 1 Hit.

```
1      str  r1, [r6]
2      str  r3, [r6]
3      ldr  r0, [r8]
4      str  r2, [r8]
5      str  r9, [r6]
```

# Write Policies

- Una posibilidad cuando se requiere una escritura, es que el dato a escribir no esté en el cache. Es decir, Write Miss.
- Hay dos formas de tratar este problema:

Write Allocate: Actúa como un Read Miss. Se lee la línea en el cache y luego se escribe.

No-Write Allocate: La operación no afecta el cache. El dato se escribe en el nivel inferior inmediato. Recién con una lectura se trae el dato al cache (Allocate).

```
1      str r1, [r6]
2      str r3, [r6]
3      ldr r0, [r8]
4      str r2, [r8]
5      str r9, [r6]
```

- Las direcciones apuntadas por R6 y R8 no están en el cache
- No-Write Allocate: Las tres escrituras a [R6] son Miss. La lectura a [R8] es Miss y la escritura Posterior Hit. Total 4 Misses y 1 Hit.
- Write Allocate: Solo el 1er. acceso a [R6] y [r8] son Miss. Total 2 Misses 3 Hit.

## Casos Prácticos. AMD Opteron Cache L1

- En 2003 AMD presenta esta línea de procesadores que implementa por primera vez la arquitectura de 64 bit AMD64, que a posteriori Intel debería incluir en su línea x86 luego del fracaso comercial de Itanium.

## Casos Prácticos. AMD Opteron Cache L1

- En 2003 AMD presenta esta línea de procesadores que implementa por primera vez la arquitectura de 64 bit AMD64, que a posteriori Intel debería incluir en su línea x86 luego del fracaso comercial de Itanium.
- El primer procesador de arquitectura denominada *SledgeHammer*, tenía un solo core con Ejecución Fuera de Orden y dos niveles de Cache: 64 KiB, y 256 KiB on chip.

## Casos Prácticos. AMD Opteron Cache L1

- En 2003 AMD presenta esta línea de procesadores que implementa por primera vez la arquitectura de 64 bit AMD64, que a posteriori Intel debería incluir en su línea x86 luego del fracaso comercial de Itanium.
- El primer procesador de arquitectura denominada *SledgeHammer*, tenía un solo core con Ejecución Fuera de Orden y dos niveles de Cache: 64 KiB, y 256 KiB on chip.
- En 64 bit, el procesador trabaja con direcciones virtuales de 48 bit, y envía al Address Bus direcciones físicas de 40 bit (con el tiempo al igual que los Intel, este número crecerá hasta 52 bit actuales)

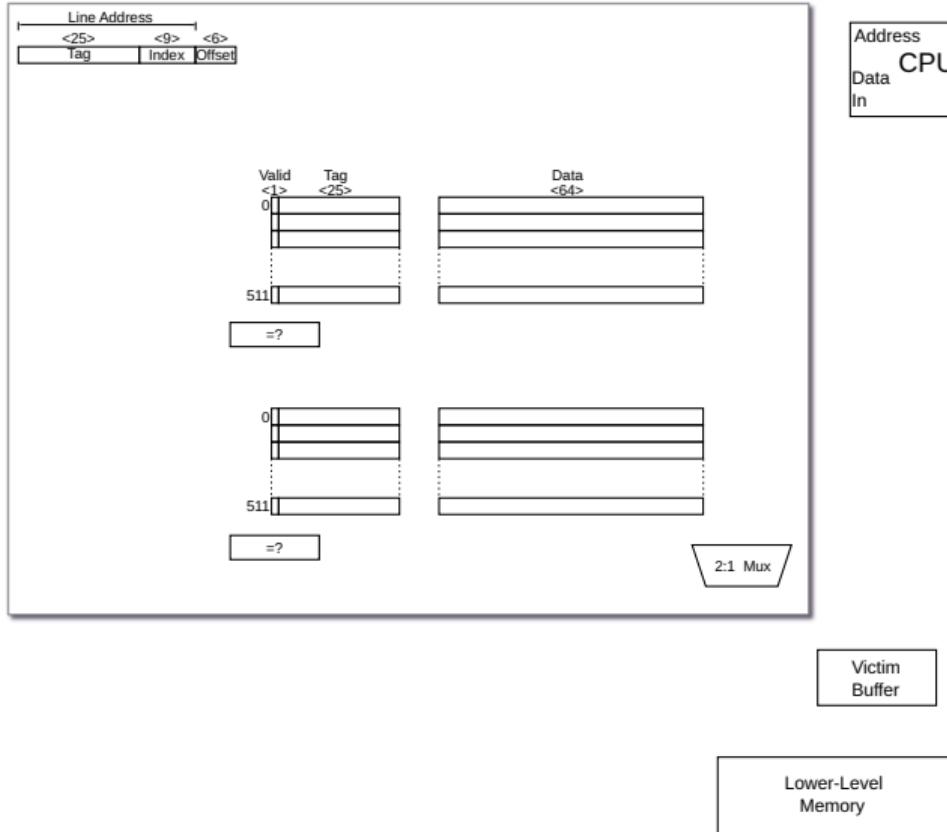
## Casos Prácticos. AMD Opteron Cache L1

- En 2003 AMD presenta esta línea de procesadores que implementa por primera vez la arquitectura de 64 bit AMD64, que a posteriori Intel debería incluir en su línea x86 luego del fracaso comercial de Itanium.
- El primer procesador de arquitectura denominada *SledgeHammer*, tenía un solo core con Ejecución Fuera de Orden y dos niveles de Cache: 64 KiB, y 256 KiB on chip.
- En 64 bit, el procesador trabaja con direcciones virtuales de 48 bit, y envía al Address Bus direcciones físicas de 40 bit (con el tiempo al igual que los Intel, este número crecerá hasta 52 bit actuales)
- El cache L1 es set-asociativo de 2 vías y trabaja con un tamaño de línea de 64 B.

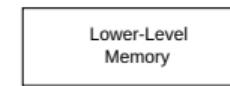
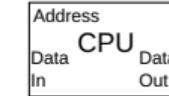
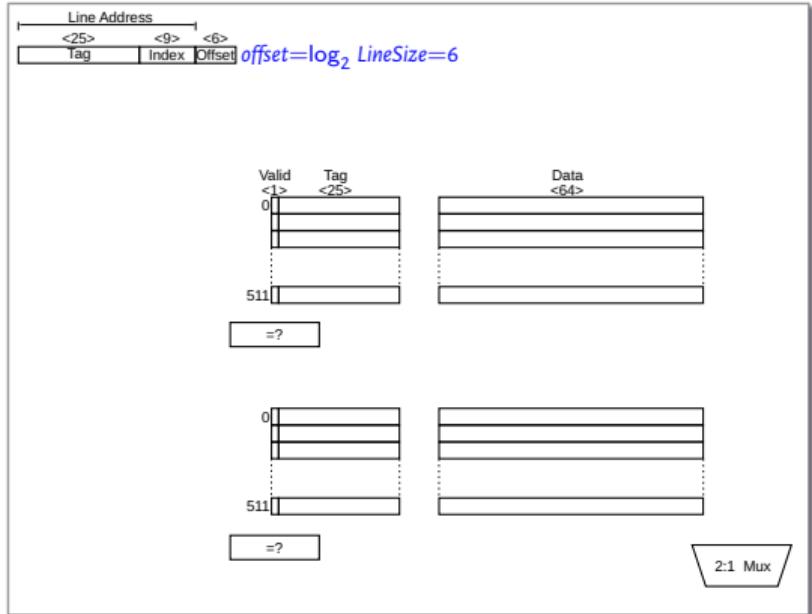
## Casos Prácticos. AMD Opteron Cache L1

- En 2003 AMD presenta esta línea de procesadores que implementa por primera vez la arquitectura de 64 bit AMD64, que a posteriori Intel debería incluir en su línea x86 luego del fracaso comercial de Itanium.
- El primer procesador de arquitectura denominada *SledgeHammer*, tenía un solo core con Ejecución Fuera de Orden y dos niveles de Cache: 64 KiB, y 256 KiB on chip.
- En 64 bit, el procesador trabaja con direcciones virtuales de 48 bit, y envía al Address Bus direcciones físicas de 40 bit (con el tiempo al igual que los Intel, este número crecerá hasta 52 bit actuales)
- El cache L1 es set-asociativo de 2 vías y trabaja con un tamaño de línea de 64 B.
- Por lo tanto el Cache L1 se compone de dos vías organizadas en 512 sets de líneas de 64 B

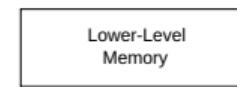
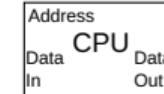
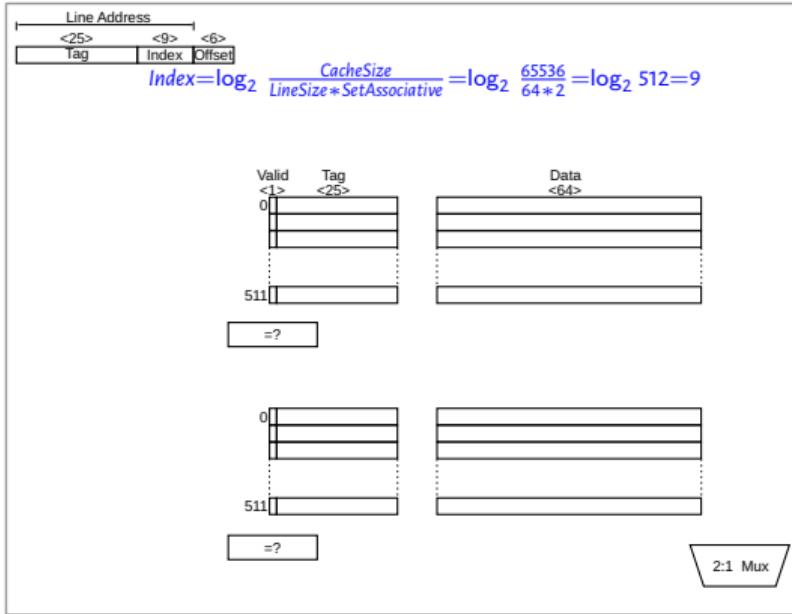
# Casos Prácticos. AMD Opteron Cache L1



# Casos Prácticos. AMD Opteron Cache L1

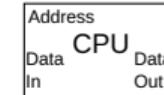
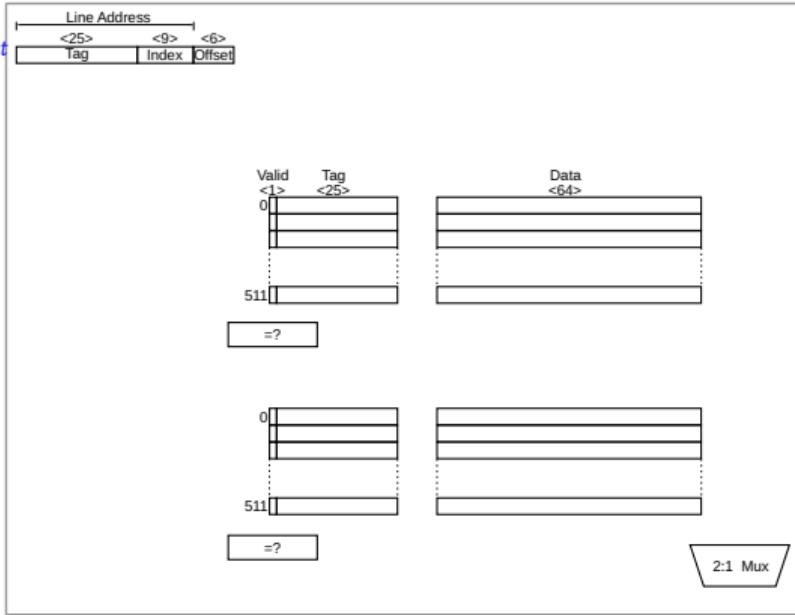


# Casos Prácticos. AMD Opteron Cache L1

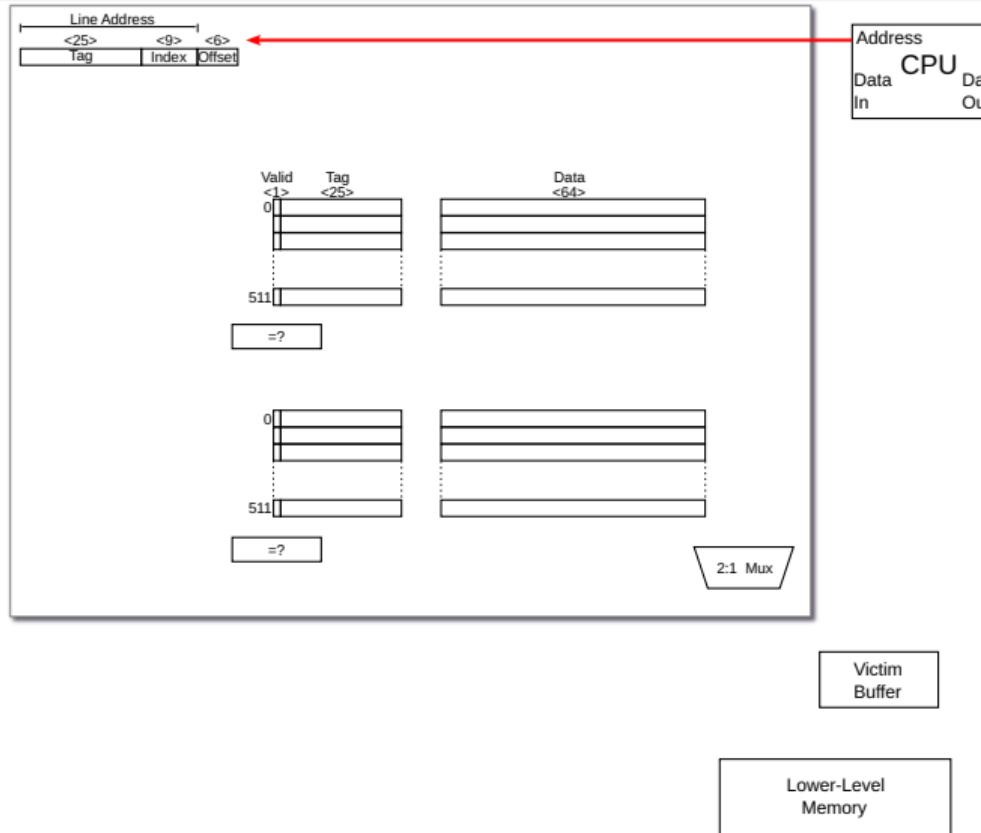


# Casos Prácticos. AMD Opteron Cache L1

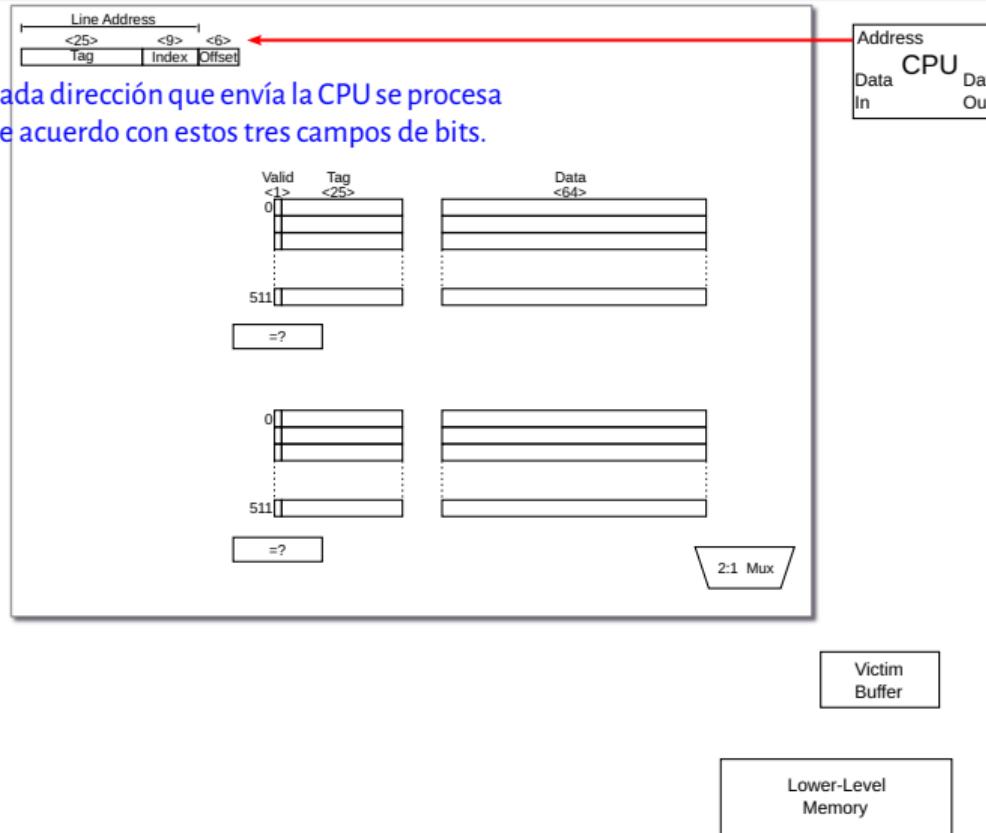
$\text{Tag} = \text{AddressSize} - \text{Index} - \text{Offset}$   
 $= 40 - 9 - 6 = 25$



# Casos Prácticos. AMD Opteron Cache L1

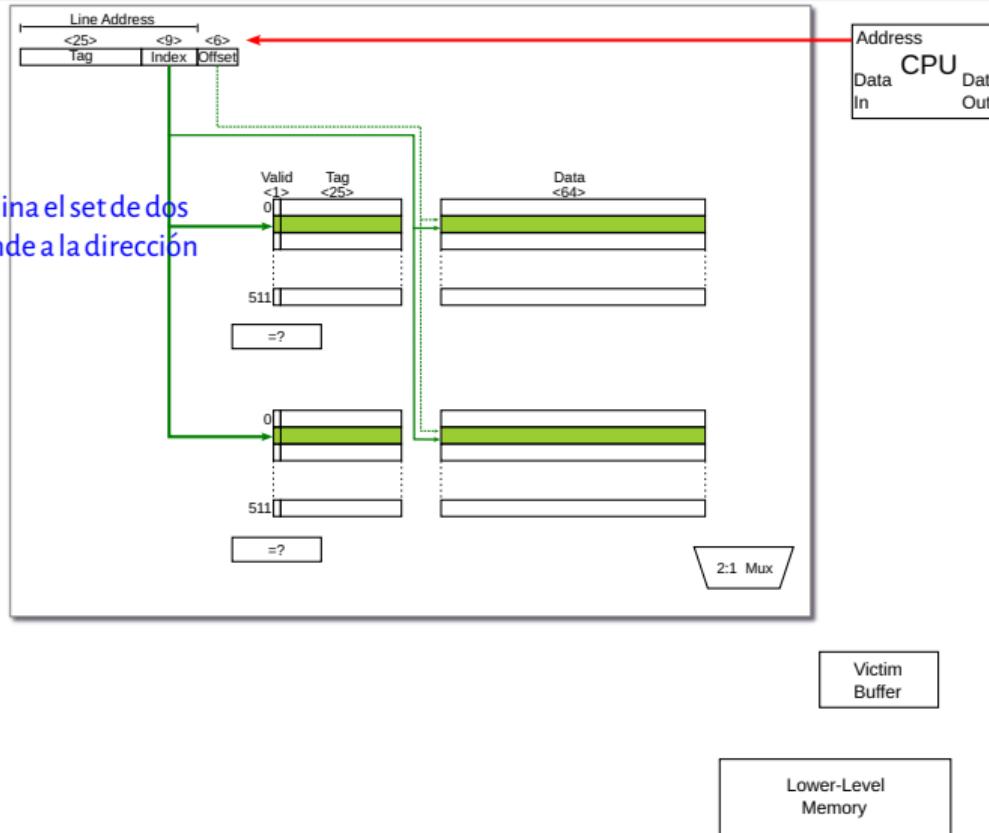


# Casos Prácticos. AMD Opteron Cache L1

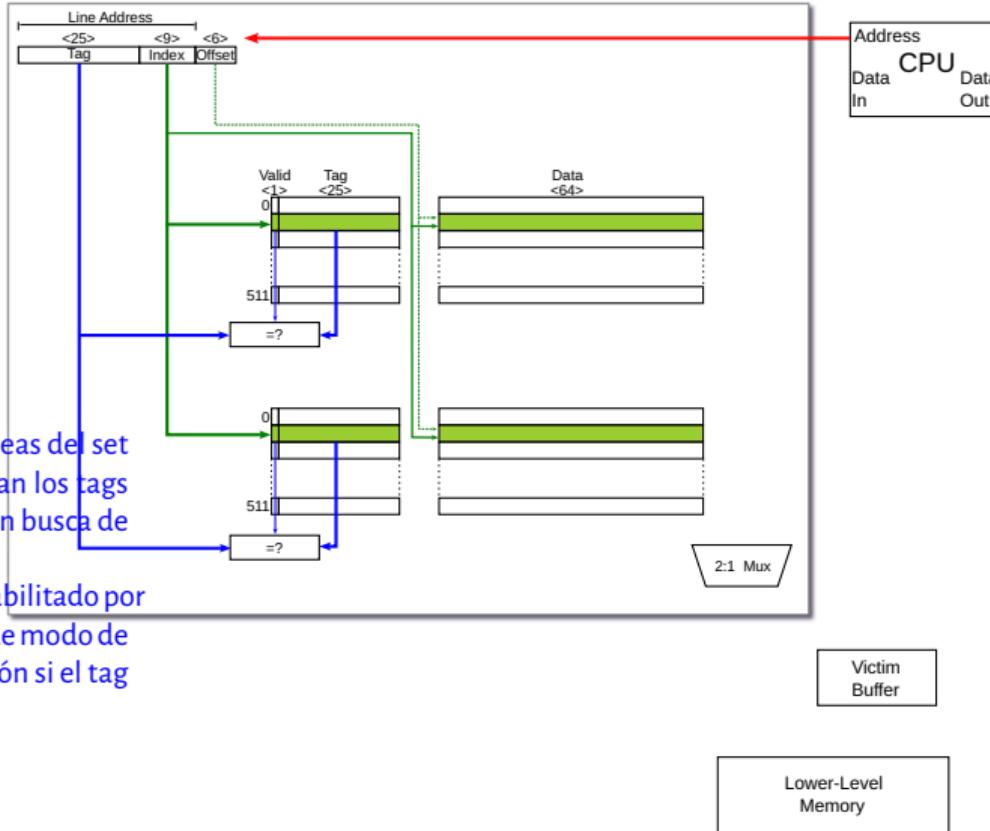


# Casos Prácticos. AMD Opteron Cache L1

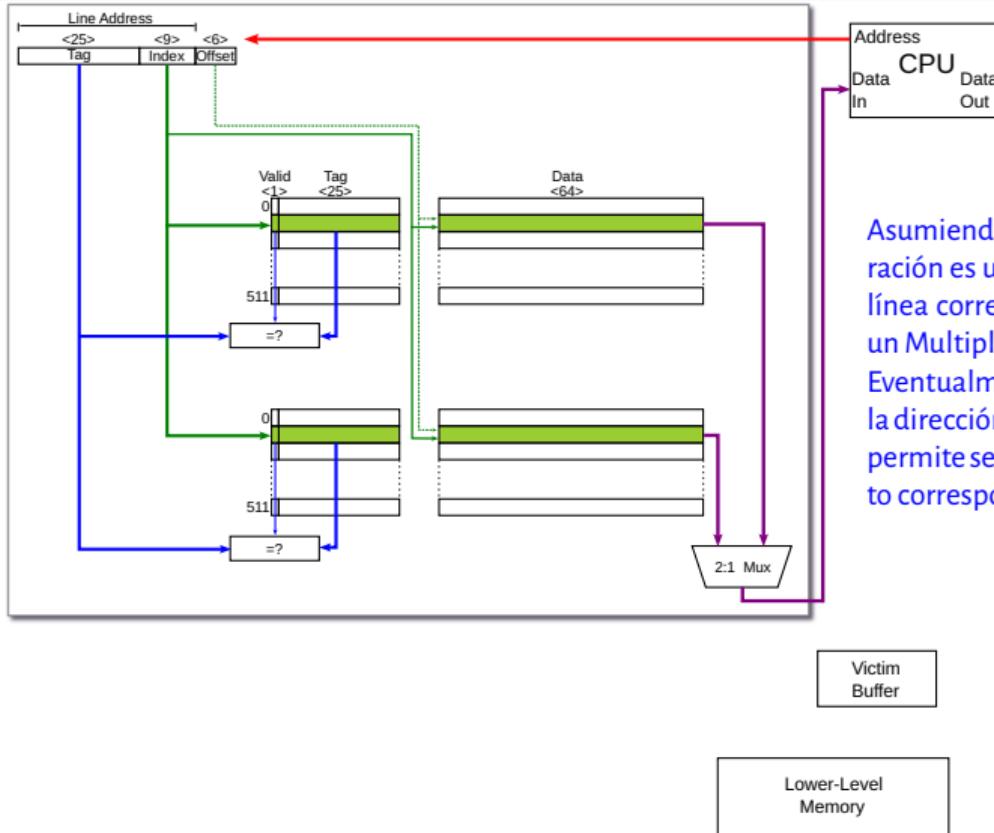
Con Index se determina el set de dos líneas que corresponde a la dirección enviada por la CPU.



# Casos Prácticos. AMD Opteron Cache L1



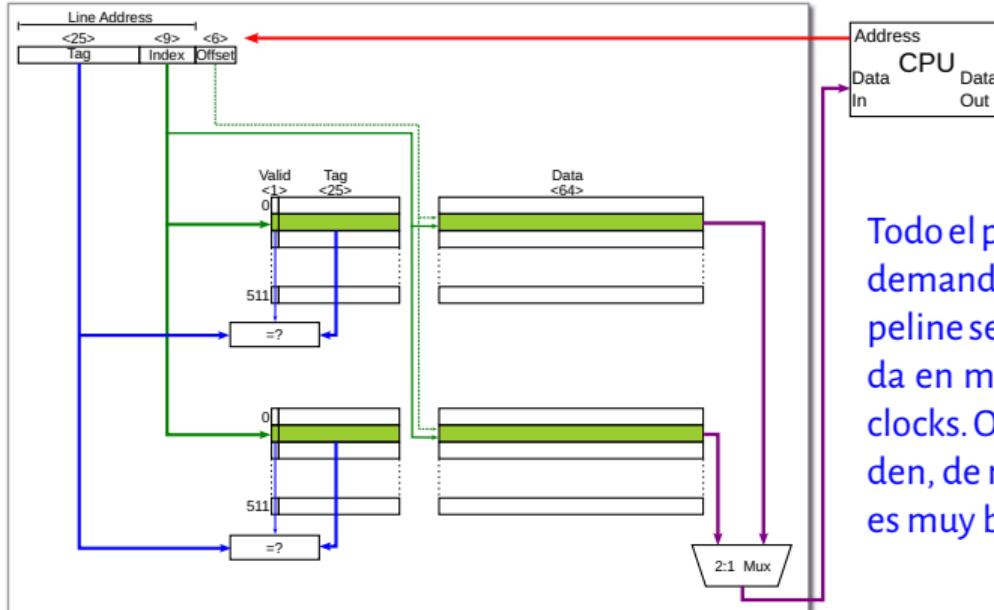
# Casos Prácticos. AMD Opteron Cache L1



Asumiendo que el resultado de la comparación es un Hit, se habilita la lectura de la línea correspondiente a la CPU, mediante un Multiplexor 2:1.

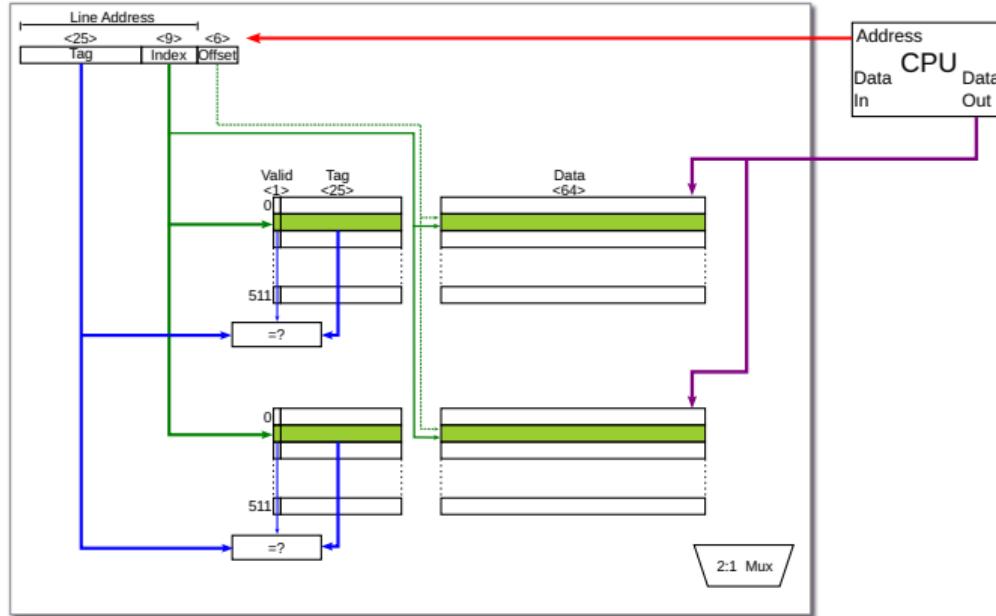
Eventualmente el campo offset de 6 bits de la dirección enviada por la CPU al principio, permite seleccionar dentro de la línea el dato correspondiente.

# Casos Prácticos. AMD Opteron Cache L1



Todo el proceso descrito hasta aquí, demanda 2 ciclos de clock. Si el pipeline se atascase por esta búsqueda en memoria, solo será por dos clocks. Opteron ejecuta Fuera de Orden, de modo que la probabilidad es muy baja.

# Casos Prácticos. AMD Opteron Cache L1



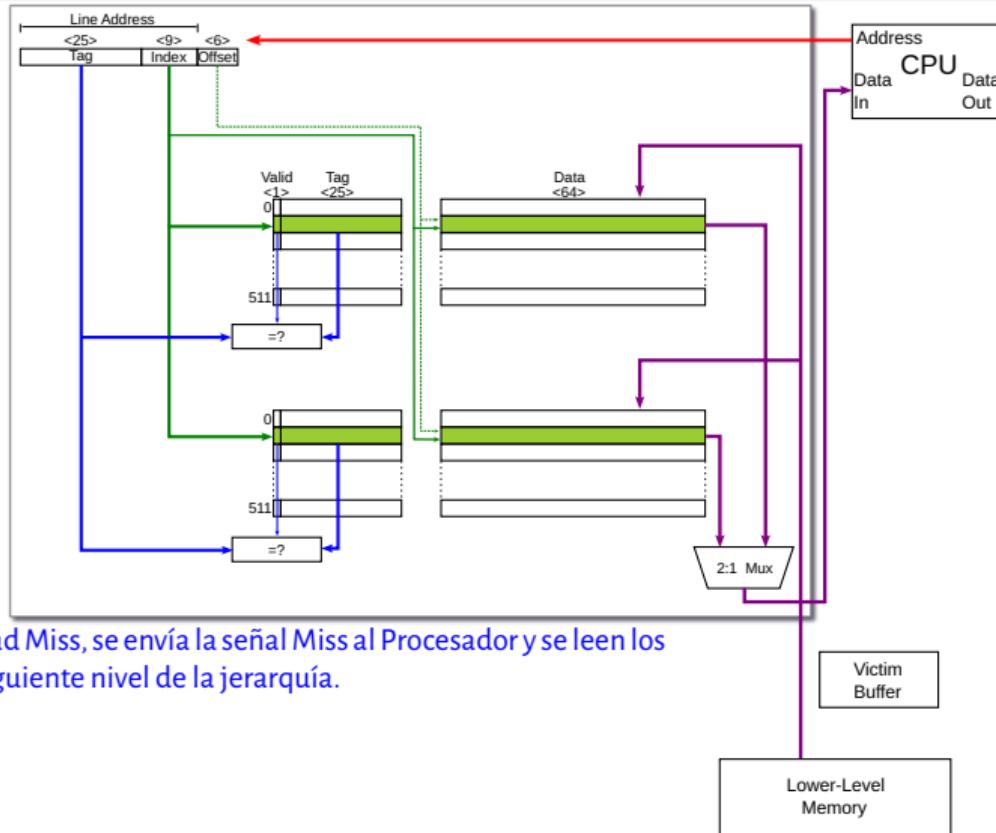
En el caso de las escrituras el proceso es algo mas complejo.

En el caso que la dirección a ser escrita esté presente en el Cache, el proceso hasta la comparación de los Tags es idéntico. Pero la escritura no se realizará hasta no tener Hit como resultado de la comparación.

Victim Buffer

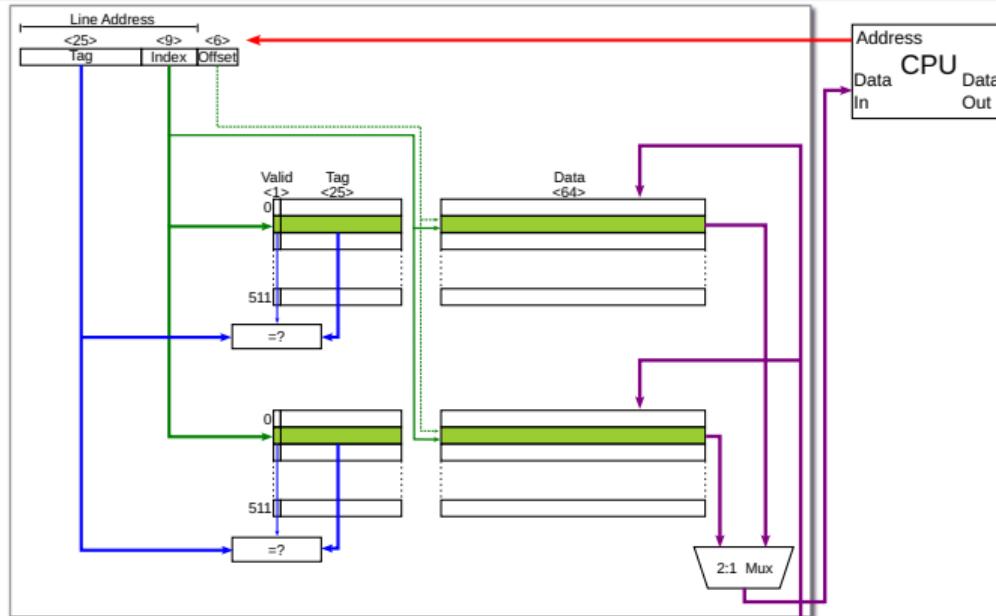
Lower-Level Memory

# Casos Prácticos. AMD Opteron Cache L1



En el caso de Read Miss, se envía la señal Miss al Procesador y se leen los 64 B desde el siguiente nivel de la jerarquía.

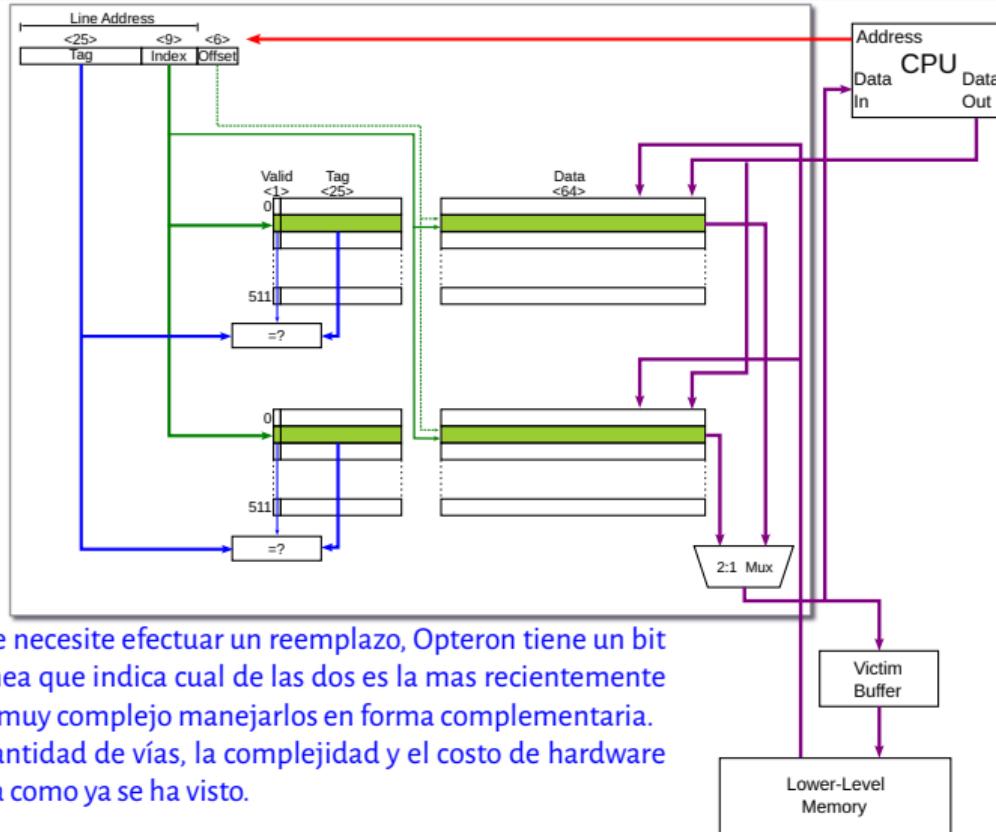
# Casos Prácticos. AMD Opteron Cache L1



El latency es de 7 clocks para los primeros 8 B, y 2 clocks para c/u de los siguientes. Total 21 clocks.

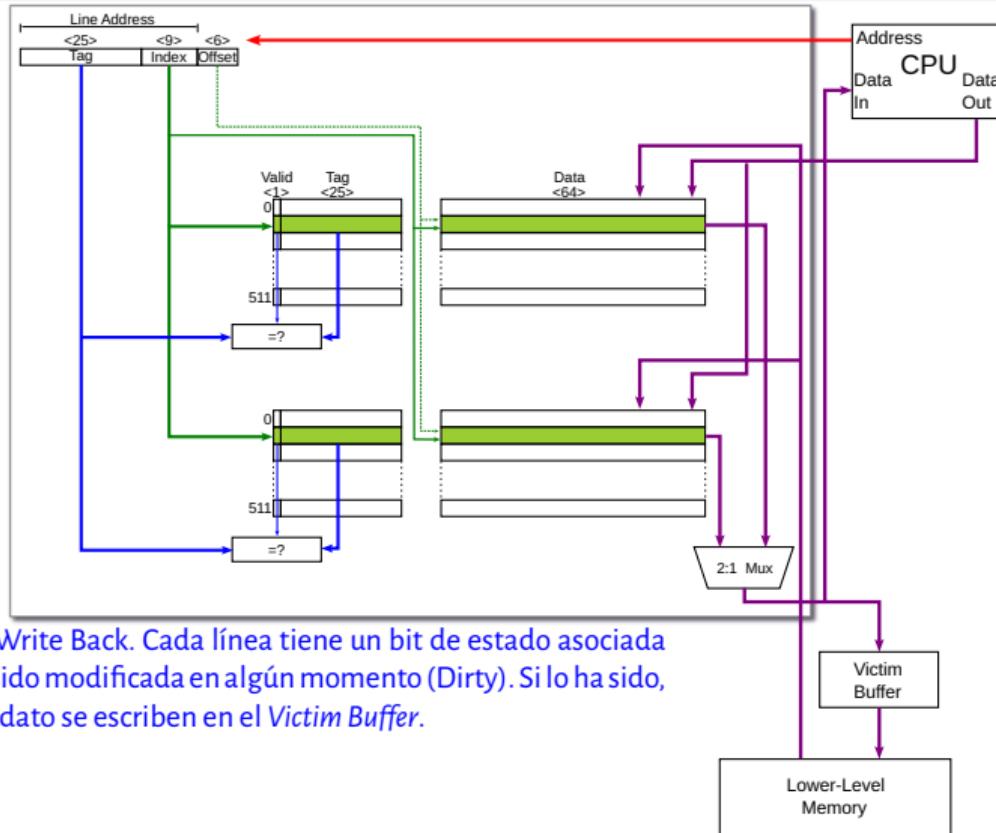
Esto sugiere que los niveles de menor jerarquía no disponen de buses tan anchos, pero a cambio soportan transferencias burst.

# Casos Prácticos. AMD Opteron Cache L1

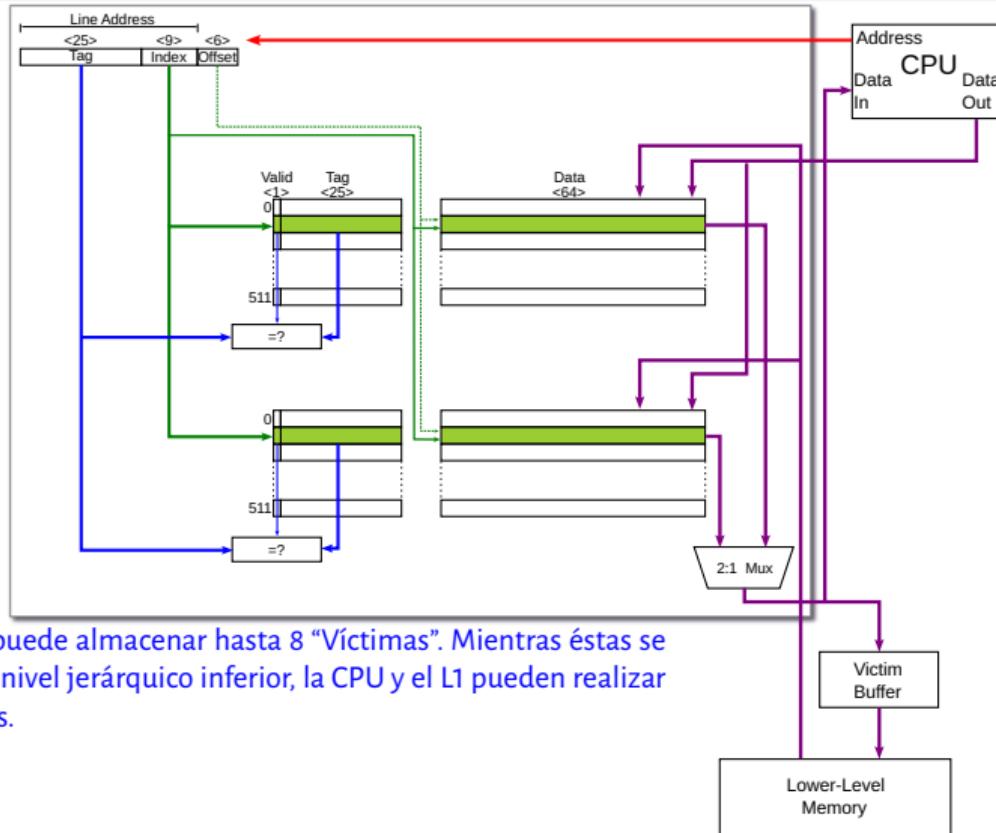


En el caso que se necesite efectuar un reemplazo, Opteron tiene un bit LRU por cada línea que indica cual de las dos es la mas recientemente accedida. No es muy complejo manejarlos en forma complementaria. Si aumenta la cantidad de vías, la complejidad y el costo de hardware de LRU aumenta como ya se ha visto.

# Casos Prácticos. AMD Opteron Cache L1

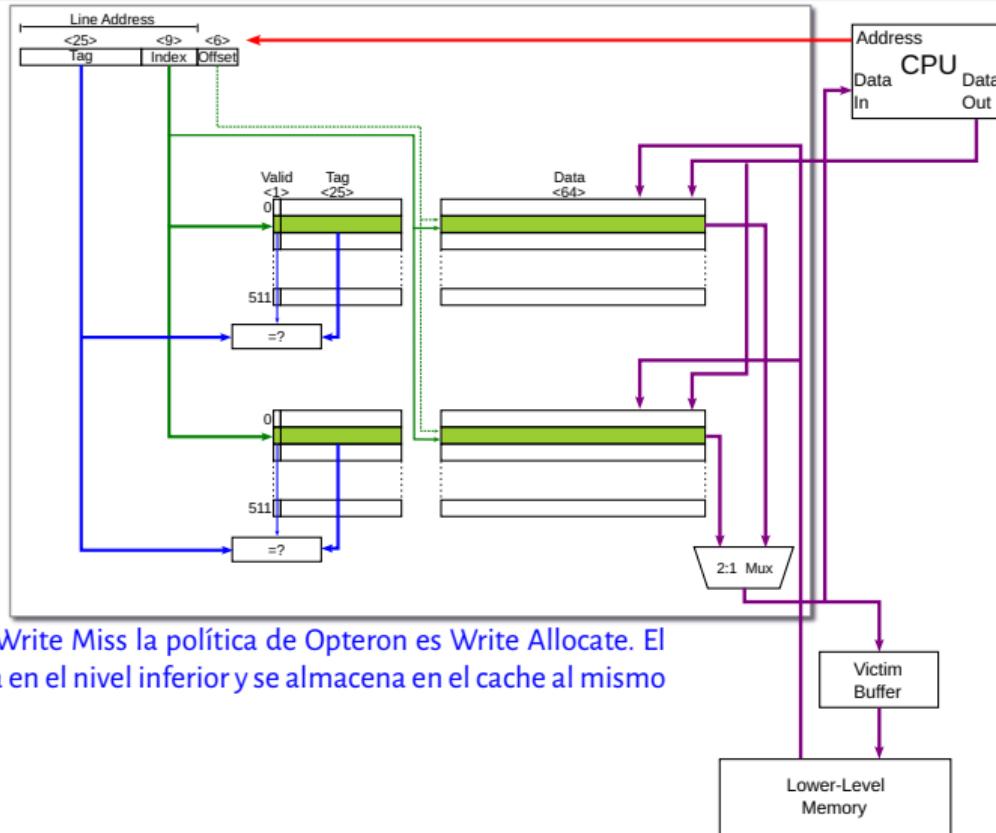


# Casos Prácticos. AMD Opteron Cache L1



El **Victim Buffer** puede almacenar hasta 8 “Víctimas”. Mientras éstas se actualizan en el nivel jerárquico inferior, la CPU y el L1 pueden realizar otras actividades.

# Casos Prácticos. AMD Opteron Cache L1



Para el caso de Write Miss la política de Opteron es Write Allocate. El dato se modifica en el nivel inferior y se almacena en el cache al mismo tiempo.

# Casos Prácticos. AMD Opteron Cache L1

- Si una CPU necesita buscar mas instrucciones, y al mismo tiempo, resolver un Load o un Store con memoria, tenemos un cuello de botella.

## Casos Prácticos. AMD Opteron Cache L1

- Si una CPU necesita buscar mas instrucciones, y al mismo tiempo, resolver un Load o un Store con memoria, tenemos un cuello de botella.
- Claro ejemplo de obstáculo estructural.

## Casos Prácticos. AMD Opteron Cache L1

- Si una CPU necesita buscar mas instrucciones, y al mismo tiempo, resolver un Load o un Store con memoria, tenemos un cuello de botella.
- Claro ejemplo de obstáculo estructural.
- Una forma de resolver este tipo de problemas es mediante la estrategia “Divide and Conquer”.

## Casos Prácticos. AMD Opteron Cache L1

- Si una CPU necesita buscar mas instrucciones, y al mismo tiempo, resolver un Load o un Store con memoria, tenemos un cuello de botella.
- Claro ejemplo de obstáculo estructural.
- Una forma de resolver este tipo de problemas es mediante la estrategia “Divide and Conquer”.
- Significa utilizar Caches separados para código (ICache) y datos (DCache).

## Casos Prácticos. AMD Opteron Cache L1

- Si una CPU necesita buscar mas instrucciones, y al mismo tiempo, resolver un Load o un Store con memoria, tenemos un cuello de botella.
- Claro ejemplo de obstáculo estructural.
- Una forma de resolver este tipo de problemas es mediante la estrategia “Divide and Conquer”.
- Significa utilizar Caches separados para código (ICache) y datos (DCache).
- Permite incluso configurar de manera diferente las Caches (tamaño de línea, cantidad de vías por ejemplo).

## Casos Prácticos. AMD Opteron Cache L1

- Si una CPU necesita buscar mas instrucciones, y al mismo tiempo, resolver un Load o un Store con memoria, tenemos un cuello de botella.
- Claro ejemplo de obstáculo estructural.
- Una forma de resolver este tipo de problemas es mediante la estrategia “Divide and Conquer”.
- Significa utilizar Caches separados para código (ICache) y datos (DCache).
- Permite incluso configurar de manera diferente las Caches (tamaño de línea, cantidad de vías por ejemplo).
- Opteron al igual que otros procesadores de su misma generación implementa esta estrategia en el primer nivel de la jerarquía de Cache.

## Casos Prácticos. AMD Opteron Cache L1

- Si una CPU necesita buscar mas instrucciones, y al mismo tiempo, resolver un Load o un Store con memoria, tenemos un cuello de botella.
- Claro ejemplo de obstáculo estructural.
- Una forma de resolver este tipo de problemas es mediante la estrategia “Divide and Conquer”.
- Significa utilizar Caches separados para código (ICache) y datos (DCache).
- Permite incluso configurar de manera diferente las Caches (tamaño de línea, cantidad de vías por ejemplo).
- Opteron al igual que otros procesadores de su misma generación implementa esta estrategia en el primer nivel de la jerarquía de Cache.
- Es decir que en verdad Opteron tiene dos Caches L1 de 64 KiB.

## Casos Prácticos. Split Cache

Tamaño [KiB]	ICache	DCache	Unificado
8	8.16	44.0	63.0
16	3.82	40.9	51.0
32	1.36	38.4	43.3
64	0.61	36.9	39.4
128	0.30	35.3	36.2
256	0.02	32.6	32.9

*Misses por cada 100 instrucciones.*

Las cache de Instrucciones tienen Miss rates notablemente mejores que las de Datos.  
Para comparar Split cache vs. Cache Unificado, debe tenerse la misma cantidad de memoria. Ej: Split Cache de 16 KiB+16 KiB con Unified de 32 KiB  
El Miss Rate no es por lo tanto una razón para usar split Cache. La performance si lo es.

## Algunas conclusiones preliminares

Lo que buscan los Sistemas Multiprocesador y los Subsistemas de E/S es lo mejor de los dos mundos: El mínimo tráfico y consumo de energía transaccional, y la menor penalización de performance que proporciona Write Back, y al mismo tiempo la robustez en materia de coherencia que proporciona Write Through.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

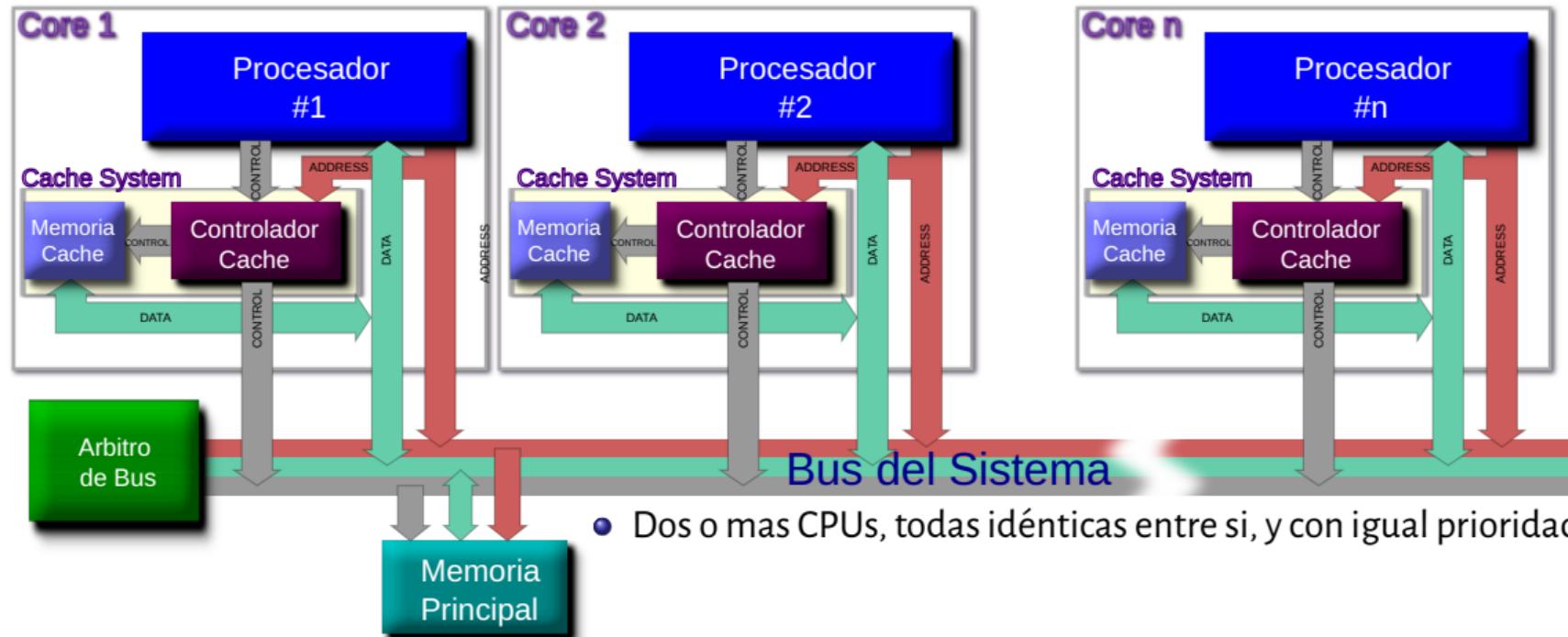
## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

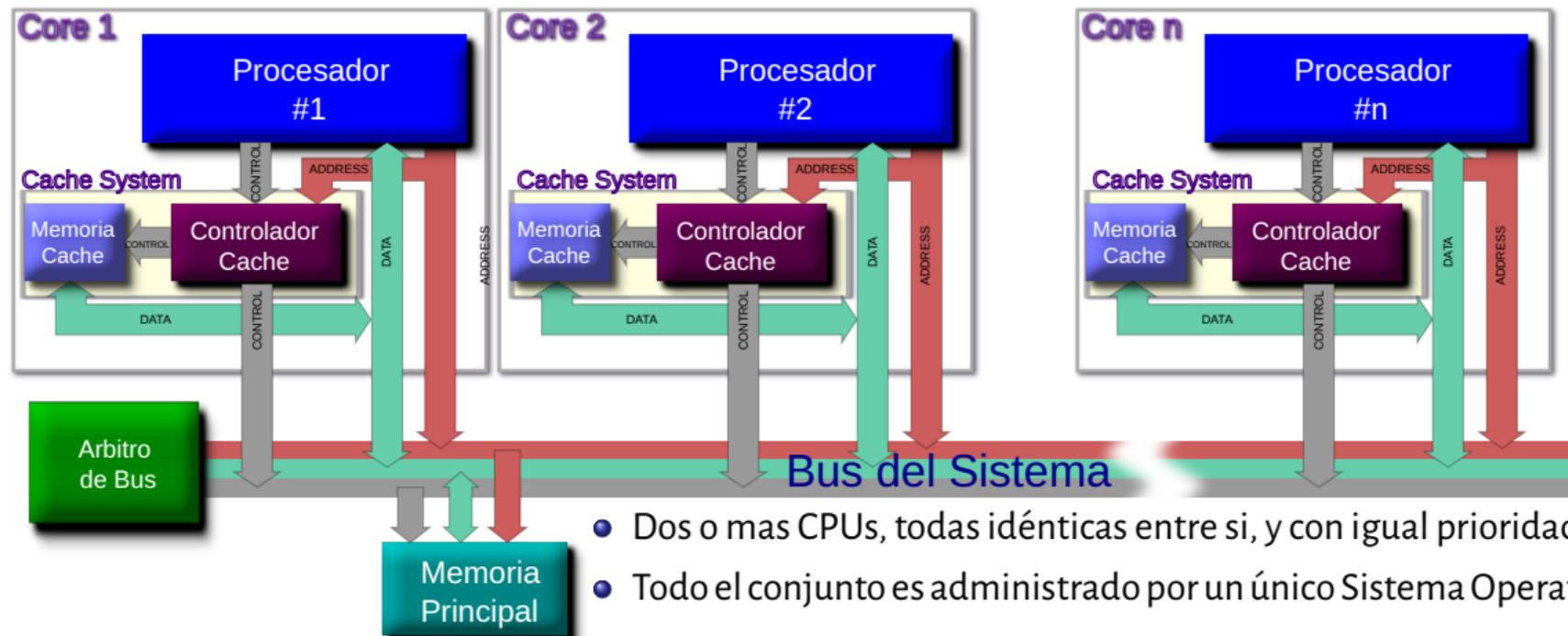
## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

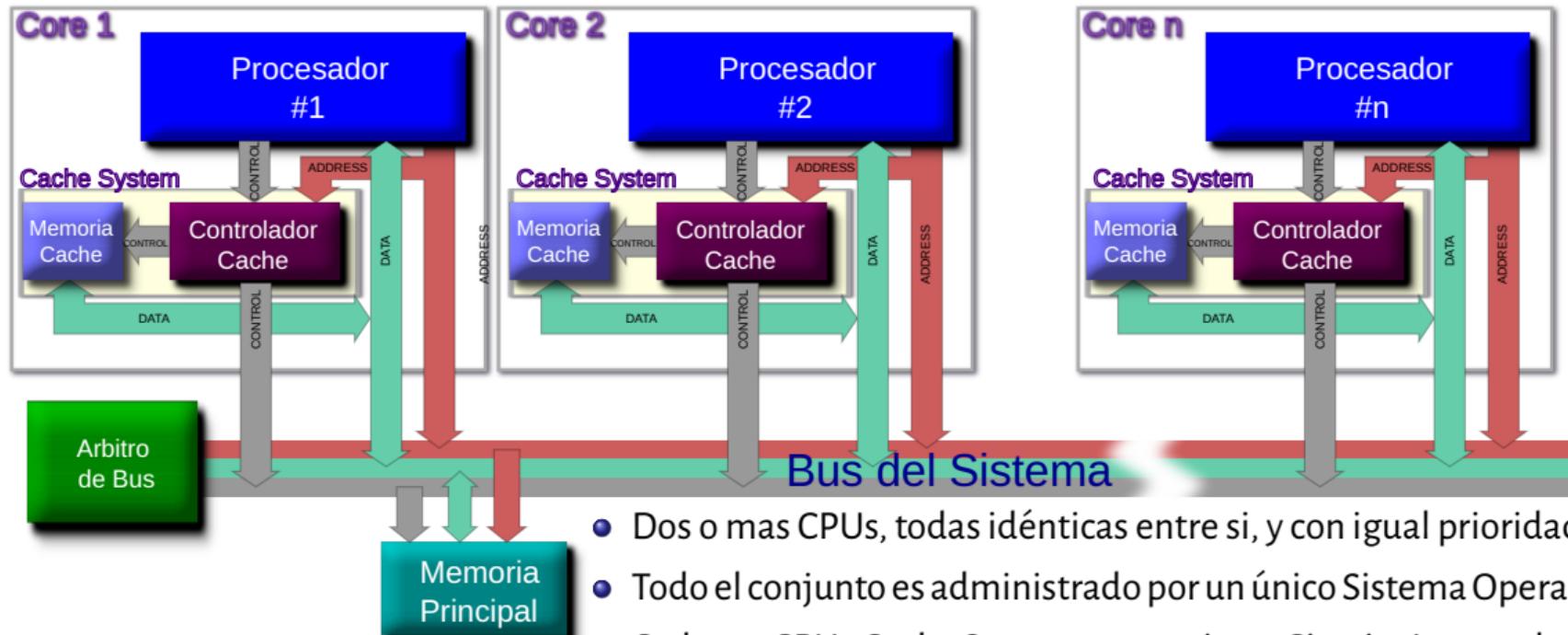
# Sistemas SMP (Simetrical Multi Processing)



# Sistemas SMP (Simetrical Multi Processing)

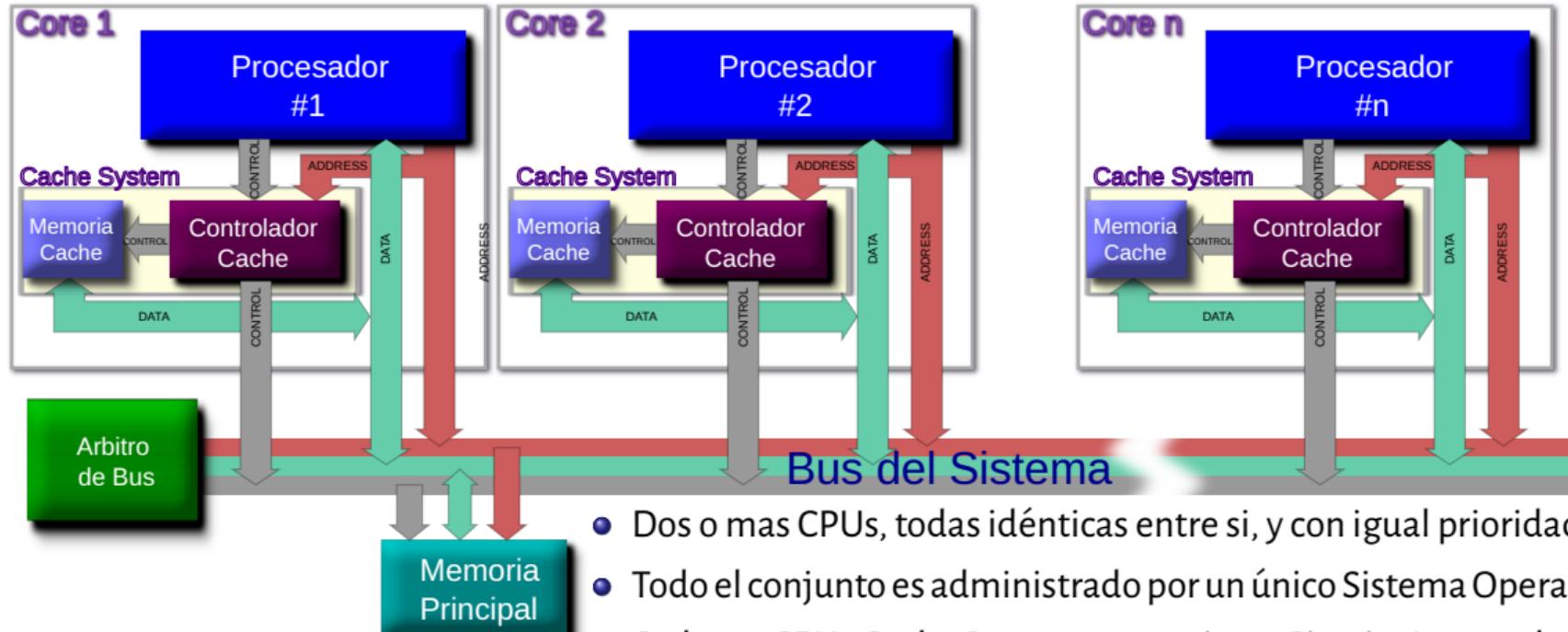


# Sistemas SMP (Simetrical Multi Processing)



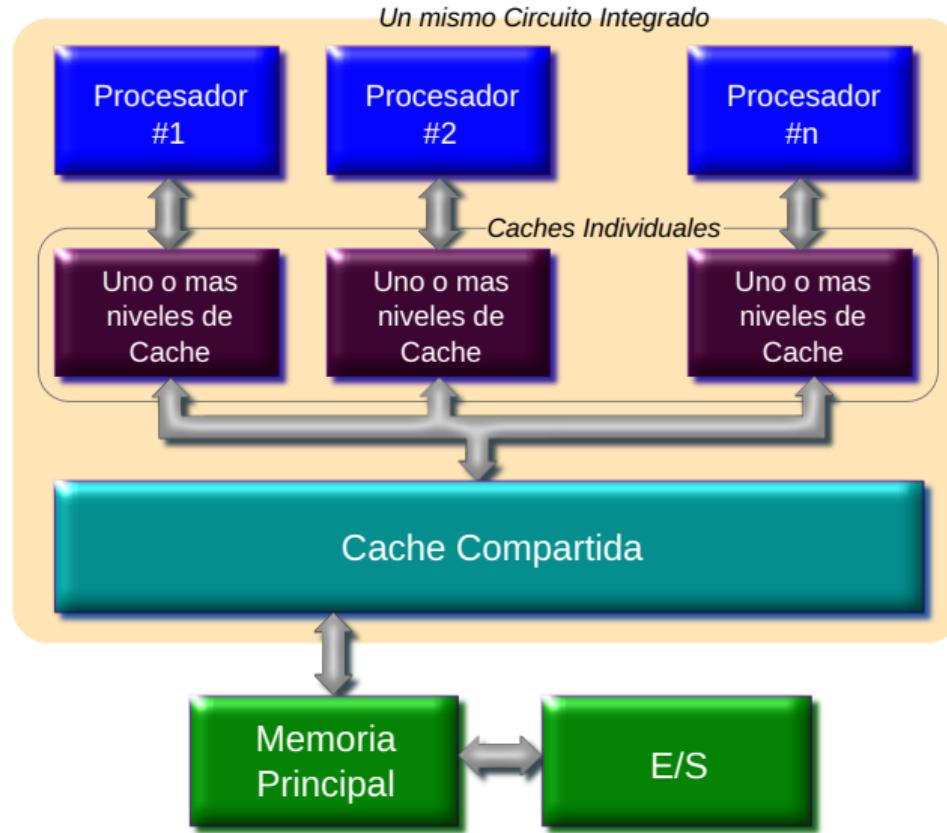
- Dos o mas CPUs, todas idénticas entre si, y con igual prioridad.
- Todo el conjunto es administrado por un único Sistema Operativo.
- Cada par CPU - Cache System en un mismo Circuito Integrado (CI), se denomina Core.

# Sistemas SMP (Simetrical Multi Processing)



- Dos o mas CPUs, todas idénticas entre si, y con igual prioridad.
- Todo el conjunto es administrado por un único Sistema Operativo.
- Cada par CPU - Cache System en un mismo Circuito Integrado (CI), se denomina Core.
- Este mismo diagrama en un único CI es un sistema Multi Core.

# Sistemas SMP Multicore



# Sistemas DSM (Distributed Shared Memory)

- Los Sistemas SMP (Mono Core o Multicore), se denominan Multiprocesadores **UMA** (por **Uniform Memory Access**)

# Sistemas DSM (Distributed Shared Memory)

- Los Sistemas SMP (Mono Core o Multicore), se denominan Multiprocesadores **UMA** (por **Uniform Memory Access**)
- Todos los cores tienen el mismo latency para acceder a memoria, aun cuando ésta se organiza en varios niveles y bancos.

# Sistemas DSM (Distributed Shared Memory)

- Los Sistemas SMP (Mono Core o Multicore), se denominan Multiprocesadores **UMA** (por **Uniform Memory Access**)
- Todos los cores tienen el mismo latency para acceder a memoria, aun cuando ésta se organiza en varios niveles y bancos.
- A medida que la cantidad de Cores aumenta, el modelo SMP Multicore requiere mas ancho de banda de la memoria DRAM, ya que de otro modo el latency de acceso se vuelve inviable.

# Sistemas DSM (Distributed Shared Memory)

- Los Sistemas SMP (Mono Core o Multicore), se denominan Multiprocesadores **UMA** (por **Uniform Memory Access**)
- Todos los cores tienen el mismo latency para acceder a memoria, aun cuando ésta se organiza en varios niveles y bancos.
- A medida que la cantidad de Cores aumenta, el modelo SMP Multicore requiere mas ancho de banda de la memoria DRAM, ya que de otro modo el latency de acceso se vuelve inviable.
- Una alternativa que está consolidándose es **DSM**.

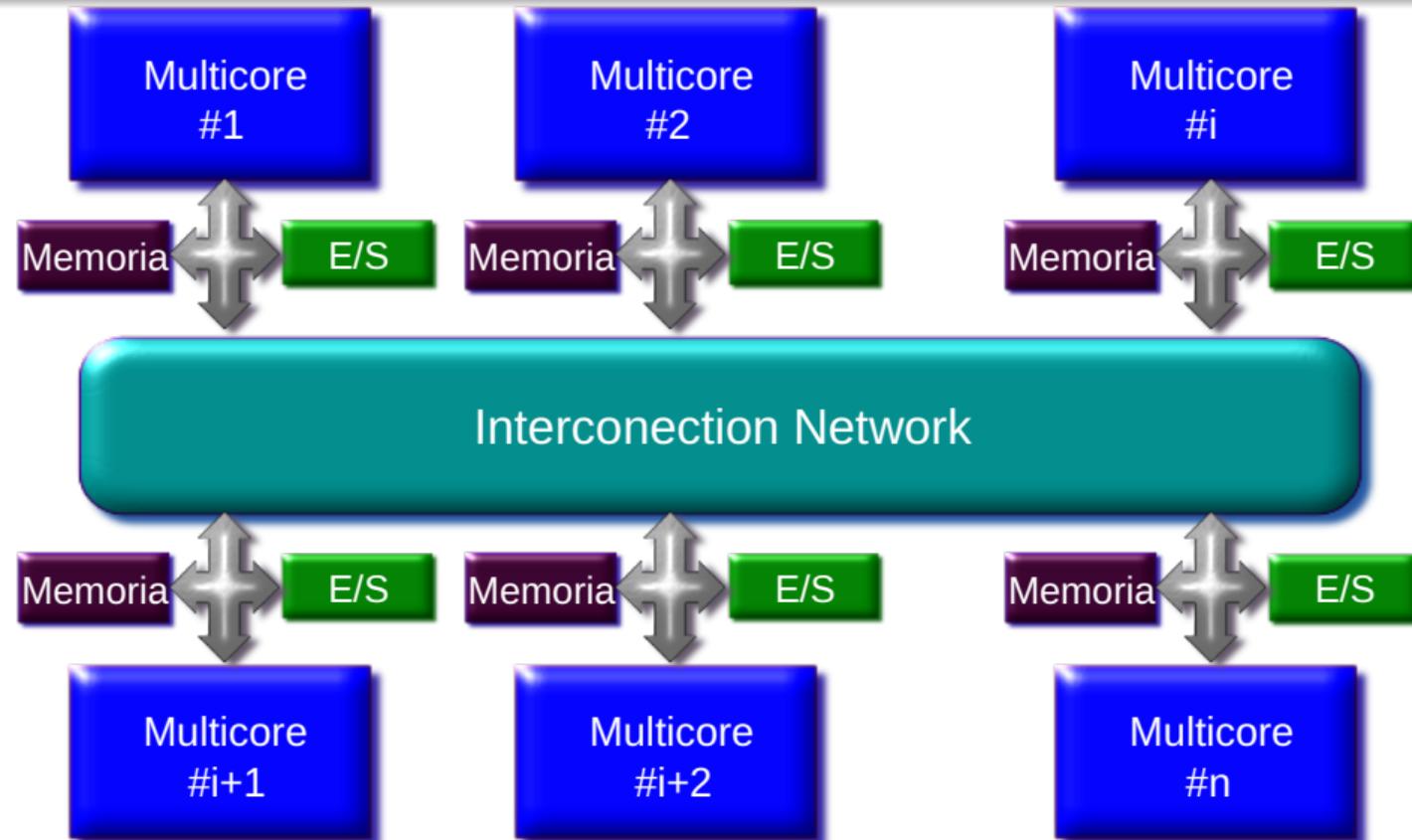
# Sistemas DSM (Distributed Shared Memory)

- Los Sistemas SMP (Mono Core o Multicore), se denominan Multiprocesadores **UMA** (por **Uniform Memory Access**)
- Todos los cores tienen el mismo latency para acceder a memoria, aun cuando ésta se organiza en varios niveles y bancos.
- A medida que la cantidad de Cores aumenta, el modelo SMP Multicore requiere mas ancho de banda de la memoria DRAM, ya que de otro modo el latency de acceso se vuelve inviable.
- Una alternativa que está consolidándose es **DSM**.
- En **DSM** la memoria se encuentra distribuida por grupos de cores. Cada grupo accede tanto a su memoria local, como a la del resto.

# Sistemas DSM (Distributed Shared Memory)

- Los Sistemas SMP (Mono Core o Multicore), se denominan Multiprocesadores **UMA** (por **Uniform Memory Access**)
- Todos los cores tienen el mismo latency para acceder a memoria, aun cuando ésta se organiza en varios niveles y bancos.
- A medida que la cantidad de Cores aumenta, el modelo SMP Multicore requiere mas ancho de banda de la memoria DRAM, ya que de otro modo el latency de acceso se vuelve inviable.
- Una alternativa que está consolidándose es **DSM**.
- En **DSM** la memoria se encuentra distribuida por grupos de cores. Cada grupo accede tanto a su memoria local, como a la del resto.
- Estos esquemas aseguran bajo latency y suficiente ancho de banda en la Memoria local de cada Core. Pero su acceso a memoria es No Uniforme (**NUMA**), ya que su latency depende de donde esté ubicado el ítem que se desea acceder

# Sistemas DSM Multicore



# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

### • Organización de Sistemas Multiprocesador

### • Coherencia de un cache

- Protocolos de Coherencia para Multicore
- Casos del Mundo real

### 5 Memorias Dinámicas

- Introducción
- Organización interna

### 6 Standards

- Estado del arte
- JEDEC SDRAM

### 7 Controladores de Memoria

- Introducción General
- Arquitectura

### 8 Configuración

- Configuración del DRAM Device

### • Entrada Salida de Datos

### 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

### 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

### 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

### 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# ¿Qué ocurre durante las escrituras?

- Está claro que cuando se escribe se trata de un dato.

# ¿Qué ocurre durante las escrituras?

- Está claro que cuando se escribe se trata de un dato.
- Cada variable almacenada en un Nivel del Cache también está alojada en el resto de los niveles inferiores hasta la DRAM.

## ¿Qué ocurre durante las escrituras?

- Está claro que cuando se escribe se trata de un dato.
- Cada variable almacenada en un Nivel del Cache también está alojada en el resto de los niveles inferiores hasta la DRAM.
- Idealmente estas copias de la variable deben mantener el mismo valor.

## ¿Qué ocurre durante las escrituras?

- Está claro que cuando se escribe se trata de un dato.
- Cada variable almacenada en un Nivel del Cache también está alojada en el resto de los niveles inferiores hasta la DRAM.
- Idealmente estas copias de la variable deben mantener el mismo valor.
- Cuando se requiere escribirla, ésta condición dejará de cumplirse.

## ¿Qué ocurre durante las escrituras?

- Está claro que cuando se escribe se trata de un dato.
- Cada variable almacenada en un Nivel del Cache también está alojada en el resto de los niveles inferiores hasta la DRAM.
- Idealmente estas copias de la variable deben mantener el mismo valor.
- Cuando se requiere escribirla, ésta condición dejará de cumplirse.
- Ésta se modifica en el cache de ese procesador, y queda diferente de la(s) copia(s) en su jerarquía de memoria, desde el nivel inmediato inferior hasta la Memoria Principal (DRAM).

## ¿Qué ocurre durante las escrituras?

- Está claro que cuando se escribe se trata de un dato.
- Cada variable almacenada en un Nivel del Cache también está alojada en el resto de los niveles inferiores hasta la DRAM.
- Idealmente estas copias de la variable deben mantener el mismo valor.
- Cuando se requiere escribirla, ésta condición dejará de cumplirse.
- Ésta se modifica en el cache de ese procesador, y queda diferente de la(s) copia(s) en su jerarquía de memoria, desde el nivel inmediato inferior hasta la Memoria Principal (DRAM).
- La estrategia para mantener la coherencia entre las diferentes copias a lo largo de la jerarquía, como hemos visto depende, entre otros factores, de número de CPUs que haya en el sistema.

# Coherencia de Memoria en Sistemas SMP

- Independientemente de la estrategia de escritura adoptada (Write Through o Write Back), en algún momento esta variable habrá de actualizarse hacia los niveles inferiores de la Jerarquía.

# Coherencia de Memoria en Sistemas SMP

- Independientemente de la estrategia de escritura adoptada (Write Through o Write Back), en algún momento esta variable habrá de actualizarse hacia los niveles inferiores de la Jerarquía.
- Pero si el sistema es SMP, y se ejecuta un programa paralelizado vía threads, éstos pueden ejecutarse en varias CPU al mismo tiempo.

# Coherencia de Memoria en Sistemas SMP

- Independientemente de la estrategia de escritura adoptada (Write Through o Write Back), en algún momento esta variable habrá de actualizarse hacia los niveles inferiores de la Jerarquía.
- Pero si el sistema es SMP, y se ejecuta un programa paralelizado vía threads, éstos pueden ejecutarse en varias CPU al mismo tiempo.
- Eventualmente pueden existir copias de una misma variable en uno o mas Cache de los restantes Cores.

# Coherencia de Memoria en Sistemas SMP

- Independientemente de la estrategia de escritura adoptada (Write Through o Write Back), en algún momento esta variable habrá de actualizarse hacia los niveles inferiores de la Jerarquía.
- Pero si el sistema es SMP, y se ejecuta un programa paralelizado vía threads, éstos pueden ejecutarse en varias CPU al mismo tiempo.
- Eventualmente pueden existir copias de una misma variable en uno o mas Cache de los restantes Cores.
- Estas copias se han vuelto obsoletas producto de la modificación.

# Coherencia de Memoria en Sistemas SMP

- Independientemente de la estrategia de escritura adoptada (Write Through o Write Back), en algún momento esta variable habrá de actualizarse hacia los niveles inferiores de la Jerarquía.
- Pero si el sistema es SMP, y se ejecuta un programa paralelizado vía threads, éstos pueden ejecutarse en varias CPU al mismo tiempo.
- Eventualmente pueden existir copias de una misma variable en uno o mas Cache de los restantes Cores.
- Estas copias se han vuelto obsoletas producto de la modificación.
- Por lo tanto los demás Cores deben enterarse cuanto antes del cambio, para al menos invalidar la línea cuyo valor es obsoleto.

# Coherencia de Memoria en Sistemas SMP

En principio pareciera que WB (Write Back) no puede utilizarse en un sistema SMP, ya que un cambio no sería actualizado hacia los niveles inferiores de la jerarquía, sino hasta que este dato se desaloje del Cache, habiéndose omitido además todos los estados intermedios de la variable en cuestión. El desafío consiste en asegurar Coherencia para los datos compartidos entre los diferentes Sistemas Cache sin sacrificar performance.

# Esquemas básicos para asegurar Coherencia

- Un sistema de Coherencia de Caches debe proporcionar *Migración* y *Replicado* para los ítems compartidos.

## Esquemas básicos para asegurar Coherencia

- Un sistema de Coherencia de Caches debe proporcionar *Migración* y *Replicado* para los ítems compartidos.
- *Migración* es lo que ocurre habitualmente cuando un ítem se mueve a un Cache de manera transparente reduciendo demora en su acceso y demanda de ancho de banda al Bus del sistema.

## Esquemas básicos para asegurar Coherencia

- Un sistema de Coherencia de Caches debe proporcionar *Migración* y *Replicado* para los ítems compartidos.
- *Migración* es lo que ocurre habitualmente cuando un ítem se mueve a un Cache de manera transparente reduciendo demora en su acceso y demanda de ancho de banda al Bus del sistema.
- Pero cuando un dato se mueve a un Cache en un sistema SMP, el sistema debe proporcionar *Replicado* si más de un Core requiere leer el dato de manera simultánea. De este modo se evitan contenciones de Bus para acceso al dato.

## Esquemas básicos para asegurar Coherencia

- Un sistema de Coherencia de Caches debe proporcionar *Migración* y *Replicado* para los ítems compartidos.
- *Migración* es lo que ocurre habitualmente cuando un ítem se mueve a un Cache de manera transparente reduciendo demora en su acceso y demanda de ancho de banda al Bus del sistema.
- Pero cuando un dato se mueve a un Cache en un sistema SMP, el sistema debe proporcionar *Replicado* si más de un Core requiere leer el dato de manera simultánea. De este modo se evitan contenciones de Bus para acceso al dato.
- Ambos atributos resultan críticos para asegurar acceso a datos compartidos sin perder performance.

## Esquemas básicos para asegurar Coherencia

- Un sistema de Coherencia de Caches debe proporcionar *Migración* y *Replicado* para los ítems compartidos.
- *Migración* es lo que ocurre habitualmente cuando un ítem se mueve a un Cache de manera transparente reduciendo demora en su acceso y demanda de ancho de banda al Bus del sistema.
- Pero cuando un dato se mueve a un Cache en un sistema SMP, el sistema debe proporcionar *Replicado* si más de un Core requiere leer el dato de manera simultánea. De este modo se evitan contenciones de Bus para acceso al dato.
- Ambos atributos resultan críticos para asegurar acceso a datos compartidos sin perder performance.
- Los Sistemas SMP implementan ambos atributos por hardware, introduciendo Protocolos de Coherencia.

# Protocolos de Coherencia

- Son mecanismos para llevar un tracking de cada línea compartida entre cores.

# Protocolos de Coherencia

- Son mecanismos para llevar un tracking de cada línea compartida entre cores.
- Requiere poder establecer estados para cada línea. Del mismo modo que en un sistema con un solo procesador se señala **Dirty** a una línea modificada.

# Protocolos de Coherencia

- Son mecanismos para llevar un tracking de cada línea compartida entre cores.
- Requiere poder establecer estados para cada línea. Del mismo modo que en un sistema con un solo procesador se señala **Dirty** a una línea modificada.
- Hay dos clases:

# Protocolos de Coherencia

- Son mecanismos para llevar un tracking de cada línea compartida entre cores.
- Requiere poder establecer estados para cada línea. Del mismo modo que en un sistema con un solo procesador se señala **Dirty** a una línea modificada.
- Hay dos clases:

**Directory based:** El estado de las líneas se mantiene en un Directorio centralizado accesible de manera rápida por todos los Cores. Puede ser un area de cache externa a cada Core. Es mas complejo. En auge a partir de Sistemas Distributed Shared Memory, en Sistemas Multicore modernos debido a los múltiples niveles de Cache integrados.

# Protocolos de Coherencia

- Son mecanismos para llevar un tracking de cada línea compartida entre cores.
- Requiere poder establecer estados para cada línea. Del mismo modo que en un sistema con un solo procesador se señala **Dirty** a una línea modificada.
- Hay dos clases:

**Directory based:** El estado de las líneas se mantiene en un Directorio centralizado accesible de manera rápida por todos los Cores. Puede ser un area de cache externa a cada Core. Es mas complejo. En auge a partir de Sistemas Distributed Shared Memory, en Sistemas Multicore modernos debido a los múltiples niveles de Cache integrados.

**Snoopy:** Cada Core mantiene el estado de sus líneas y puede monitorear lo que hace el resto mediante un bus dedicado. Es el mas común.

# Protocolos de Coherencia

- Dos alternativas para implementar un protocolo de coherencia:

# Protocolos de Coherencia

- Dos alternativas para implementar un protocolo de coherencia:

**write invalidate** Asegura acceso exclusivo al procesador que va a escribir el dato, antes de que la escritura se lleve a cabo. El resto debe poder detectar la escritura e invalidar la línea que contiene el dato en caso de tenerla en su Cache. Garantiza que no existe otra copia del dato una vez escrito éste en el cache destino.

# Protocolos de Coherencia

- Dos alternativas para implementar un protocolo de coherencia:

**write invalidate** Asegura acceso exclusivo al procesador que va a escribir el dato, antes de que la escritura se lleve a cabo. El resto debe poder detectar la escritura e invalidar la línea que contiene el dato en caso de tenerla en su Cache. Garantiza que no existe otra copia del dato una vez escrito éste en el cache destino.

**write update** También conocido como **write broadcast**, ya que actualiza la línea en todos los Caches que la tengan consumiendo bastante ancho de banda. La ventaja es que asegura menos Miss Rate

# Protocolos de Coherencia

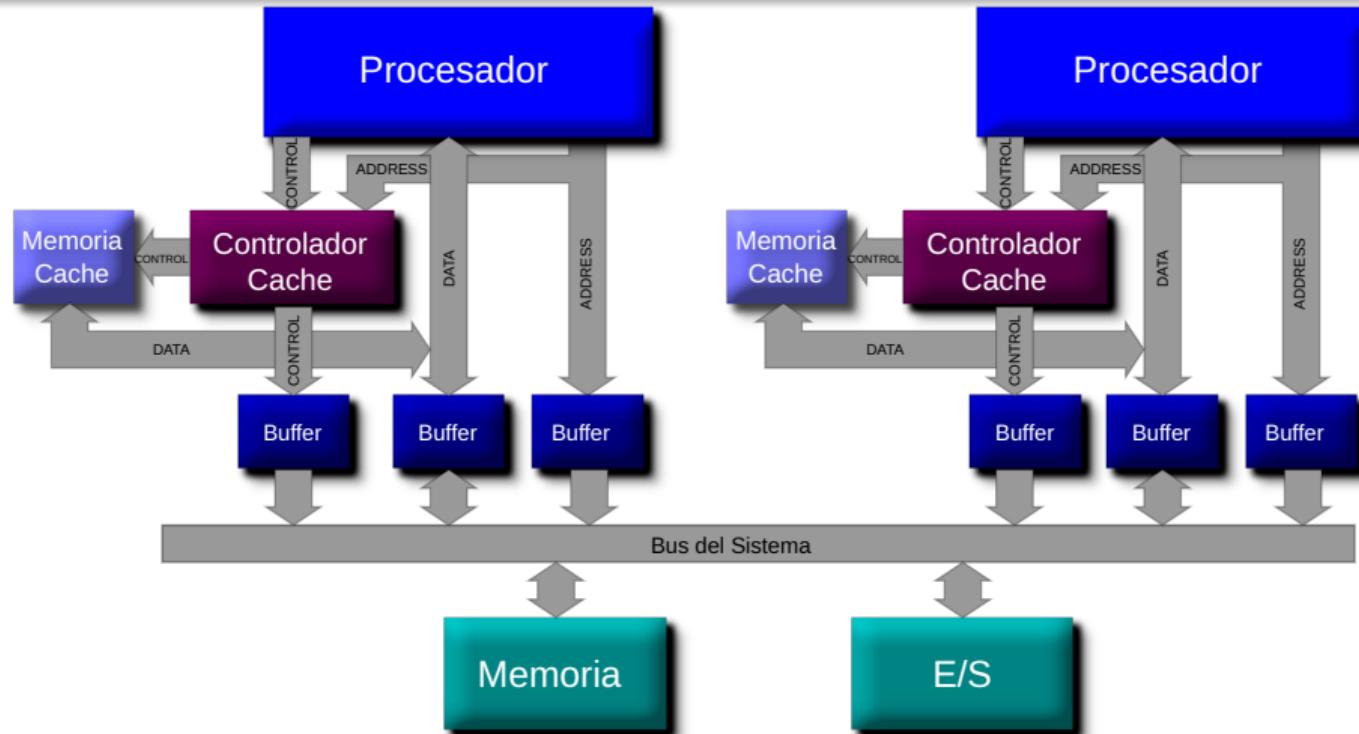
- Dos alternativas para implementar un protocolo de coherencia:

**write invalidate** Asegura acceso exclusivo al procesador que va a escribir el dato, antes de que la escritura se lleve a cabo. El resto debe poder detectar la escritura e invalidar la línea que contiene el dato en caso de tenerla en su Cache. Garantiza que no existe otra copia del dato una vez escrito éste en el cache destino.

**write update** También conocido como **write broadcast**, ya que actualiza la línea en todos los Caches que la tengan consumiendo bastante ancho de banda. La ventaja es que asegura menos Miss Rate

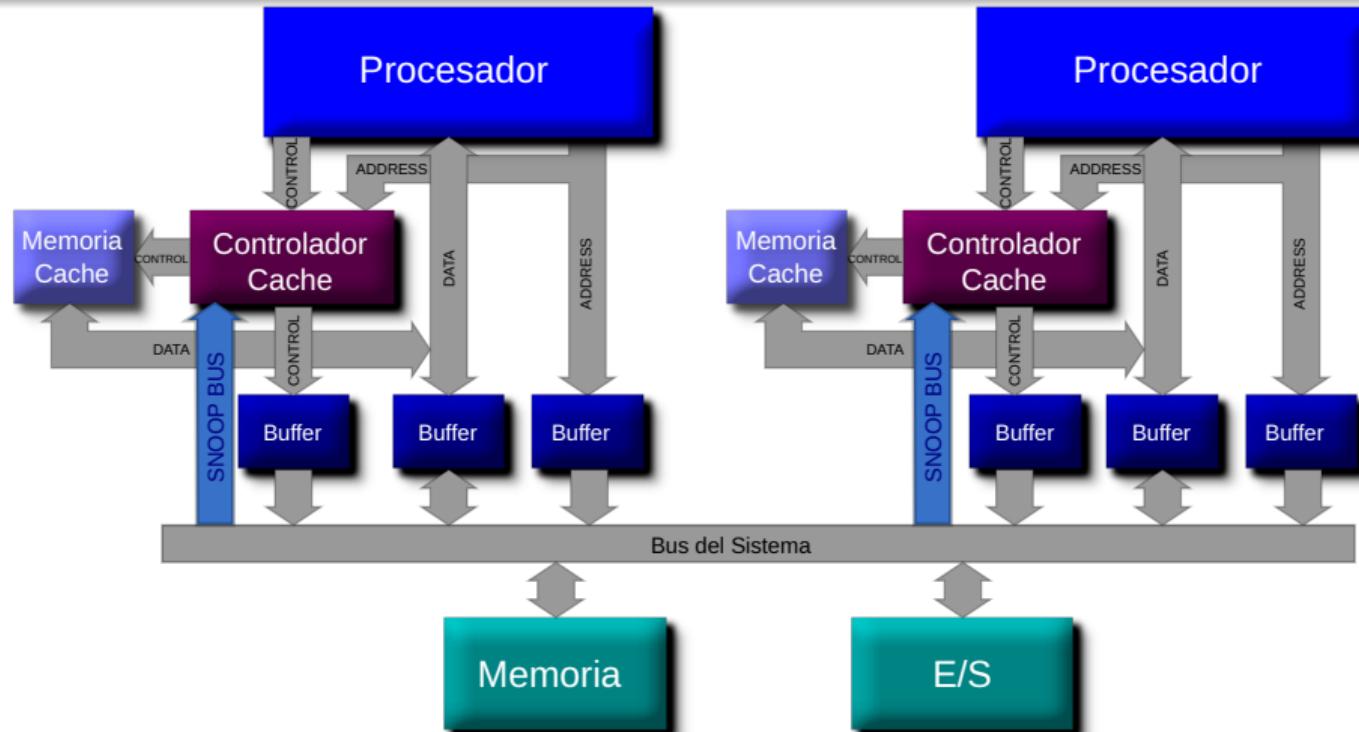
- **write invalidate** es por lejos el mas aplicado, en especial porque al crecer el número de procesadores, el ahorro de ancho de banda en el bus es evidente.

# Solución: hardware adicional: El Snoop bus



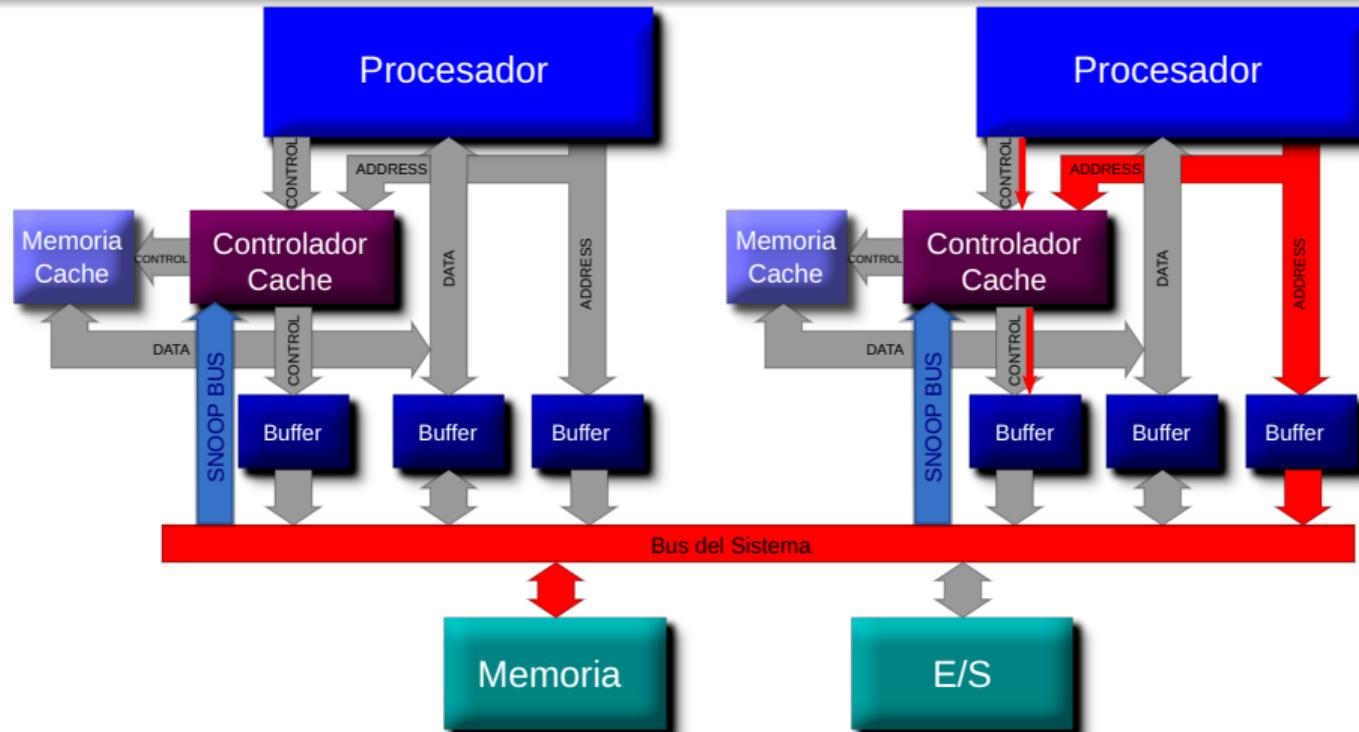
No es necesario implementar un Bus específico. La información que necesita el Controlador Cache es la Dirección de Memoria puesta en el bus, y si ésta se va a leer o escribir. Esta circula por el bus del sistema.

# Solución: hardware adicional: El Snoop bus



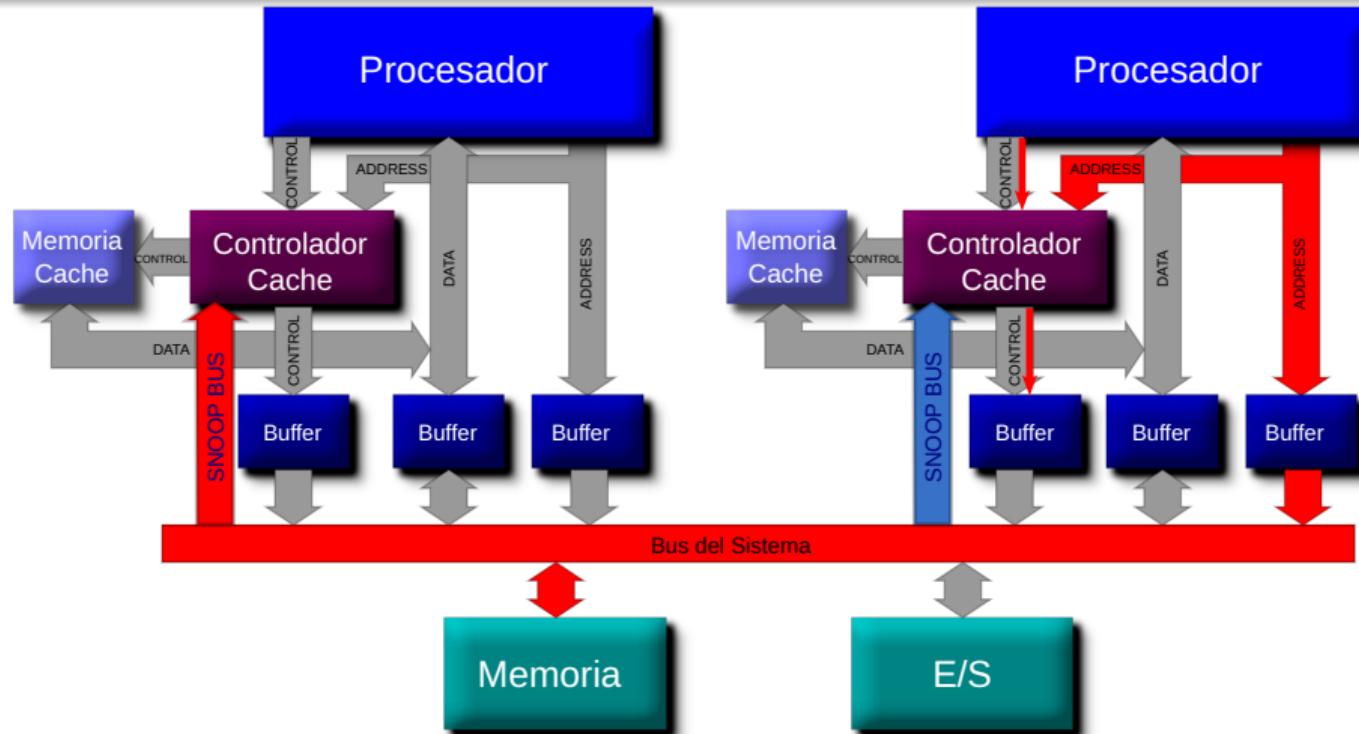
El Snoop Bus es entonces un conjunto de líneas entrantes a cada Controlador Cache con el cual éste “espía” lo que hacen los demás procesadores en particular con la memoria.

# Solución: hardware adicional: El Snooper bus



Si un procesador necesita un dato y no esta en su Cache (Cache Miss), genera el acceso hacia la jerarquía inferior. Si fue un Hit, lo resuelve desde su Cache. Es decir, baja por la jerarquía solo por Cache Miss

# Solución: hardware adicional: El Snoop bus



El otro Controlador Cache detecta el acceso mediante su Snoop Bus, y determina si lo tiene en su Cache. Si lo tiene (Cache Hit) activa el protocolo de Coherencia. De otro modo simplemente la descarta.

# Coherencia Write Invalidate - Snoop Bus

# Coherencia Write Invalidate - Snoop Bus

- En los multiprocesadores de chips múltiples más antiguos como en el gráfico anterior, el Snoop Bus utilizado para la coherencia es el bus de acceso a la memoria compartida.

## Coherencia Write Invalidate - Snoop Bus

- En los multiprocesadores de chips múltiples más antiguos como en el gráfico anterior, el Snoop Bus utilizado para la coherencia es el bus de acceso a la memoria compartida.
- En un multicore de un solo chip como el Intel Core-i7, puede conectarse el Snoop Bus a la conexión entre las Caches privadas (L1 y L2) y la Cache compartida (L3).

## Coherencia Write Invalidate - Snoop Bus

- En los multiprocesadores de chips múltiples más antiguos como en el gráfico anterior, el Snoop Bus utilizado para la coherencia es el bus de acceso a la memoria compartida.
- En un multicore de un solo chip como el Intel Core-i7, puede conectarse el Snoop Bus a la conexión entre las Caches privadas (L1 y L2) y la Cache compartida (L3).
- Cuando se produce una escritura en una línea compartida, el procesador que escribe debe tener acceso exclusivo al bus para colocar la dirección que los demás deben invalidar una vez leída desde su Snoop Bus.

# Coherencia Write Invalidate - Arbitración del Bus

# Coherencia Write Invalidate - Arbitración del Bus

- Si dos procesadores intentan escribir diferentes líneas compartidas al mismo tiempo, sus accesos al bus para invalidación se serializarán mediante la arbitraje del bus de cada sistema.

## Coherencia Write Invalidate - Arbitración del Bus

- Si dos procesadores intentan escribir diferentes líneas compartidas al mismo tiempo, sus accesos al bus para invalidación se serializarán mediante la arbitraje del bus de cada sistema.
- Si dos procesadores intentaran escribir la misma línea, la serialización impuesta por la arbitraje del bus también serializará sus escrituras.

## Coherencia Write Invalidate - Arbitración del Bus

- Si dos procesadores intentan escribir diferentes líneas compartidas al mismo tiempo, sus accesos al bus para invalidación se serializarán mediante la arbitraje del bus de cada sistema.
- Si dos procesadores intentaran escribir la misma línea, la serialización impuesta por la arbitraje del bus también serializará sus escrituras.
- Una consecuencia de este esquema es que una escritura en un elemento de datos compartido no puede completarse hasta que se obtenga acceso al bus.

## Coherencia Write Invalidate - Arbitración del Bus

- Si dos procesadores intentan escribir diferentes líneas compartidas al mismo tiempo, sus accesos al bus para invalidación se serializarán mediante la arbitraje del bus de cada sistema.
- Si dos procesadores intentaran escribir la misma línea, la serialización impuesta por la arbitraje del bus también serializará sus escrituras.
- Una consecuencia de este esquema es que una escritura en un elemento de datos compartido no puede completarse hasta que se obtenga acceso al bus.
- Todos los esquemas de coherencia requieren algún método de serialización de accesos a la misma línea de caché, ya sea serializando el acceso al Bus o a otra estructura compartida (Cache Directory por ejemplo).

# Coherencia Write Invalidate - Cache Search

## Coherencia Write Invalidate - Cache Search

- Además de invalidar las copias pendientes de la línea que está siendo escrita en el Cache, también se debe localizar un ítem de cuando se produce un Cache Miss.

## Coherencia Write Invalidate - Cache Search

- Además de invalidar las copias pendientes de la línea que está siendo escrita en el Cache, también se debe localizar un ítem de cuando se produce un Cache Miss.
- En un Cache ***write-through***, todas las líneas escritas se envían a la memoria siempre.

## Coherencia Write Invalidate - Cache Search

- Además de invalidar las copias pendientes de la línea que está siendo escrita en el Cache, también se debe localizar un ítem de cuando se produce un Cache Miss.
- En un Cache ***write-through***, todas las líneas escritas se envían a la memoria siempre.
- Siempre se puede obtener el valor más reciente de un elemento de datos en cualquier punto de la jerarquía.

## Coherencia Write Invalidate - Cache Search

- Además de invalidar las copias pendientes de la línea que está siendo escrita en el Cache, también se debe localizar un ítem de cuando se produce un Cache Miss.
- En un Cache ***write-through***, todas las líneas escritas se envían a la memoria siempre.
- Siempre se puede obtener el valor más reciente de un elemento de datos en cualquier punto de la jerarquía.
- Los buffers de escritura deben tratarse como entradas de Cache adicionales, ya que puede estar allí el dato modificado esperando para ser copiado al siguiente nivel jerárquico.

## Coherencia Write Invalidate - Cache Search

- Además de invalidar las copias pendientes de la línea que está siendo escrita en el Cache, también se debe localizar un ítem de cuando se produce un Cache Miss.
- En un Cache ***write-through***, todas las líneas escritas se envían a la memoria siempre.
- Siempre se puede obtener el valor más reciente de un elemento de datos en cualquier punto de la jerarquía.
- Los buffers de escritura deben tratarse como entradas de Cache adicionales, ya que puede estar allí el dato modificado esperando para ser copiado al siguiente nivel jerárquico.
- Para un Cache ***write-back***, el valor del dato más reciente puede estar en un Cache privado, en vez de en el Cache compartido, o en la Memoria Principal.

## Coherencia Write Invalidate - Cache Search

- Además de invalidar las copias pendientes de la línea que está siendo escrita en el Cache, también se debe localizar un ítem de cuando se produce un Cache Miss.
- En un Cache ***write-through***, todas las líneas escritas se envían a la memoria siempre.
- Siempre se puede obtener el valor más reciente de un elemento de datos en cualquier punto de la jerarquía.
- Los buffers de escritura deben tratarse como entradas de Cache adicionales, ya que puede estar allí el dato modificado esperando para ser copiado al siguiente nivel jerárquico.
- Para un Cache ***write-back***, el valor del dato más reciente puede estar en un Cache privado, en vez de en el Cache compartido, o en la Memoria Principal.
- Por eso los Cache ***write-back*** usan el mismo esquema de Snooping tanto para Cache Miss como para escrituras.

# Coherencia Write Invalidate - Cache Search

# Coherencia Write Invalidate - Cache Search

- Si un Cache tiene una copia Dirty de la línea solicitada, el Cache Controller habilita la lectura de esta línea desde el Cache y cancela su acceso a la Memoria Principal (o al Cache L3).

## Coherencia Write Invalidate - Cache Search

- Si un Cache tiene una copia Dirty de la línea solicitada, el Cache Controller habilita la lectura de esta línea desde el Cache y cancela su acceso a la Memoria Principal (o al Cache L3).
- Debido a que las Cache **write-back** emplean el mínimo ancho de banda posible de memoria, pueden soportar una mayor cantidad de procesadores más rápidos.

## Coherencia Write Invalidate - Cache Search

- Si un Cache tiene una copia Dirty de la línea solicitada, el Cache Controller habilita la lectura de esta línea desde el Cache y cancela su acceso a la Memoria Principal (o al Cache L3).
- Debido a que las Cache ***write-back*** emplean el mínimo ancho de banda posible de memoria, pueden soportar una mayor cantidad de procesadores más rápidos.
- Como resultado, todos los procesadores multicore utilizan ***write-back*** en los niveles más externos del Cache, por lo que nos concentraremos en el análisis de coherencia en Caches ***write-back***.

## Coherencia Write Invalidate - Cache Search

- Si un Cache tiene una copia Dirty de la línea solicitada, el Cache Controller habilita la lectura de esta línea desde el Cache y cancela su acceso a la Memoria Principal (o al Cache L3).
- Debido a que las Cache ***write-back*** emplean el mínimo ancho de banda posible de memoria, pueden soportar una mayor cantidad de procesadores más rápidos.
- Como resultado, todos los procesadores multicore utilizan ***write-back*** en los niveles más externos del Cache, por lo que nos concentraremos en el análisis de coherencia en Caches ***write-back***.
- Los Tags normales del Cache, sirven para el proceso de Snoop, y el bit de válido para cada línea hace trivial la implementación de la invalidación.

## Coherencia Write Invalidate - Cache Search

- Si un Cache tiene una copia Dirty de la línea solicitada, el Cache Controller habilita la lectura de esta línea desde el Cache y cancela su acceso a la Memoria Principal (o al Cache L3).
- Debido a que las Cache ***write-back*** emplean el mínimo ancho de banda posible de memoria, pueden soportar una mayor cantidad de procesadores más rápidos.
- Como resultado, todos los procesadores multicore utilizan ***write-back*** en los niveles más externos del Cache, por lo que nos concentraremos en el análisis de coherencia en Caches ***write-back***.
- Los Tags normales del Cache, sirven para el proceso de Snoop, y el bit de validez para cada línea hace trivial la implementación de la invalidación.
- Los Read Miss, ya sean generados por una invalidación o por algún otro evento, también son sencillos porque simplemente se basan en el Snoop.

# Coherencia Write Invalidate - Cache Write

## Coherencia Write Invalidate - Cache Write

- Para escrituras, necesitamos saber si hay otras copias de la línea almacenadas en Cache porque, si no las hay, en un Cache **write-back** la escritura no necesita cursarse en el bus.

## Coherencia Write Invalidate - Cache Write

- Para escrituras, necesitamos saber si hay otras copias de la línea almacenadas en Cache porque, si no las hay, en un Cache **write-back** la escritura no necesita cursarse en el bus.
- De este modo se reduce tanto el tiempo de escritura para el procesador como el ancho de banda solicitado al Bus.

## Coherencia Write Invalidate - Cache Write

- Para escrituras, necesitamos saber si hay otras copias de la línea almacenadas en Cache porque, si no las hay, en un Cache **write-back** la escritura no necesita cursarse en el bus.
- De este modo se reduce tanto el tiempo de escritura para el procesador como el ancho de banda solicitado al Bus.
- Para saber si una línea de Cache se comparte o no, puede agregarse un bit de estado adicional asociado con cada línea.

## Coherencia Write Invalidate - Cache Write

- Para escrituras, necesitamos saber si hay otras copias de la línea almacenadas en Cache porque, si no las hay, en un Cache **write-back** la escritura no necesita cursarse en el bus.
- De este modo se reduce tanto el tiempo de escritura para el procesador como el ancho de banda solicitado al Bus.
- Para saber si una línea de Cache se comparte o no, puede agregarse un bit de estado adicional asociado con cada línea.
- Con un bit que indica si la línea se comparte, el Controlador Cache puede decidir si una escritura debe generar una invalidación.

## Coherencia Write Invalidate - Cache Write

- Para escrituras, necesitamos saber si hay otras copias de la línea almacenadas en Cache porque, si no las hay, en un Cache **write-back** la escritura no necesita cursarse en el bus.
- De este modo se reduce tanto el tiempo de escritura para el procesador como el ancho de banda solicitado al Bus.
- Para saber si una línea de Cache se comparte o no, puede agregarse un bit de estado adicional asociado con cada línea.
- Con un bit que indica si la línea se comparte, el Controlador Cache puede decidir si una escritura debe generar una invalidación.
- Cuando se produce una escritura en una línea cuyo estado es compartida, el Controlador Cache genera una invalidación en el Bus, genera el **write-back**, y marca la línea como Exclusiva.

## Coherencia Write Invalidate - Cache Write

- Para escrituras, necesitamos saber si hay otras copias de la línea almacenadas en Cache porque, si no las hay, en un Cache **write-back** la escritura no necesita cursarse en el bus.
- De este modo se reduce tanto el tiempo de escritura para el procesador como el ancho de banda solicitado al Bus.
- Para saber si una línea de Cache se comparte o no, puede agregarse un bit de estado adicional asociado con cada línea.
- Con un bit que indica si la línea se comparte, el Controlador Cache puede decidir si una escritura debe generar una invalidación.
- Cuando se produce una escritura en una línea cuyo estado es compartida, el Controlador Cache genera una invalidación en el Bus, genera el **write-back**, y marca la línea como Exclusiva.
- Ese core no enviará más invalidaciones al escribir esa línea.

## Coherencia Write Invalidate - Cache Write

- Para escrituras, necesitamos saber si hay otras copias de la línea almacenadas en Cache porque, si no las hay, en un Cache **write-back** la escritura no necesita cursarse en el bus.
- De este modo se reduce tanto el tiempo de escritura para el procesador como el ancho de banda solicitado al Bus.
- Para saber si una línea de Cache se comparte o no, puede agregarse un bit de estado adicional asociado con cada línea.
- Con un bit que indica si la línea se comparte, el Controlador Cache puede decidir si una escritura debe generar una invalidación.
- Cuando se produce una escritura en una línea cuyo estado es compartida, el Controlador Cache genera una invalidación en el Bus, genera el **write-back**, y marca la línea como Exclusiva.
- Ese core no enviará más invalidaciones al escribir esa línea.
- El Cache que tiene la única copia de una línea se denomina **owner** (propietario) de esa línea.

# Coherencia Write Invalidate - Cache Write

- Si otro procesador solicita más tarde este línea de Cache, el estado debe volver a compartir.

## Coherencia Write Invalidate - Cache Write

- Si otro procesador solicita más tarde este línea de Cache, el estado debe volver a compartir.
- Esta solicitud por el Bus responde a un Cache Miss en el procesador remoto. Y se detecta por el Snoop Bus el acceso a memoria.

## Coherencia Write Invalidate - Cache Write

- Si otro procesador solicita más tarde este línea de Cache, el estado debe volver a compartir.
- Esta solicitud por el Bus responde a un Cache Miss en el procesador remoto. Y se detecta por el Snoop Bus el acceso a memoria.
- Cada transacción de Bus obliga a comparar los Tags en el Cache, y podría interferir con los accesos del Procesador a la memoria Cache.

## Coherencia Write Invalidate - Cache Write

- Si otro procesador solicita más tarde este línea de Cache, el estado debe volver a compartir.
- Esta solicitud por el Bus responde a un Cache Miss en el procesador remoto. Y se detecta por el Snoop Bus el acceso a memoria.
- Cada transacción de Bus obliga a comparar los Tags en el Cache, y podría interferir con los accesos del Procesador a la memoria Cache.
- Una forma de reducir esta concurrencia es duplicar los Tags y dirigir los accesos de Snoop a los duplicados.

## Coherencia Write Invalidate - Cache Write

- Si otro procesador solicita más tarde este línea de Cache, el estado debe volver a compartirse.
- Esta solicitud por el Bus responde a un Cache Miss en el procesador remoto. Y se detecta por el Snoop Bus el acceso a memoria.
- Cada transacción de Bus obliga a comparar los Tags en el Cache, y podría interferir con los accesos del Procesador a la memoria Cache.
- Una forma de reducir esta concurrencia es duplicar los Tags y dirigir los accesos de Snoop a los duplicados.
- Otro enfoque es usar en el Cache L3 compartido un directorio que indique para cada línea, si se comparte y qué núcleos tienen copia.

## Coherencia Write Invalidate - Cache Write

- Si otro procesador solicita más tarde este línea de Cache, el estado debe volver a compartirse.
- Esta solicitud por el Bus responde a un Cache Miss en el procesador remoto. Y se detecta por el Snoop Bus el acceso a memoria.
- Cada transacción de Bus obliga a comparar los Tags en el Cache, y podría interferir con los accesos del Procesador a la memoria Cache.
- Una forma de reducir esta concurrencia es duplicar los Tags y dirigir los accesos de Snoop a los duplicados.
- Otro enfoque es usar en el Cache L3 compartido un directorio que indique para cada línea, si se comparte y qué núcleos tienen copia.
- Con esta información, las invalidaciones se pueden dirigir solo a aquellos cachés con copias de la línea de caché.

## Coherencia Write Invalidate - Cache Write

- Si otro procesador solicita más tarde este línea de Cache, el estado debe volver a compartida.
- Esta solicitud por el Bus responde a un Cache Miss en el procesador remoto. Y se detecta por el Snoop Bus el acceso a memoria.
- Cada transacción de Bus obliga a comparar los Tags en el Cache, y podría interferir con los accesos del Procesador a la memoria Cache.
- Una forma de reducir esta concurrencia es duplicar los Tags y dirigir los accesos de Snoop a los duplicados.
- Otro enfoque es usar en el Cache L3 compartido un directorio que indique para cada línea, si se comparte y qué núcleos tienen copia.
- Con esta información, las invalidaciones se pueden dirigir solo a aquellos cachés con copias de la línea de caché.
- Requiere que el Cache L3 siempre tenga una copia de cualquier elemento de datos de L1 o L2, (propiedad llamada inclusión).

# Implementación de Protocolos de coherencia

- El Snoop bus es un recurso útil para obtener información que permita tomar acciones para mantener la coherencia de acuerdo con la actividad que éste registra en el Bus del Sistema.

# Implementación de Protocolos de coherencia

- El Snoop bus es un recurso útil para obtener información que permita tomar acciones para mantener la coherencia de acuerdo con la actividad que éste registra en el Bus del Sistema.
- Para actuar de acuerdo con las políticas de escritura que adoptemos se requiere un conjunto de acciones a tomar para cada caso posible de actividad en el Bus (un protocolo) y hardware adicional, para implementarlas.

# Implementación de Protocolos de coherencia

- El Snoop bus es un recurso útil para obtener información que permita tomar acciones para mantener la coherencia de acuerdo con la actividad que éste registra en el Bus del Sistema.
- Para actuar de acuerdo con las políticas de escritura que adoptemos se requiere un conjunto de acciones a tomar para cada caso posible de actividad en el Bus (un protocolo) y hardware adicional, para implementarlas.
- Copy Back es el método menos demandante del bus del sistema, optimizando de esta manera su utilización, pero parece inapropiado cuando se trata de mantener coherentes los datos entre dos o mas caches.

# Implementación de Protocolos de coherencia

- El Snoop bus es un recurso útil para obtener información que permita tomar acciones para mantener la coherencia de acuerdo con la actividad que éste registra en el Bus del Sistema.
- Para actuar de acuerdo con las políticas de escritura que adoptemos se requiere un conjunto de acciones a tomar para cada caso posible de actividad en el Bus (un protocolo) y hardware adicional, para implementarlas.
- Copy Back es el método menos demandante del bus del sistema, optimizando de esta manera su utilización, pero parece inapropiado cuando se trata de mantener coherentes los datos entre dos o mas caches.
- Para poder utilizar este método de escritura siempre que sea posible y reemplazarlo solo cuando la misma dirección física está presente en por lo menos dos caches, se han desarrollado protocolos de coherencia.

# Implementación de Protocolos de Coherencia

- Se implementan con una máquina de estados finita en el Controlador Cache de cada Core, que cambia el estado de cada línea de acuerdo con, requerimientos del Procesador Local del Core o requerimientos desde el Bus.

# Implementación de Protocolos de Coherencia

- Se implementan con una máquina de estados finita en el Controlador Cache de cada Core, que cambia el estado de cada línea de acuerdo con, requerimientos del Procesador Local del Core o requerimientos desde el Bus.
- Podemos pensar en un controlador individual asociado con cada línea, que pueda proceder independientemente con las operaciones de Snoop o con las solicitudes de caché para diferentes líneas.

# Implementación de Protocolos de Coherencia

- Se implementan con una máquina de estados finita en el Controlador Cache de cada Core, que cambia el estado de cada línea de acuerdo con, requerimientos del Procesador Local del Core o requerimientos desde el Bus.
- Podemos pensar en un controlador individual asociado con cada línea, que pueda proceder independientemente con las operaciones de Snoop o con las solicitudes de caché para diferentes líneas.
- En las implementaciones reales, un controlador soporta múltiples operaciones entrelazadas a líneas diferentes.

# Implementación de Protocolos de Coherencia

- Se implementan con una máquina de estados finita en el Controlador Cache de cada Core, que cambia el estado de cada línea de acuerdo con, requerimientos del Procesador Local del Core o requerimientos desde el Bus.
- Podemos pensar en un controlador individual asociado con cada línea, que pueda proceder independientemente con las operaciones de Snoop o con las solicitudes de caché para diferentes líneas.
- En las implementaciones reales, un controlador soporta múltiples operaciones entrelazadas a líneas diferentes.
- Una operación puede iniciarse antes de que se complete otra, aunque solo se permite un acceso a la memoria caché o un acceso al bus a la vez.

# Implementación de Protocolos de Coherencia

- Se implementan con una máquina de estados finita en el Controlador Cache de cada Core, que cambia el estado de cada línea de acuerdo con, requerimientos del Procesador Local del Core o requerimientos desde el Bus.
- Podemos pensar en un controlador individual asociado con cada línea, que pueda proceder independientemente con las operaciones de Snoop o con las solicitudes de caché para diferentes líneas.
- En las implementaciones reales, un controlador soporta múltiples operaciones entrelazadas a líneas diferentes.
- Una operación puede iniciarse antes de que se complete otra, aunque solo se permite un acceso a la memoria caché o un acceso al bus a la vez.
- Aunque nos referimos a un bus en las siguientes descripciones, cualquier red de interconexión que admita una transmisión a todos los Controladores Cache y a sus Caches privados asociados se puede utilizar para implementar Snoop.

# Implementación de Protocolos de Coherencia

- Cualquier protocolo de coherencia debe definir dos atributos para una línea de un Cache: Su Propiedad (Ownership) y su Coherencia.

# Implementación de Protocolos de Coherencia

- Cualquier protocolo de coherencia debe definir dos atributos para una línea de un Cache: Su Propiedad (Ownership) y su Coherencia.
- Ambos atributos definirán las acciones que el protocolo deberá implementar para cada actividad posible en el Bus del Sistema detectable por el Snoop Bus.

# Implementación de Protocolos de Coherencia

- Cualquier protocolo de coherencia debe definir dos atributos para una línea de un Cache: Su Propiedad (Ownership) y su Coherencia.
- Ambos atributos definirán las acciones que el protocolo deberá implementar para cada actividad posible en el Bus del Sistema detectable por el Snoop Bus.
- Un Cache es propietario de la línea cuando es el único que lo tiene. De otro modo, el Ownership es Compartido (**Shared**).

# Implementación de Protocolos de Coherencia

- Cualquier protocolo de coherencia debe definir dos atributos para una línea de un Cache: Su Propiedad (Ownership) y su Coherencia.
- Ambos atributos definirán las acciones que el protocolo deberá implementar para cada actividad posible en el Bus del Sistema detectable por el Snoop Bus.
- Un Cache es propietario de la línea cuando es el único que lo tiene. De otro modo, el Ownership es Compartido (**Shared**).
- Una Línea puede estar coherente con el resto de las copias disponibles en el sistema cuando todas las copias son iguales.

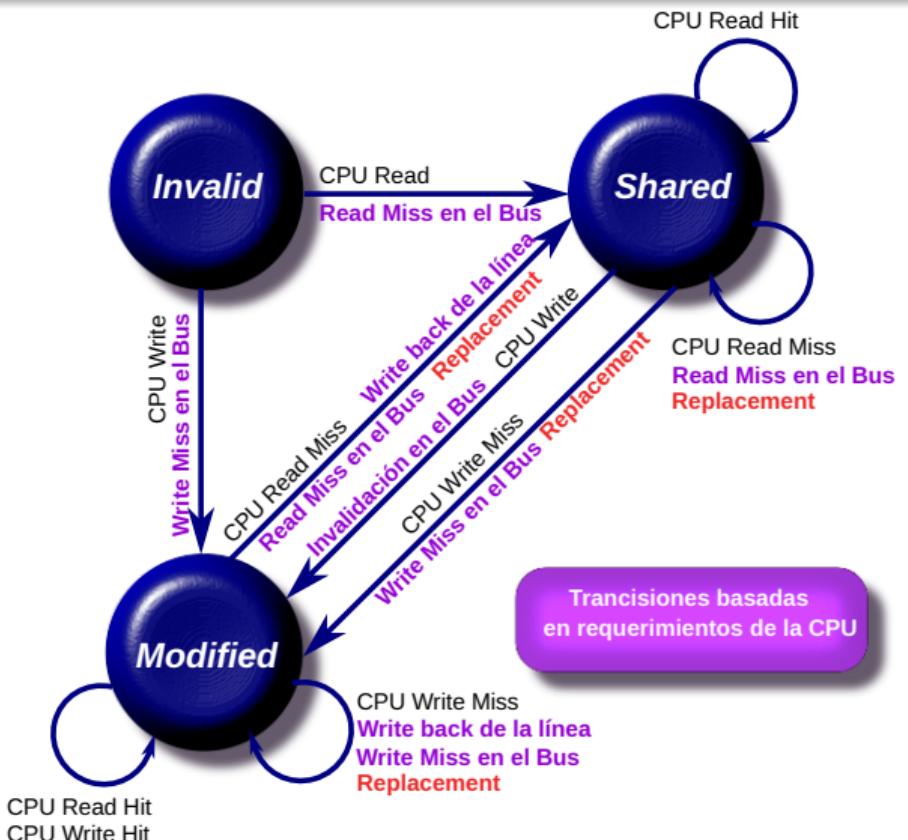
# Implementación de Protocolos de Coherencia

- Cualquier protocolo de coherencia debe definir dos atributos para una línea de un Cache: Su Propiedad (Ownership) y su Coherencia.
- Ambos atributos definirán las acciones que el protocolo deberá implementar para cada actividad posible en el Bus del Sistema detectable por el Snoop Bus.
- Un Cache es propietario de la línea cuando es el único que lo tiene. De otro modo, el Ownership es Compartido (**Shared**).
- Una Línea puede estar coherente con el resto de las copias disponibles en el sistema cuando todas las copias son iguales.
- Un alínea está incoherente si su copia está Dirty (**Modified**) en algún cache producto de alguna escritura.

# Implementación de Protocolos de Coherencia

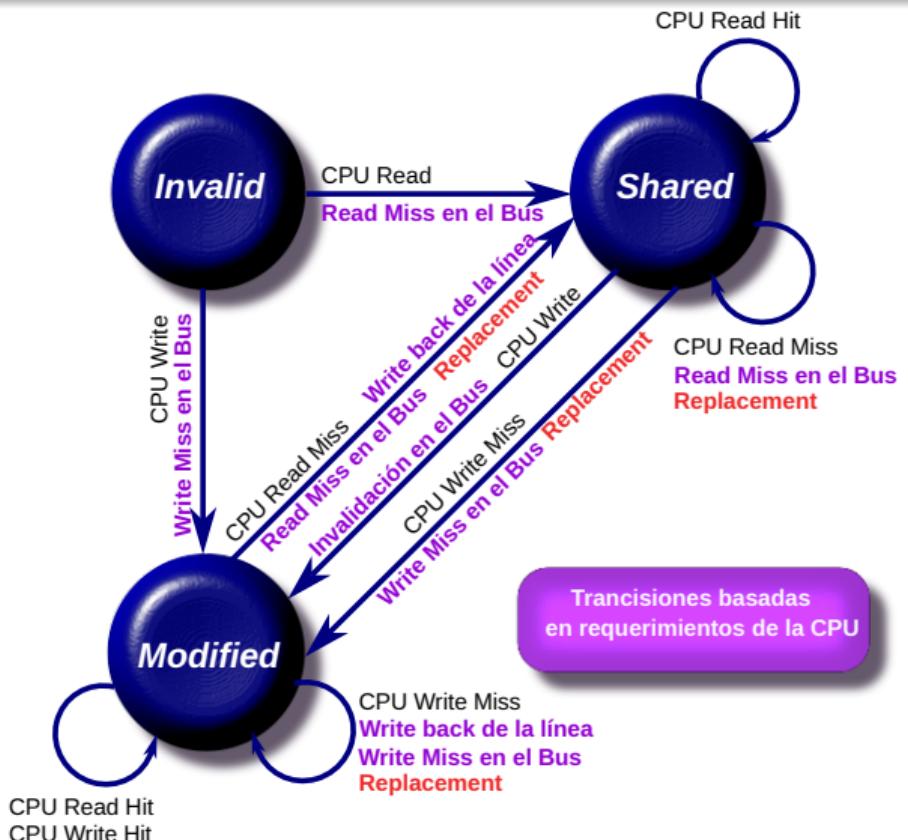
- Cualquier protocolo de coherencia debe definir dos atributos para una línea de un Cache: Su Propiedad (Ownership) y su Coherencia.
- Ambos atributos definirán las acciones que el protocolo deberá implementar para cada actividad posible en el Bus del Sistema detectable por el Snoop Bus.
- Un Cache es propietario de la línea cuando es el único que lo tiene. De otro modo, el Ownership es Compartido (**Shared**).
- Una Línea puede estar coherente con el resto de las copias disponibles en el sistema cuando todas las copias son iguales.
- Un alínea está incoherente si su copia está Dirty (**Modified**) en algún cache producto de alguna escritura.
- Esto último implica que se ha perdido la coherencia. Puede tolerarse esta situación siempre que el cache que tiene la copia **Modified**, tenga el Ownership de la línea.

# Implementación de Protocolos de Coherencia. MSI



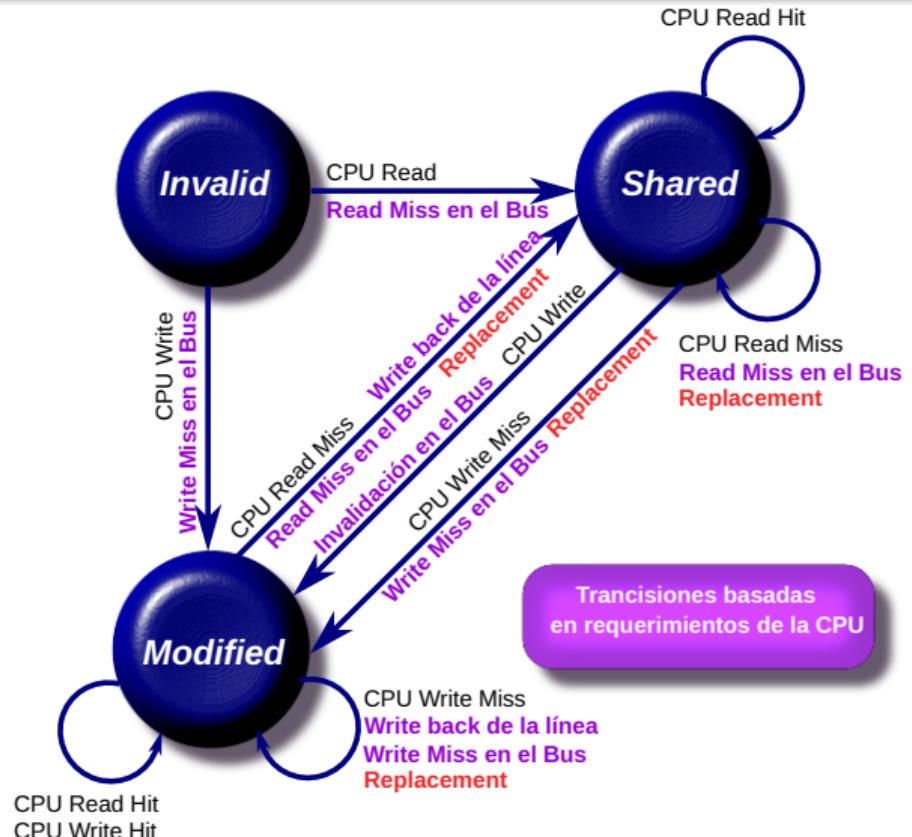
- El protocolo más simple que podemos implementar es de tres estados. Se llama MSI.

# Implementación de Protocolos de Coherencia. MSI



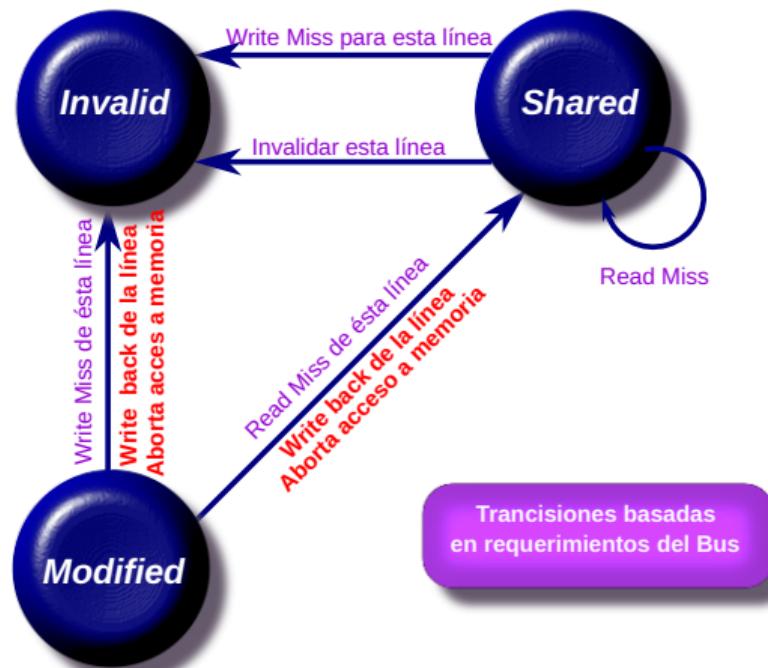
- El protocolo más simple que podemos implementar es de tres estados. Se llama MSI.
- **Shared** significa que la línea puede potencialmente estar compartida.

# Implementación de Protocolos de Coherencia. MSI



- El protocolo más simple que podemos implementar es de tres estados. Se llama MSI.
  - **Shared** significa que la línea puede potencialmente estar compartida.
  - **Modified** significa que la línea ha sido modificada en el Cache local. Como este protocolo es **write-invalidate**, este estado es además **Exclusivo**.

# Implementación de Protocolos de Coherencia. MSI



- Cuando se detecta un **Write Miss** o un **Invalidate** en el Bus, todos los Caches que poseen esa línea deben invalidarla.

# Implementación de Protocolos de Coherencia. MESI

- Agrega el estado Exclusive para disminuir la actividad en el Bus: Una línea en estado **E** puede escribirse sin generar invalidaciones en el Bus.

# Implementación de Protocolos de Coherencia. MESI

- Agrega el estado Exclusive para disminuir la actividad en el Bus: Una línea en estado **E** puede escribirse sin generar invalidaciones en el Bus.
- Aplica a cache L1 de datos y L2/L3 (para cache L1 de código solo Shared e Invalid)

# Implementación de Protocolos de Coherencia. MESI

- Agrega el estado Exclusive para disminuir la actividad en el Bus: Una línea en estado **E** puede escribirse sin generar invalidaciones en el Bus.
- Aplica a cache L1 de datos y L2/L3 (para cache L1 de código solo Shared e Invalid)

**M - Modified** Línea presente solamente en éste cache que varió respecto de su valor en memoria del sistema (dirty). Requiere write back hacia la memoria del sistema antes que otro procesador lea desde allí el dato (que ya no es válido).

# Implementación de Protocolos de Coherencia. MESI

- Agrega el estado Exclusive para disminuir la actividad en el Bus: Una línea en estado **E** puede escribirse sin generar invalidaciones en el Bus.
- Aplica a cache L1 de datos y L2/L3 (para cache L1 de código solo Shared e Invalid)

**M - Modified** Línea presente solamente en éste cache que varió respecto de su valor en memoria del sistema (dirty). Requiere write back hacia la memoria del sistema antes que otro procesador lea desde allí el dato (que ya no es válido).

**E – Exclusive** Línea presente solo en esta cache, que coincide con la copia en memoria principal (clean).

# Implementación de Protocolos de Coherencia. MESI

- Agrega el estado Exclusive para disminuir la actividad en el Bus: Una línea en estado **E** puede escribirse sin generar invalidaciones en el Bus.
- Aplica a cache L1 de datos y L2/L3 (para cache L1 de código solo Shared e Invalid)

**M - Modified** Línea presente solamente en éste cache que varió respecto de su valor en memoria del sistema (dirty). Requiere write back hacia la memoria del sistema antes que otro procesador lea desde allí el dato (que ya no es válido).

**E - Exclusive** Línea presente solo en esta cache, que coincide con la copia en memoria principal (clean).

**S - Shared** Línea del cache presente y **puede** estar almacenada en los caches de otros procesadores.

# Implementación de Protocolos de Coherencia. MESI

- Agrega el estado Exclusive para disminuir la actividad en el Bus: Una línea en estado **E** puede escribirse sin generar invalidaciones en el Bus.
- Aplica a cache L1 de datos y L2/L3 (para cache L1 de código solo Shared e Invalid)

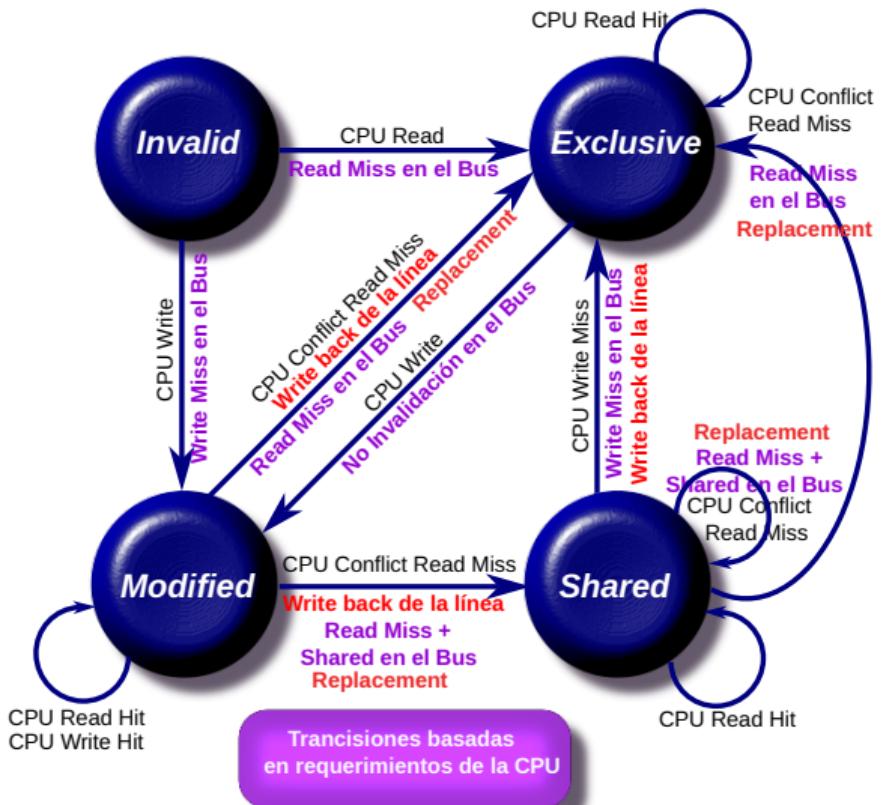
**M - Modified** Línea presente solamente en éste cache que varió respecto de su valor en memoria del sistema (dirty). Requiere write back hacia la memoria del sistema antes que otro procesador lea desde allí el dato (que ya no es válido).

**E - Exclusive** Línea presente solo en esta cache, que coincide con la copia en memoria principal (clean).

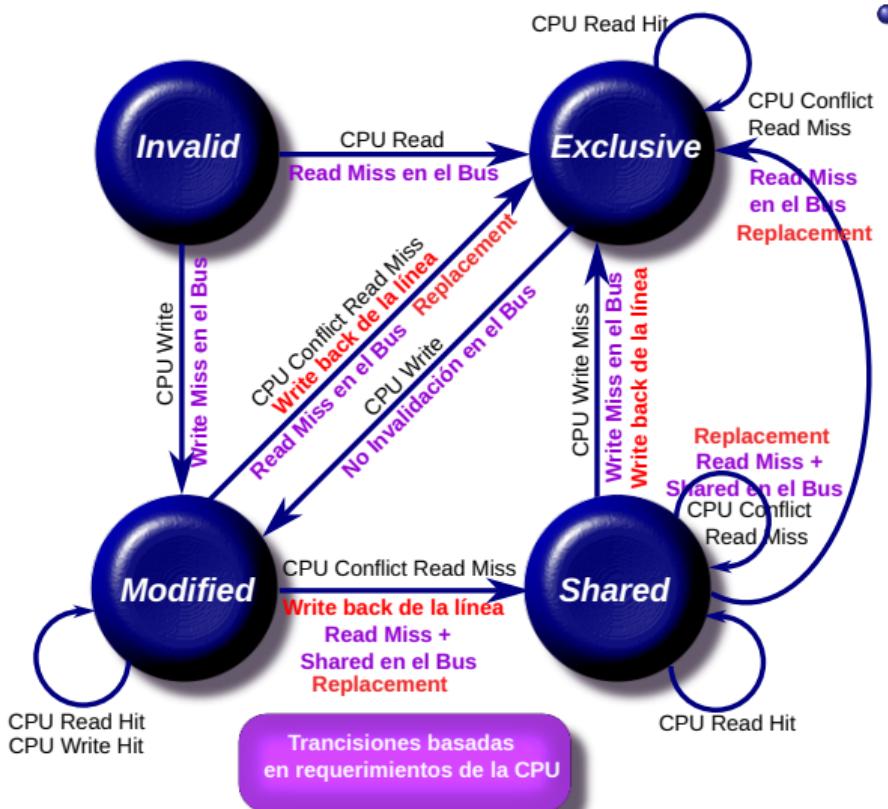
**S - Shared** Línea del cache presente y **puede** estar almacenada en los caches de otros procesadores.

**I - Invalid** Línea de cache no es válida.

## Protocolo MESI - Diagrama de estados

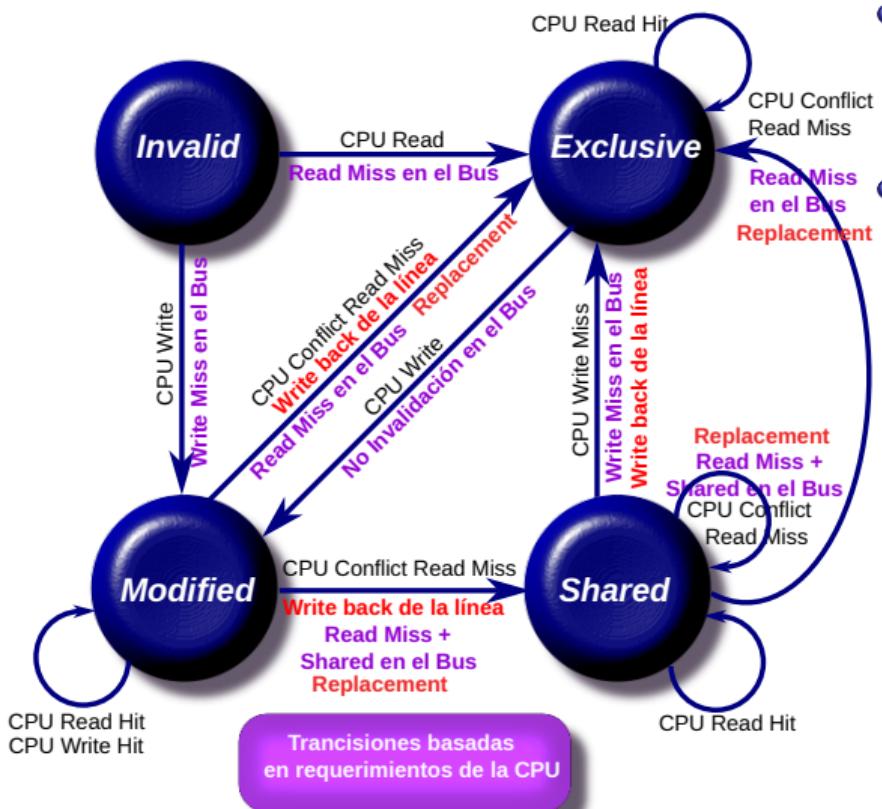


# Protocolo MESI - Diagrama de estados



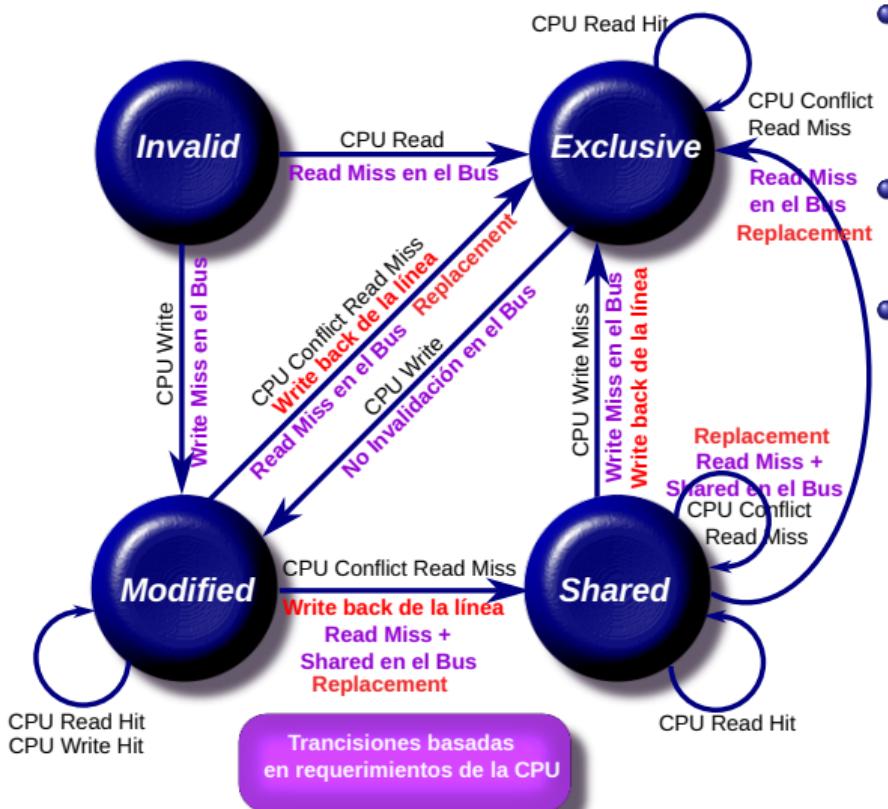
- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.

# Protocolo MESI - Diagrama de estados



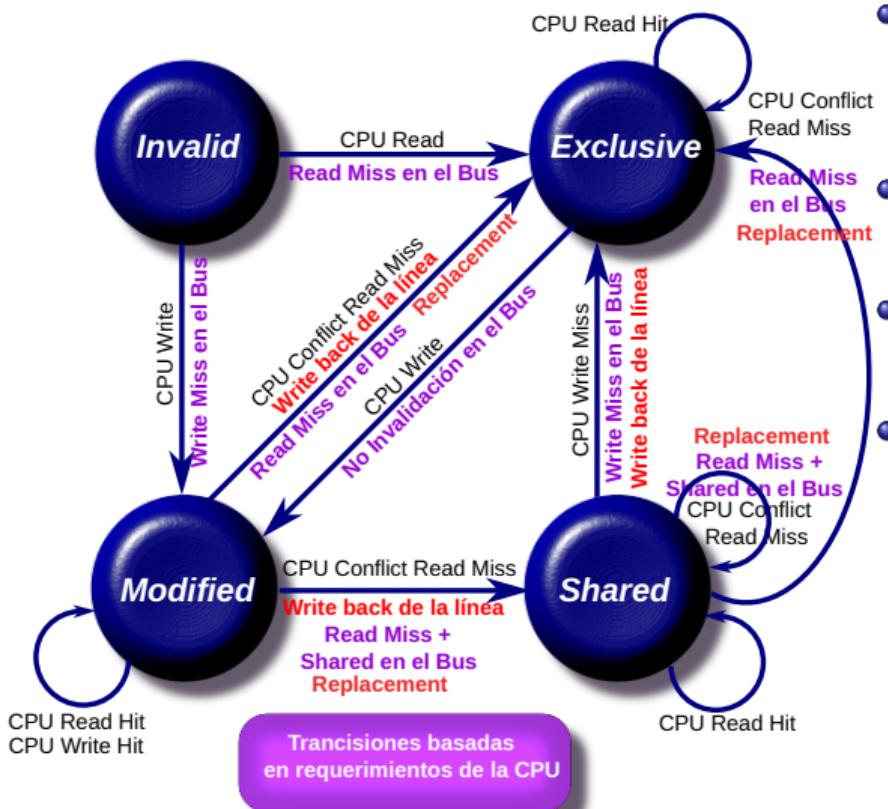
- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- La líneas inválidas en cada nivel del Cache se buscan en la jerarquía inferior inmediata.

# Protocolo MESI - Diagrama de estados



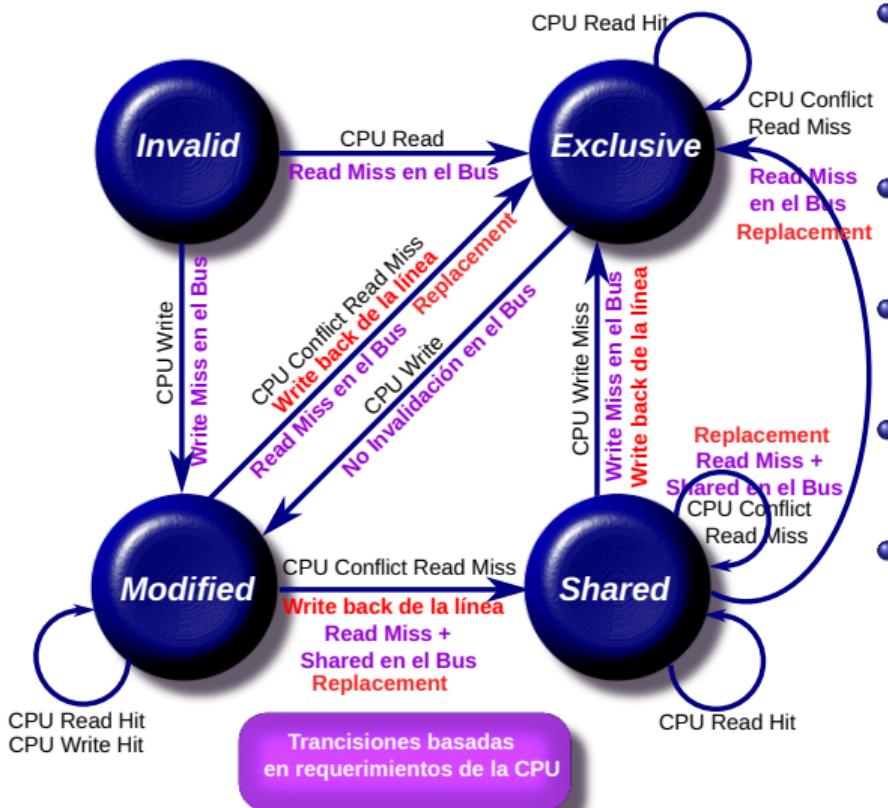
- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- La líneas inválidas en cada nivel del Cache se buscan en la jerarquía inferior inmediata.
- Una línea **Shared** o **Exclusive**, puede ser descartada y pasar a Inválida en cualquier momento.

# Protocolo MESI - Diagrama de estados



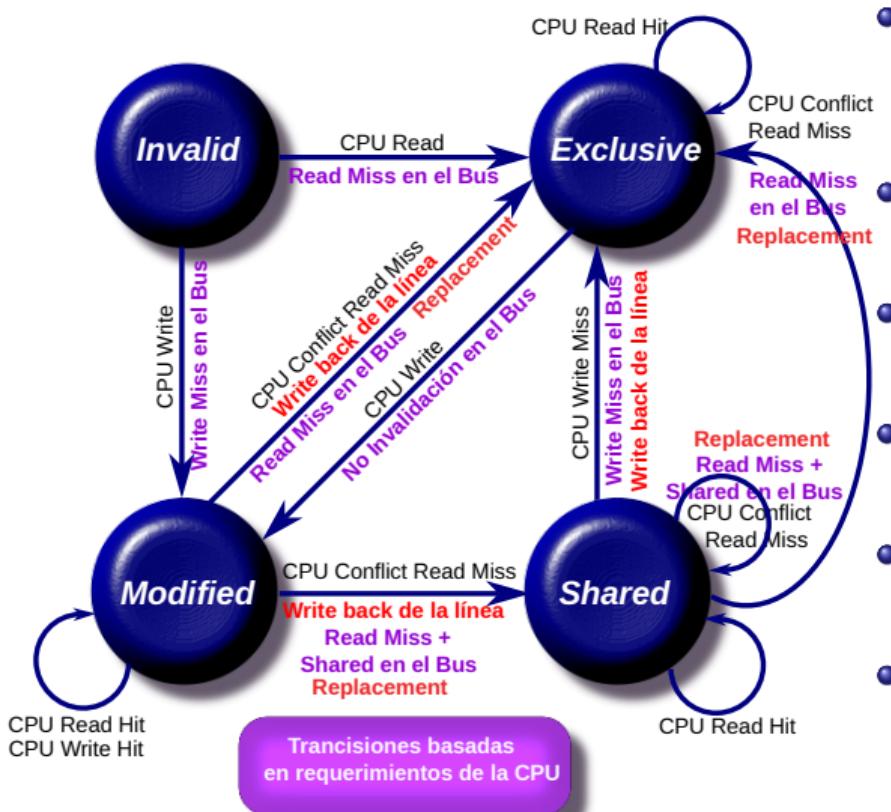
- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- La líneas inválidas en cada nivel del Cache se buscan en la jerarquía inferior inmediata.
- Una línea **Shared** o **Exclusive**, puede ser descartada y pasar a Inválida en cualquier momento.
- Una línea **Modified**, también, solo que en este caso se requiere Write Back previamente..

# Protocolo MESI - Diagrama de estados



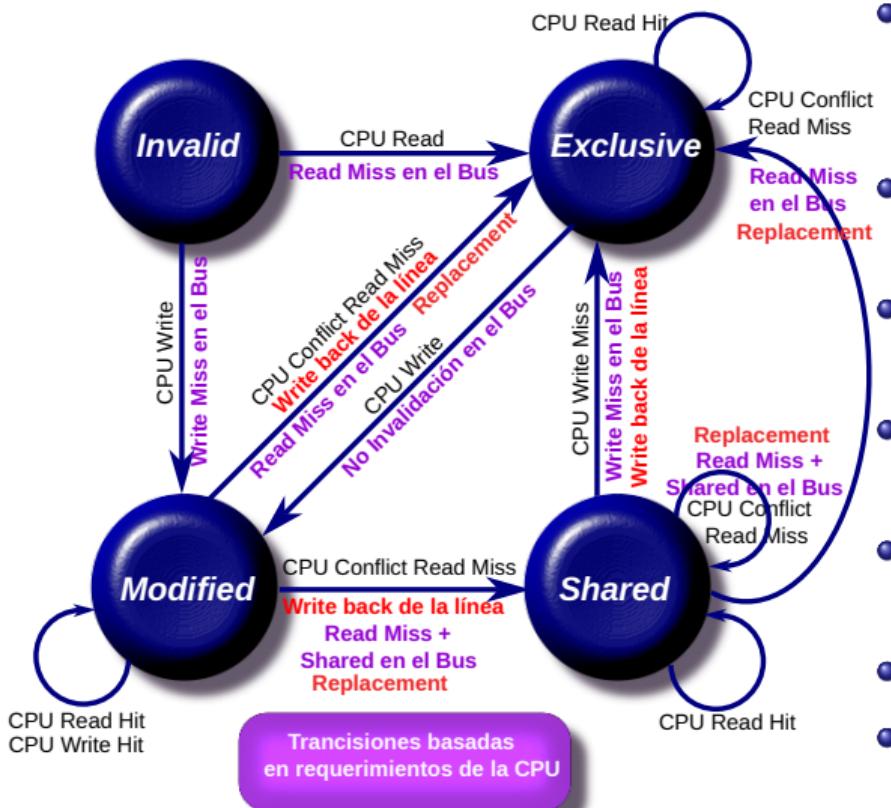
- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- La líneas inválidas en cada nivel del Cache se buscan en la jerarquía inferior inmediata.
- Una línea **Shared** o **Exclusive**, puede ser descartada y pasar a Inválida en cualquier momento.
- Una línea **Modified**, también, solo que en este caso se requiere Write Back previamente..
- Una línea **Modified** puede escribirse desde su CPU en cualquier momento.

# Protocolo MESI - Diagrama de estados



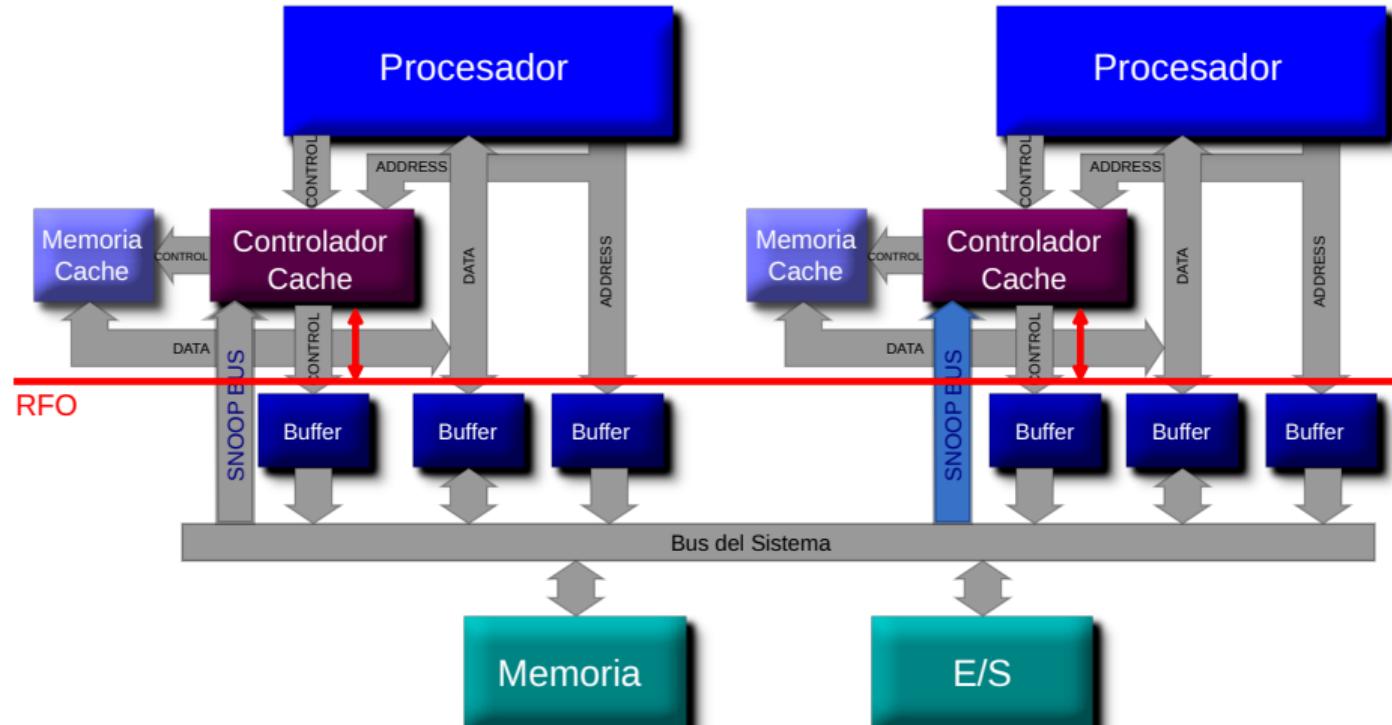
- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- La líneas inválidas en cada nivel del Cache se buscan en la jerarquía inferior inmediata.
- Una línea **Shared** o **Exclusive**, puede ser descartada y pasar a Inválida en cualquier momento.
- Una línea **Modified**, también, solo que en este caso se requiere Write Back previamente..
- Una línea **Modified** puede escribirse desde su CPU en cualquier momento.
- Una línea **Exclusive**, también pero pasará a **Modified**.

# Protocolo MESI - Diagrama de estados



- Todas las lecturas de líneas enviadas por el procesador se resuelven desde el cache, excepto si el estado de esa línea es **Invalid**.
- La líneas inválidas en cada nivel del Cache se buscan en la jerarquía inferior inmediata.
- Una línea **Shared** o **Exclusive**, puede ser descartada y pasar a Inválida en cualquier momento.
- Una línea **Modified**, también, solo que en este caso se requiere Write Back previamente..
- Una línea **Modified** puede escribirse desde su CPU en cualquier momento.
- Una línea **Exclusive**, también pero pasará a **Modified**.
- Para escribir una línea **Shared**, todas las demás caches que tienen esa línea la deben invalidar previamente.

# Protocolos de Coherencia: Señales



Para ello se emplea una operación broadcast denominada **Read For Ownership**. Suplementada por la señal homónima entre Controladores.

# Read For Ownership

- Los protocolos de coherencia combinan una escritura de una línea con un broadcast de invalidación al resto de los controladores.

## Read For Ownership

- Los protocolos de coherencia combinan una escritura de una línea con un broadcast de invalidación al resto de los controladores.
- Es enviada por el Cache que necesita escribir una línea en estado **Shared** o **Invalid**.

## Read For Ownership

- Los protocolos de coherencia combinan una escritura de una línea con un broadcast de invalidación al resto de los controladores.
- Es enviada por el Cache que necesita escribir una línea en estado **Shared** o **Invalid**.
- El resto de los caches que tienen esta línea almacenada la invalidan al recibir esta transacción.

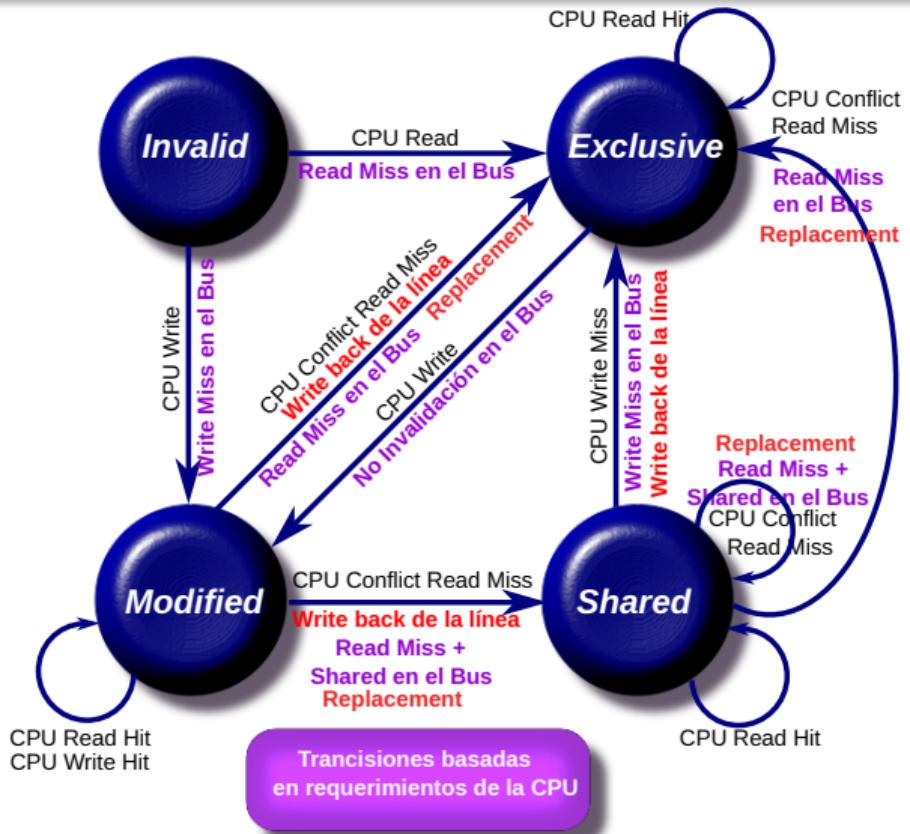
## Read For Ownership

- Los protocolos de coherencia combinan una escritura de una línea con un broadcast de invalidación al resto de los controladores.
- Es enviada por el Cache que necesita escribir una línea en estado **Shared** o **Invalid**.
- El resto de los caches que tienen esta línea almacenada la invalidan al recibir esta transacción.
- Desde el punto de vista del controlador cache es una lectura de la línea cacheada con toma del bus del sistema para escribir el contenido de la línea en el nivel jerárquico inferior (**Write Back**).

## Read For Ownership

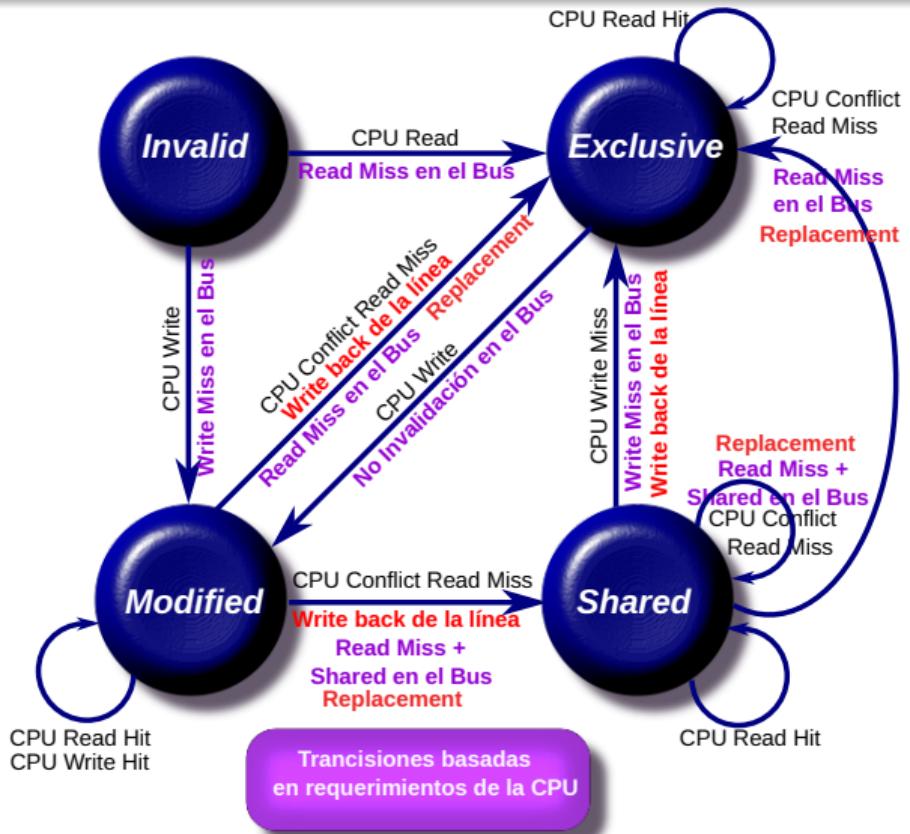
- Los protocolos de coherencia combinan una escritura de una línea con un broadcast de invalidación al resto de los controladores.
- Es enviada por el Cache que necesita escribir una línea en estado **Shared** o **Invalid**.
- El resto de los caches que tienen esta línea almacenada la invalidan al recibir esta transacción.
- Desde el punto de vista del controlador cache es una lectura de la línea cacheada con toma del bus del sistema para escribir el contenido de la línea en el nivel jerárquico inferior (**Write Back**).
- Es una operación exclusiva que fuerza al resto de los controladores que tienen esa línea almacenada a invalidarla.

# Protocolo MESI - Diagrama de estados



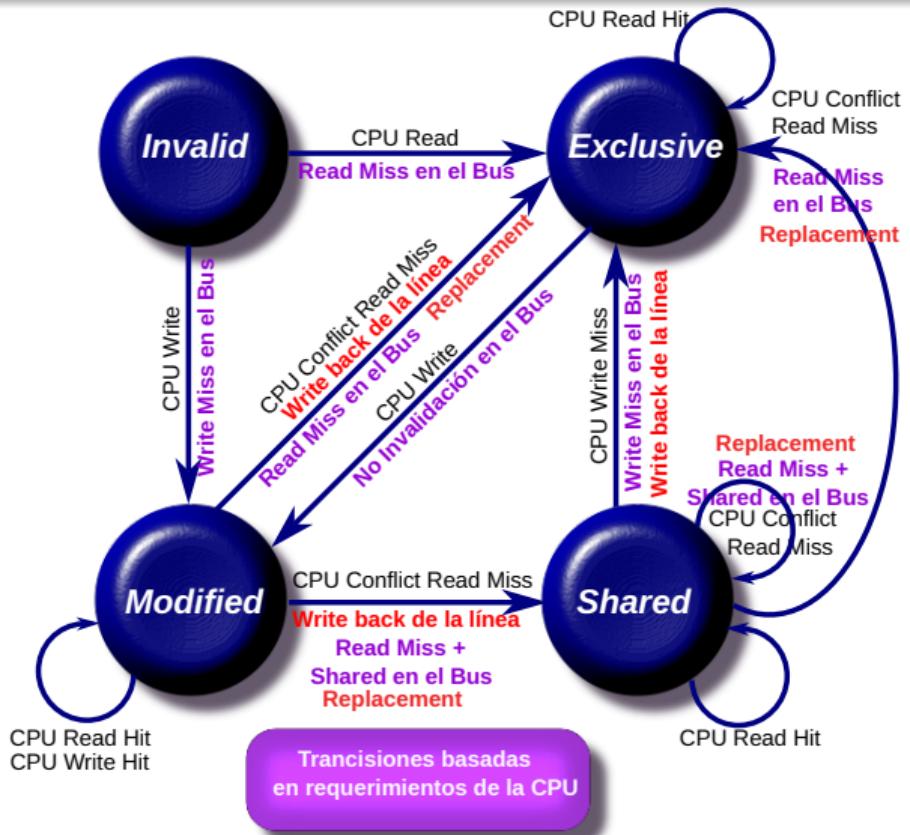
- **Modified** y **Exclusive** son estados precisos: el protocolo asegura que una línea **M** o **E** solo está en este cache (señalan ownership!).

# Protocolo MESI - Diagrama de estados



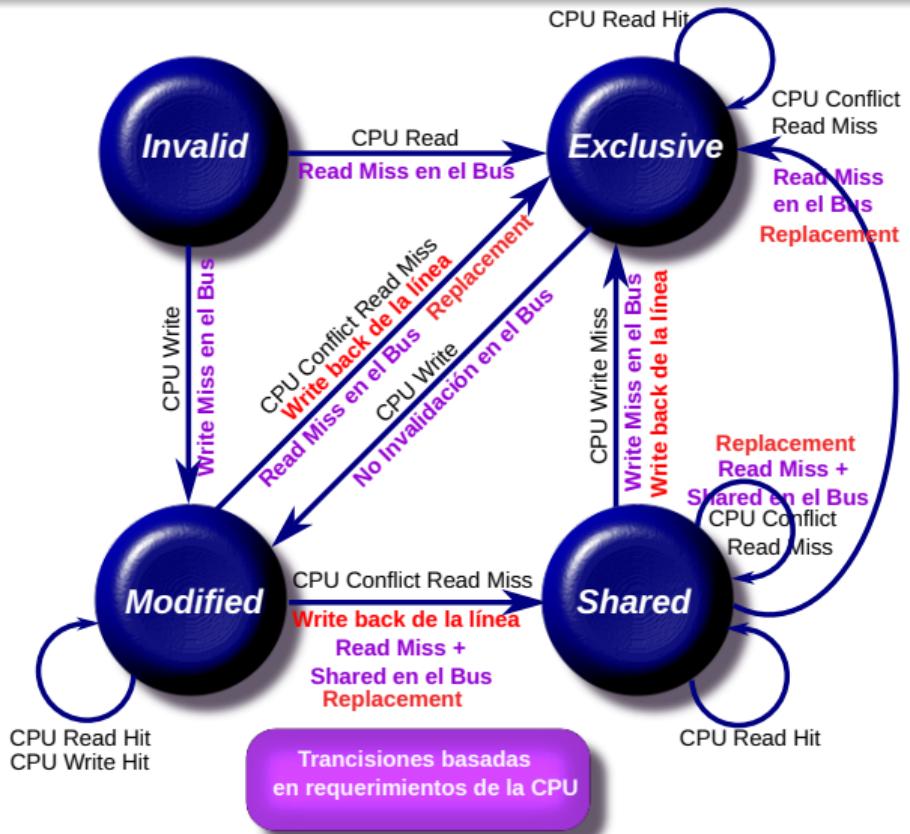
- **Modified** y **Exclusive** son estados precisos: el protocolo asegura que una línea **M** o **E** solo está en este cache (señalan ownership!).
- **Exclusive** permite optimizar el mínimo de transacciones en el bus ya que al ser escrito cambia a **Modified** pero no propaga el cambio.

## Protocolo MESI - Diagrama de estados



- **Modified** y **Exclusive** son estados precisos: el protocolo asegura que una línea **M** o **E** solo está en este cache (señalan ownership!).
  - **Exclusive** permite optimizar el mínimo de transacciones en el bus ya que al ser escrito cambia a **Modified** pero no propaga el cambio.
  - **Modified** acepta lectura y escritura sin usar el bus del sistema ni cambia de estado.

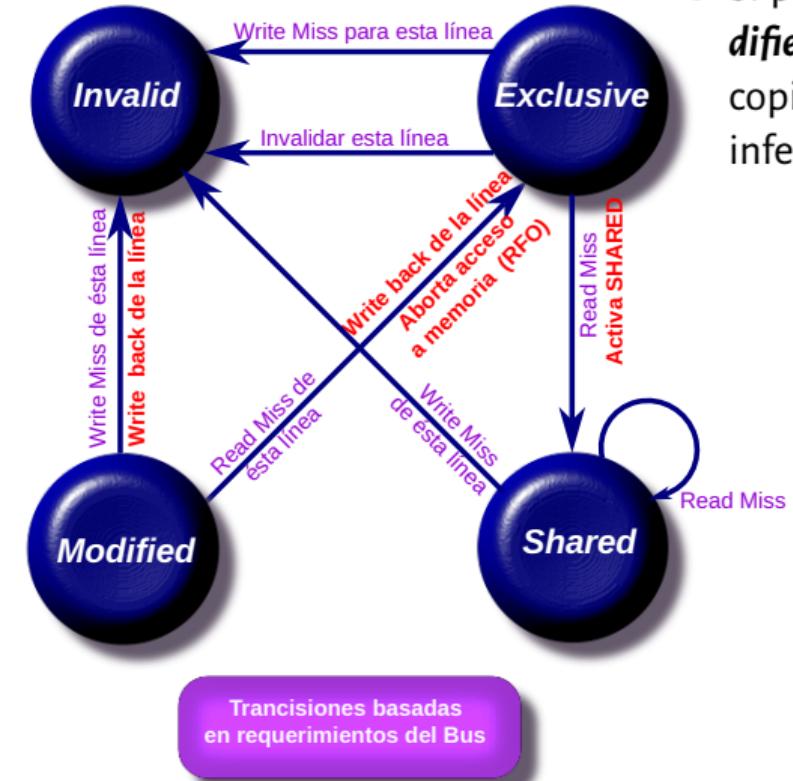
# Protocolo MESI - Diagrama de estados



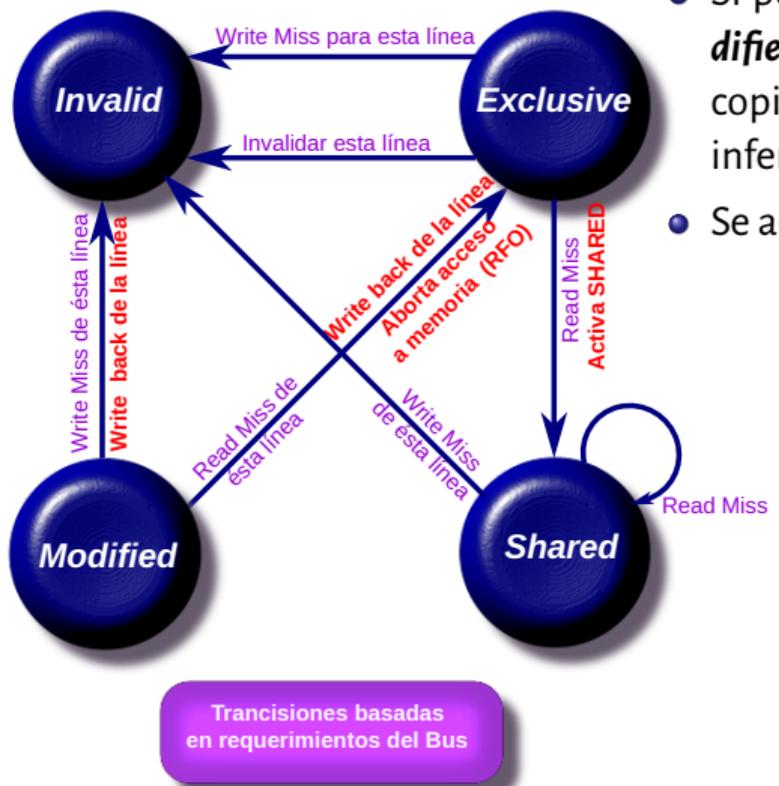
- **Modified** y **Exclusive** son estados precisos: el protocolo asegura que una línea **M** o **E** solo está en este cache (señalan ownership!).
- **Exclusive** permite optimizar el mínimo de transacciones en el bus ya que al ser escrito cambia a **Modified** pero no propaga el cambio.
- **Modified** acepta lectura y escritura sin usar el bus del sistema ni cambia de estado.
- **Shared** es un estado impreciso. Aunque otros caches descarten su copia de esta línea, esta acción no se informa, ni hay modo en que cada cache pueda llevar la cuenta de cuantos caches tienen la copia de una línea **Shared**. Por lo tanto nunca puede pasar a **Exclusive**.

# Protocolo MESI - Diagrama de estados

- Si por el Snoop Bus se detecta un Read Miss de una línea **Modified** en el cache local, el Controlador asume que tiene la única copia válida (Ownership), y que está incoherente con los niveles inferiores de la jerarquía (**Modified** y **Exclusive** usan copy back).



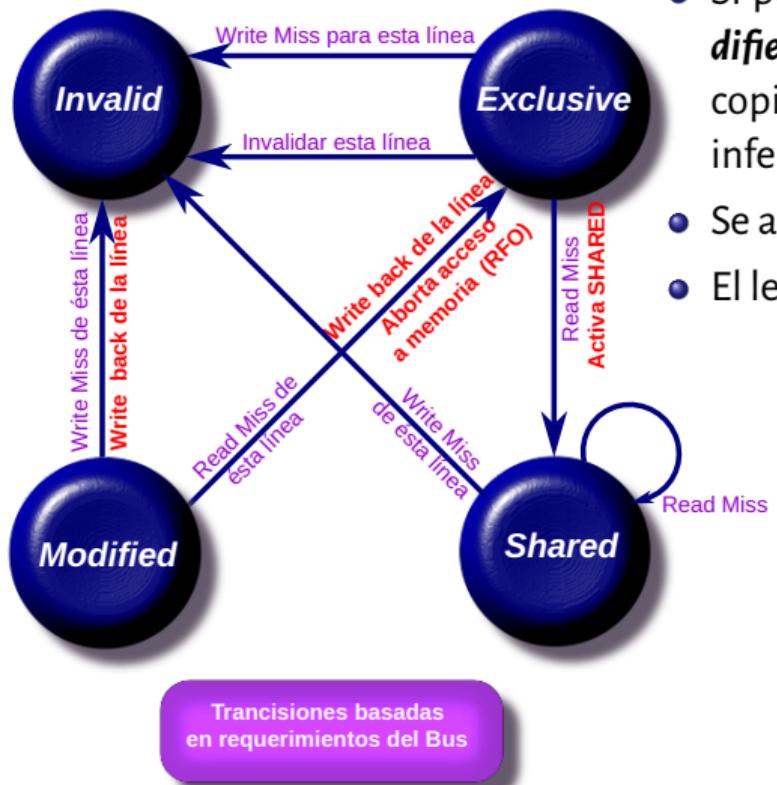
## Protocolo MESI - Diagrama de estados



- Si por el Snoop Bus se detecta un Read Miss de una línea **Modified** en el cache local, el Controlador asume que tiene la única copia válida (Ownership), y que está incoherente con los niveles inferiores de la jerarquía (**Modified** y **Exclusive** usan copy back).
  - Se activa RFO indicando al lector que el dato está incoherente.

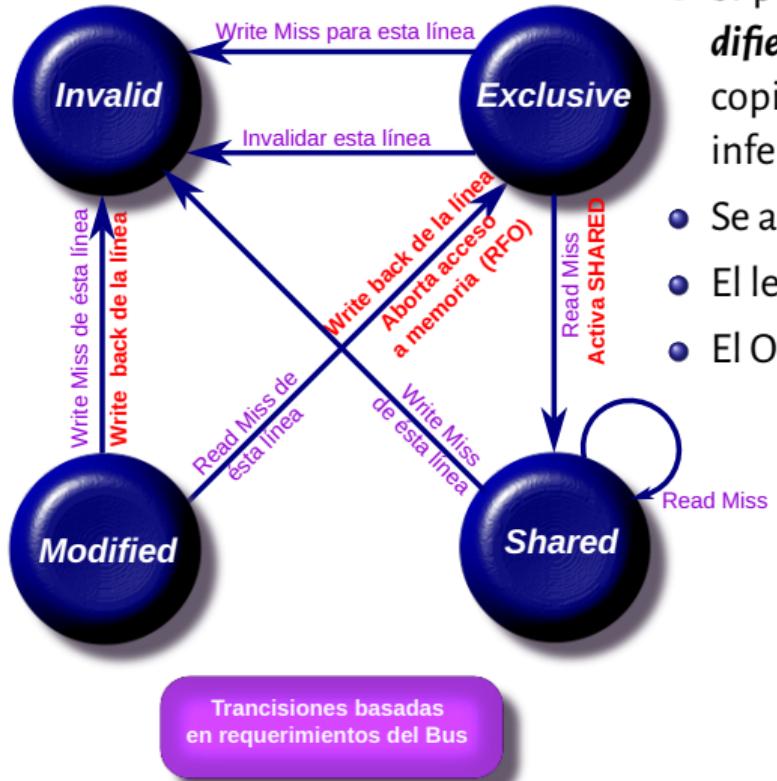
## Trancisiones basadas en requerimientos del Bus

# Protocolo MESI - Diagrama de estados



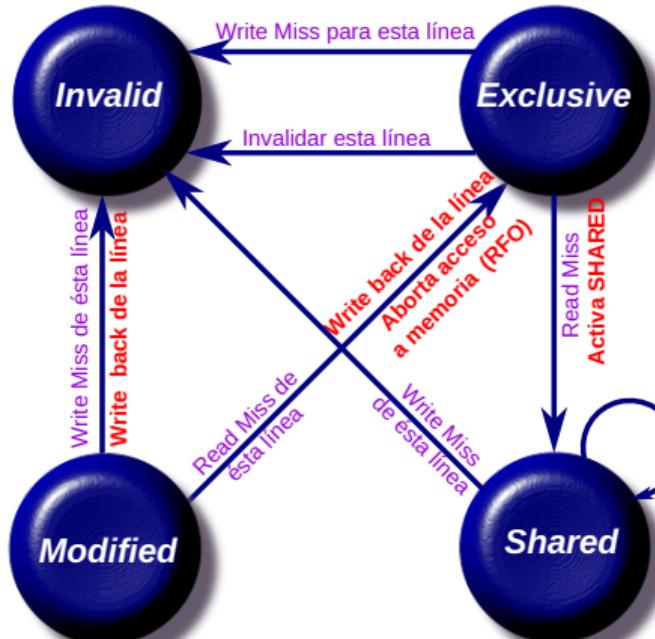
- Si por el Snoop Bus se detecta un Read Miss de una línea **Modified** en el cache local, el Controlador asume que tiene la única copia válida (Ownership), y que está incoherente con los niveles inferiores de la jerarquía (**Modified** y **Exclusive** usan copy back).
- Se activa RFO indicando al lector que el dato está incoherente.
- El lector aborta la lectura y pone sus Buses en Alta Impedancia.

# Protocolo MESI - Diagrama de estados



- Si por el Snoop Bus se detecta un Read Miss de una línea **Modified** en el cache local, el Controlador asume que tiene la única copia válida (Ownership), y que está incoherente con los niveles inferiores de la jerarquía (**Modified** y **Exclusive** usan copy back).
- Se activa RFO indicando al lector que el dato está incoherente.
- El lector aborta la lectura y pone sus Buses en Alta Impedancia.
- El Owner realiza el Copy Back de la línea.

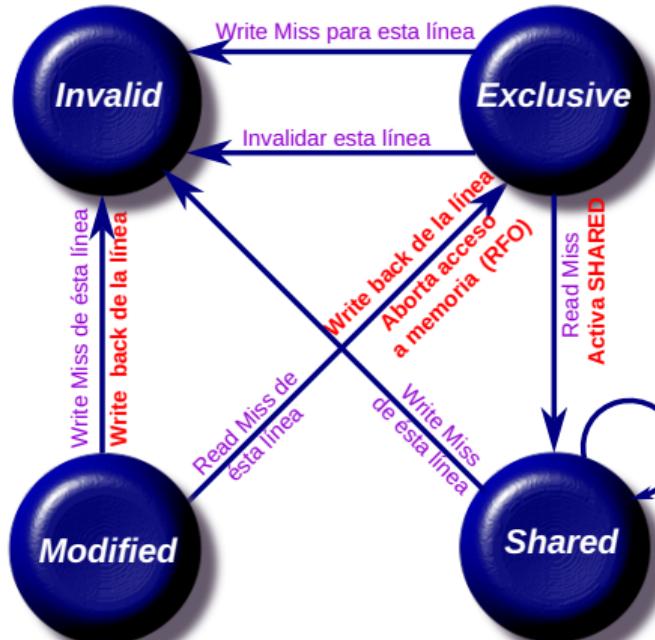
# Protocolo MESI - Diagrama de estados



Transiciones basadas  
en requerimientos del Bus

- Si por el Snoop Bus se detecta un Read Miss de una línea **Modified** en el cache local, el Controlador asume que tiene la única copia válida (Ownership), y que está incoherente con los niveles inferiores de la jerarquía (**Modified** y **Exclusive** usan copy back).
- Se activa RFO indicando al lector que el dato está incoherente.
- El lector aborta la lectura y pone sus Buses en Alta Impedancia.
- El Owner realiza el Copy Back de la línea.
- Ahora sigue teniendo la única copia de la línea, pero ésta es coherente con el resto de la jerarquía. La pone en estado en estado **Exclusive**, y desactiva RFO.

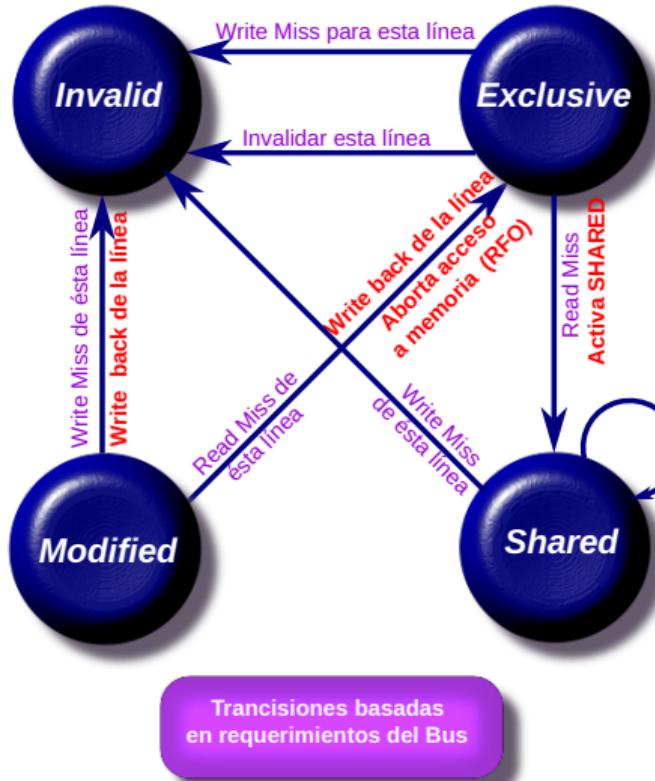
# Protocolo MESI - Diagrama de estados



Transiciones basadas  
en requerimientos del Bus

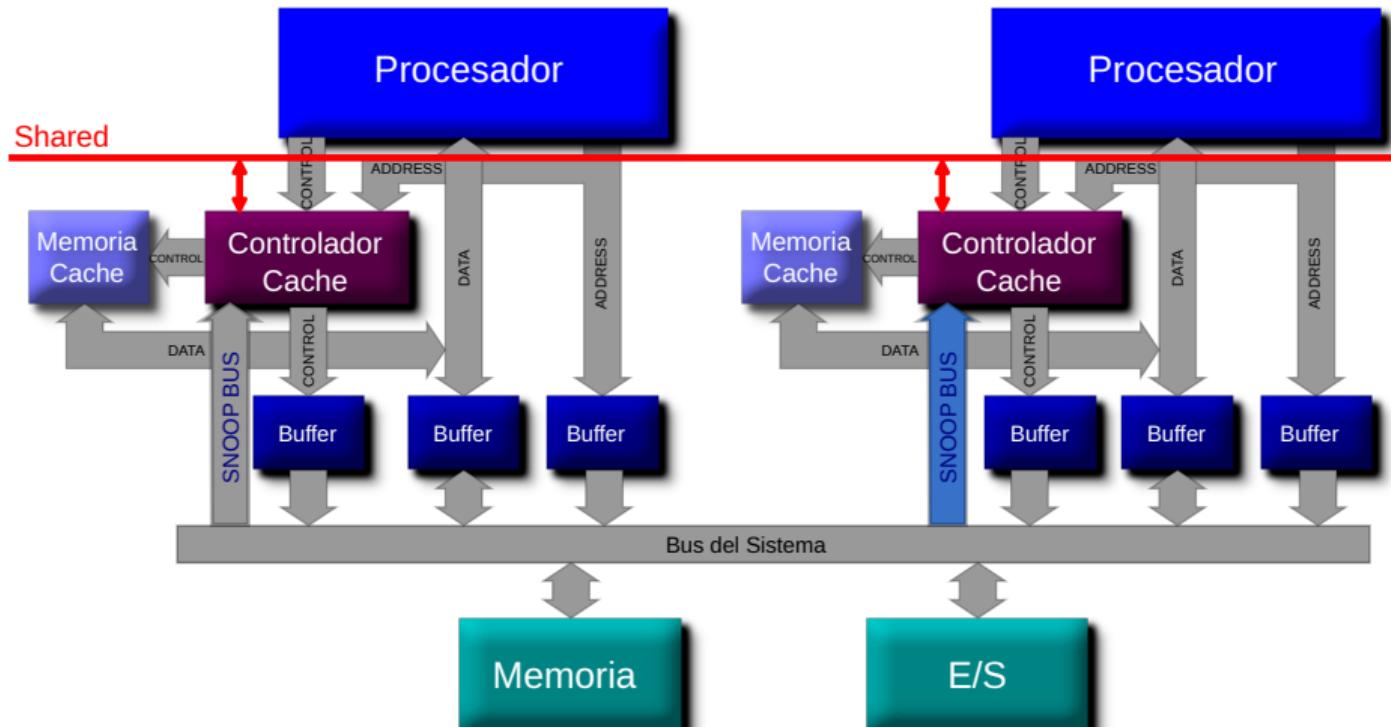
- Si por el Snoop Bus se detecta un Read Miss de una línea **Modified** en el cache local, el Controlador asume que tiene la única copia válida (Ownership), y que está incoherente con los niveles inferiores de la jerarquía (**Modified** y **Exclusive** usan copy back).
- Se activa RFO indicando al lector que el dato está incoherente.
- El lector aborta la lectura y pone sus Buses en Alta Impedancia.
- El Owner realiza el Copy Back de la línea.
- Ahora sigue teniendo la única copia de la línea, pero ésta es coherente con el resto de la jerarquía. La pone en estado en estado **Exclusive**, y desactiva RFO.
- Si el Snoop Bus detecta un acceso a una línea que tiene almacenada como **Exclusive** en su cache, cambia su estado a **Shared** y activa la línea Shared para avisar al resto.

# Protocolo MESI - Diagrama de estados



- Si por el Snoop Bus se detecta un Read Miss de una línea **Modified** en el cache local, el Controlador asume que tiene la única copia válida (Ownership), y que está incoherente con los niveles inferiores de la jerarquía (**Modified** y **Exclusive** usan copy back).
- Se activa RFO indicando al lector que el dato está incoherente.
- El lector aborta la lectura y pone sus Buses en Alta Impedancia.
- El Owner realiza el Copy Back de la línea.
- Ahora sigue teniendo la única copia de la línea, pero ésta es coherente con el resto de la jerarquía. La pone en estado en estado **Exclusive**, y desactiva RFO.
- Si el Snoop Bus detecta un acceso a una línea que tiene almacenada como **Exclusive** en su cache, cambia su estado a **Shared** y activa la línea Shared para avisar al resto.
- El que la está leyendo completa la lectura y pone la línea **Shared**.

# Protocolos de Coherencia: Señales



El lector de la línea accede al Bus producto de un Read Miss. No tiene forma de saber si hay una copia de la línea en otro cache. Por eso se emplea una línea Shared.

# Protocolo MESI. Límitaciones y fortalezas

- **MESI** es un protocolo eficiente para mantener la coherencia en un sistema multiprocesador, con diferentes niveles jerárquicos de memoria.

# Protocolo MESI. Límitaciones y fortalezas

- **MESI** es un protocolo eficiente para mantener la coherencia en un sistema multiprocesador, con diferentes niveles jerárquicos de memoria.
- Consigue optimizar el uso del Bus del sistema, habilitando escrituras **write back**, siempre que sea posible.

# Protocolo MESI. Límitaciones y fortalezas

- **MESI** es un protocolo eficiente para mantener la coherencia en un sistema multiprocesador, con diferentes niveles jerárquicos de memoria.
- Consigue optimizar el uso del Bus del sistema, habilitando escrituras **write back**, siempre que sea posible.
- El hecho de invalidar las copias cada vez que se escribe un dato y que esta escritura sea **write through**, es el área de oportunidad para optimizar sus virtudes.

# Protocolo MESI. Límitaciones y fortalezas

- **MESI** es un protocolo eficiente para mantener la coherencia en un sistema multiprocesador, con diferentes niveles jerárquicos de memoria.
- Consigue optimizar el uso del Bus del sistema, habilitando escrituras **write back**, siempre que sea posible.
- El hecho de invalidar las copias cada vez que se escribe un dato y que esta escritura sea **write through**, es el área de oportunidad para optimizar sus virtudes.
- Un efecto de **write invalidate** es que probablemente aquellos Cores que invalidaron su copia por un requerimiento de escritura, vuelvan a necesitarla. Solo la obtendrán desde los niveles jerárquicos inferiores con el costo de performance evidente.

# Protocolo MESI. Límitaciones y fortalezas

- **MESI** es un protocolo eficiente para mantener la coherencia en un sistema multiprocesador, con diferentes niveles jerárquicos de memoria.
- Consigue optimizar el uso del Bus del sistema, habilitando escrituras **write back**, siempre que sea posible.
- El hecho de invalidar las copias cada vez que se escribe un dato y que esta escritura sea **write through**, es el área de oportunidad para optimizar sus virtudes.
- Un efecto de **write invalidate** es que probablemente aquellos Cores que invalidaron su copia por un requerimiento de escritura, vuelvan a necesitarla. Solo la obtendrán desde los niveles jerárquicos inferiores con el costo de performance evidente.
- Este efecto aumenta con el número de procesadores.

# Protocolo MESI. Límitaciones y fortalezas

- **MESI** es un protocolo eficiente para mantener la coherencia en un sistema multiprocesador, con diferentes niveles jerárquicos de memoria.
- Consigue optimizar el uso del Bus del sistema, habilitando escrituras **write back**, siempre que sea posible.
- El hecho de invalidar las copias cada vez que se escribe un dato y que esta escritura sea **write through**, es el área de oportunidad para optimizar sus virtudes.
- Un efecto de **write invalidate** es que probablemente aquellos Cores que invalidaron su copia por un requerimiento de escritura, vuelvan a necesitarla. Solo la obtendrán desde los niveles jerárquicos inferiores con el costo de performance evidente.
- Este efecto aumenta con el número de procesadores.
- Fue menester buscar mejoras para implementar tecnologías multicore de manera exitosa.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- **Protocolos de Coherencia para Multicore**

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Implementación de Protocolos de Coherencia. MESIF

- Patente US 6.922.756 B2 Herbert H.J. Hum, James R. Goodman, asignada a Intel Corporation. Julio de 2005.

# Implementación de Protocolos de Coherencia. MESIF

- Patente US 6.922.756 B2 Herbert H.J. Hum, James R. Goodman, asignada a Intel Corporation. Julio de 2005.
- **MESIF** optimiza el acceso al Bus en sistemas con memoria distribuida.

# Implementación de Protocolos de Coherencia. MESIF

- Patente US 6.922.756 B2 Herbert H.J. Hum, James R. Goodman, asignada a Intel Corporation. Julio de 2005.
- **MESIF** optimiza el acceso al Bus en sistemas con memoria distribuida.
- Intel agrega a **MESI**, un estado adicional: **Forward** como una forma especial del estado **S**.

# Implementación de Protocolos de Coherencia. MESIF

- Patente US 6.922.756 B2 Herbert H.J. Hum, James R. Goodman, asignada a Intel Corporation. Julio de 2005.
- **MESIF** optimiza el acceso al Bus en sistemas con memoria distribuida.
- Intel agrega a **MESI**, un estado adicional: **Forward** como una forma especial del estado **S**.
- El protocolo debe asegurar que entre los caches que tienen una línea determinada en estado **S**, una de ellas tenga estado **F**.

# Implementación de Protocolos de Coherencia. MESIF

- Patente US 6.922.756 B2 Herbert H.J. Hum, James R. Goodman, asignada a Intel Corporation. Julio de 2005.
- **MESIF** optimiza el acceso al Bus en sistemas con memoria distribuida.
- Intel agrega a **MESI**, un estado adicional: **Forward** como una forma especial del estado **S**.
- El protocolo debe asegurar que entre los caches que tienen una línea determinada en estado **S**, una de ellas tenga estado **F**.
- La invalidación de una línea **F** y **S** no se informa. En esta situación si no queda ninguna línea en estado **F**, el próximo **Read Miss** será resuelto desde el nivel jerárquico inferior, con la penalización de performance consecuente.

# Implementación de Protocolos de Coherencia. MESIF

- Patente US 6.922.756 B2 Herbert H.J. Hum, James R. Goodman, asignada a Intel Corporation. Julio de 2005.
- **MESIF** optimiza el acceso al Bus en sistemas con memoria distribuida.
- Intel agrega a **MESI**, un estado adicional: **Forward** como una forma especial del estado **S**.
- El protocolo debe asegurar que entre los caches que tienen una línea determinada en estado **S**, una de ellas tenga estado **F**.
- La invalidación de una línea **F** y **S** no se informa. En esta situación si no queda ninguna línea en estado **F**, el próximo **Read Miss** será resuelto desde el nivel jerárquico inferior, con la penalización de performance consecuente.
- Para minimizar este efecto se asigna el estado **F** mediante el criterio **LRU**. El Cache que resuelve el Forward cede el estado **F**, a la línea del Cache que termina de leerse.

# Implementación de Protocolos de Coherencia. MOESI

- Este protocolo lo emplea AMD desde los procesadores Opteron.

# Implementación de Protocolos de Coherencia. MOESI

- Este protocolo lo emplea AMD desde los procesadores Opteron.
- Tambien es empleado por los procesadores de ARM (ARMv7 y ARMv8).

# Implementación de Protocolos de Coherencia. MOESI

- Este protocolo lo emplea AMD desde los procesadores Opteron.
- También es empleado por los procesadores de ARM (ARMv7 y ARMv8).
- Agrega un quinto estado a los que propone **MESI: Owned**.

# Implementación de Protocolos de Coherencia. MOESI

- Este protocolo lo emplea AMD desde los procesadores Opteron.
- Tambien es empleado por los procesadores de ARM (ARMv7 y ARMv8).
- Agrega un quinto estado a los que propone **MESI: Owned**.
- Invalid, Modified**, y **Exclusive**, se comparten como en **MESI**.

# Implementación de Protocolos de Coherencia. MOESI

- Este protocolo lo emplea AMD desde los procesadores Opteron.
- Tambien es empleado por los procesadores de ARM (ARMv7 y ARMv8).
- Agrega un quinto estado a los que propone **MESI: Owned**.
- Invalid, Modified**, y **Exclusive**, se comparten como en **MESI**.
- Para el estado **Shared**, introduce una diferencia no menor, consecuencia del nuevo estado.

# Implementación de Protocolos de Coherencia. MOESI

- Este protocolo lo emplea AMD desde los procesadores Opteron.
- Tambien es empleado por los procesadores de ARM (ARMv7 y ARMv8).
- Agrega un quinto estado a los que propone **MESI: Owned**.
- Invalid, Modified**, y **Exclusive**, se comparten como en **MESI**.
- Para el estado **Shared**, introduce una diferencia no menor, consecuencia del nuevo estado.
- Shared** indica que esta copia de la linea puede estar en otros Caches (sigue siendo impreciso), y es válida (es la copia mas reciente de la linea). Hasta aqui es igual que en **MESI**

# Implementación de Protocolos de Coherencia. MOESI

- Este protocolo lo emplea AMD desde los procesadores Opteron.
- Tambien es empleado por los procesadores de ARM (ARMv7 y ARMv8).
- Agrega un quinto estado a los que propone **MESI: Owned**.
- Invalid, Modified**, y **Exclusive**, se comparten como en **MESI**.
- Para el estado **Shared**, introduce una diferencia no menor, consecuencia del nuevo estado.
- Shared** indica que esta copia de la linea puede estar en otros Caches (sigue siendo impreciso), y es válida (es la copia mas reciente de la linea). Hasta aqui es igual que en **MESI**
- Pero a diferencia de **MESI**, la copia en memoria principal puede ahora no ser válida.

# Implementación de Protocolos de Coherencia. MOESI

- Este protocolo lo emplea AMD desde los procesadores Opteron.
- Tambien es empleado por los procesadores de ARM (ARMv7 y ARMv8).
- Agrega un quinto estado a los que propone **MESI: Owned**.
- Invalid, Modified**, y **Exclusive**, se comparten como en **MESI**.
- Para el estado **Shared**, introduce una diferencia no menor, consecuencia del nuevo estado.
- Shared** indica que esta copia de la linea puede estar en otros Caches (sigue siendo impreciso), y es válida (es la copia mas reciente de la linea). Hasta aqui es igual que en **MESI**
- Pero a diferencia de **MESI**, la copia en memoria principal puede ahora no ser válida.
- Si ningún Cache tiene una copia de esa misma linea en estado **Owned**, la copia de memoria principal es válida. Sino...no. Y aquí reside la diferencia.

# Implementación de Protocolos de Coherencia. MOESI

- Solo una copia de una línea determinada puede tomar estado **Owned** en un Cache.

# Implementación de Protocolos de Coherencia. MOESI

- Solo una copia de una línea determinada puede tomar estado **Owned** en un Cache.
- Indica que se trata de una de varias copias válidas de la línea, pero con la diferencia que ésta es la única con derecho para escribirla sin efectuar **write back**.

# Implementación de Protocolos de Coherencia. MOESI

- Solo una copia de una línea determinada puede tomar estado **Owned** en un Cache.
- Indica que se trata de una de varias copias válidas de la línea, pero con la diferencia que ésta es la única con derecho para escribirla sin efectuar **write back**.
- El resto de las copias compartidas permanecen **Shared** y válidas.

# Implementación de Protocolos de Coherencia. MOESI

- Solo una copia de una línea determinada puede tomar estado **Owned** en un Cache.
- Indica que se trata de una de varias copias válidas de la línea, pero con la diferencia que ésta es la única con derecho para escribirla sin efectuar **write back**.
- El resto de las copias compartidas permanecen **Shared** y válidas.
- Cada vez que se escribe una línea **Owned**, se activa la transacción por el Bus local y el resto de los Caches que tienen copias **Shared** de esa línea la actualizan.

# Implementación de Protocolos de Coherencia. MOESI

- Solo una copia de una línea determinada puede tomar estado **Owned** en un Cache.
- Indica que se trata de una de varias copias válidas de la línea, pero con la diferencia que ésta es la única con derecho para escribirla sin efectuar **write back**.
- El resto de las copias compartidas permanecen **Shared** y válidas.
- Cada vez que se escribe una línea **Owned**, se activa la transacción por el Bus local y el resto de los Caches que tienen copias **Shared** de esa línea la actualizan.
- De este modo disminuyen los **write back** por intentar leer una línea Dirty, y las invalidaciones al escribir una linea **Shared**.

# Implementación de Protocolos de Coherencia. MOESI

- Solo una copia de una línea determinada puede tomar estado **Owned** en un Cache.
- Indica que se trata de una de varias copias válidas de la línea, pero con la diferencia que ésta es la única con derecho para escribirla sin efectuar **write back**.
- El resto de las copias compartidas permanecen **Shared** y válidas.
- Cada vez que se escribe una línea **Owned**, se activa la transacción por el Bus local y el resto de los Caches que tienen copias **Shared** de esa línea la actualizan.
- De este modo disminuyen los **write back** por intentar leer una línea Dirty, y las invalidaciones al escribir una linea **Shared**.
- Además el cache que tiene la línea **Owned** se encarga de proveerla a “nuevos lectores”, los que le asignarán al leer **Shared**.

# Implementación de Protocolos de Coherencia. MOESI

- Solo una copia de una línea determinada puede tomar estado **Owned** en un Cache.
- Indica que se trata de una de varias copias válidas de la línea, pero con la diferencia que ésta es la única con derecho para escribirla sin efectuar **write back**.
- El resto de las copias compartidas permanecen **Shared** y válidas.
- Cada vez que se escribe una línea **Owned**, se activa la transacción por el Bus local y el resto de los Caches que tienen copias **Shared** de esa línea la actualizan.
- De este modo disminuyen los **write back** por intentar leer una línea Dirty, y las invalidaciones al escribir una linea **Shared**.
- Además el cache que tiene la línea **Owned** se encarga de proveerla a “nuevos lectores”, los que le asignarán al leer **Shared**.
- En caso de que nadie tenga la línea, ésta se trae desde los niveles jerárquicos inferiores.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore

### • Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

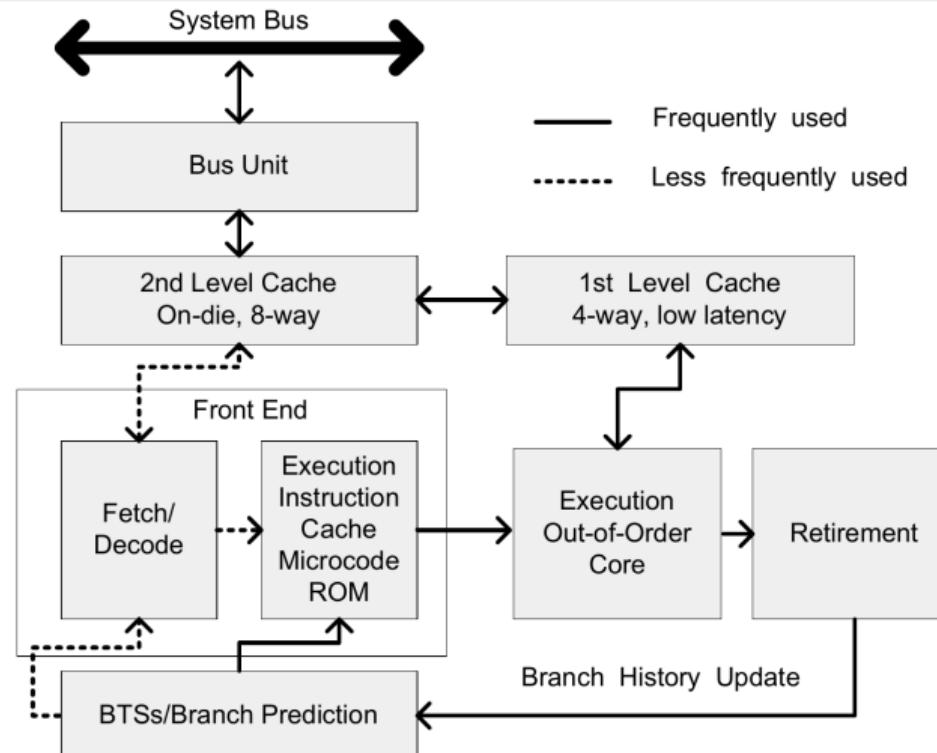
## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

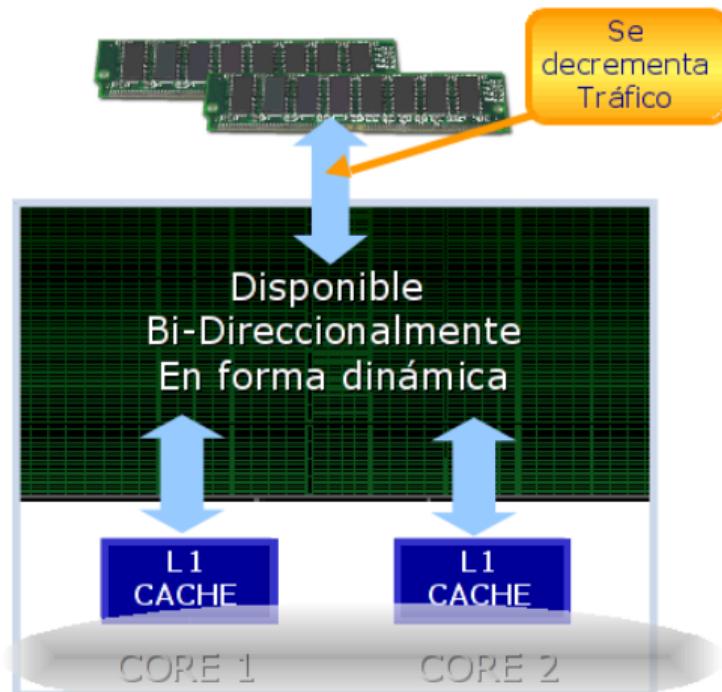
## 2do. Nivel On chip



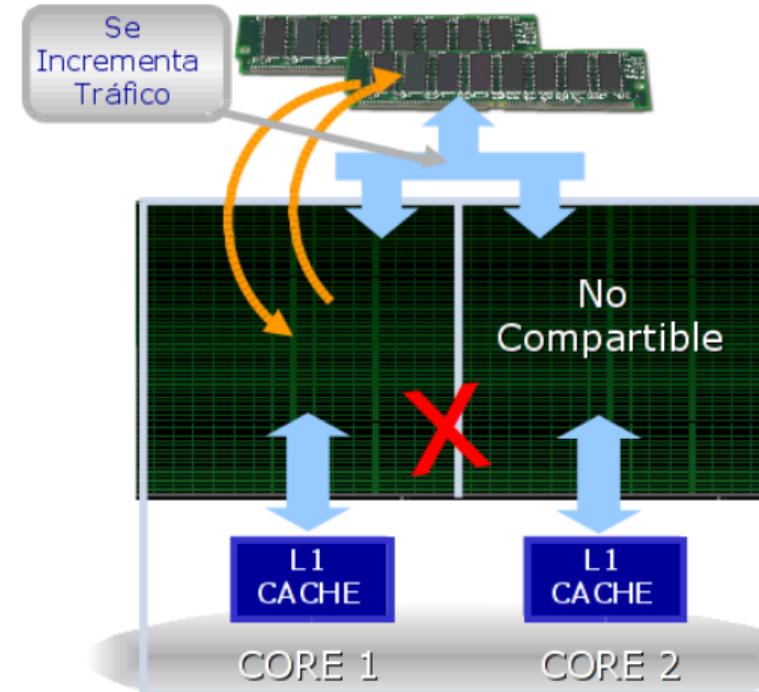
Intel P6 Architecture

# Smart Cache

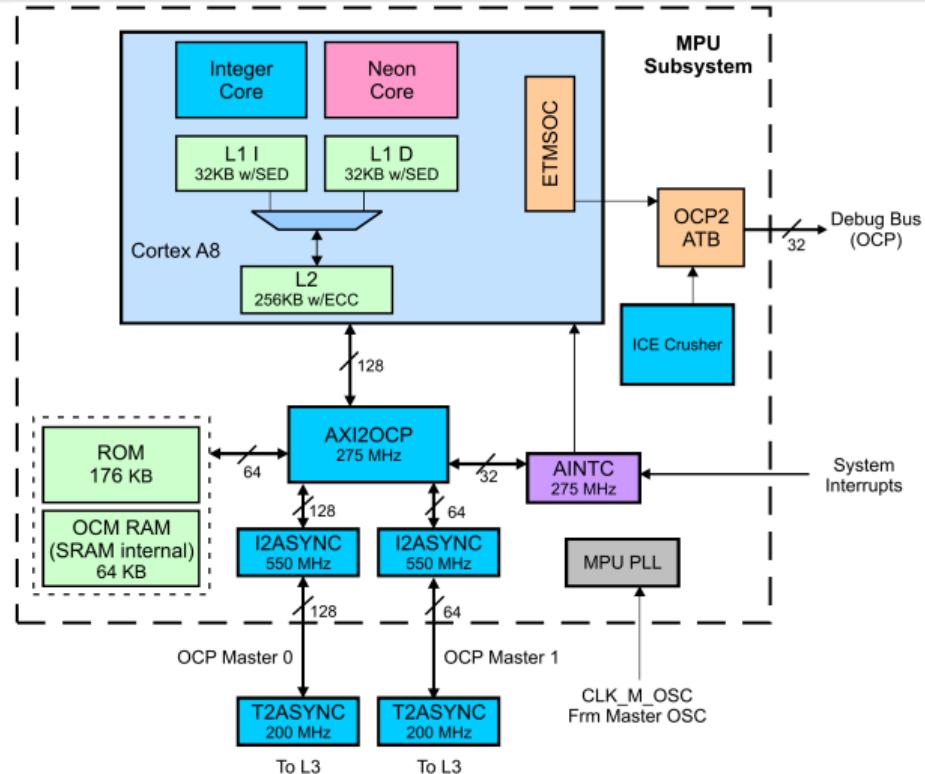
## L2 Compartida Microarquitectura Core



## L2 Independiente

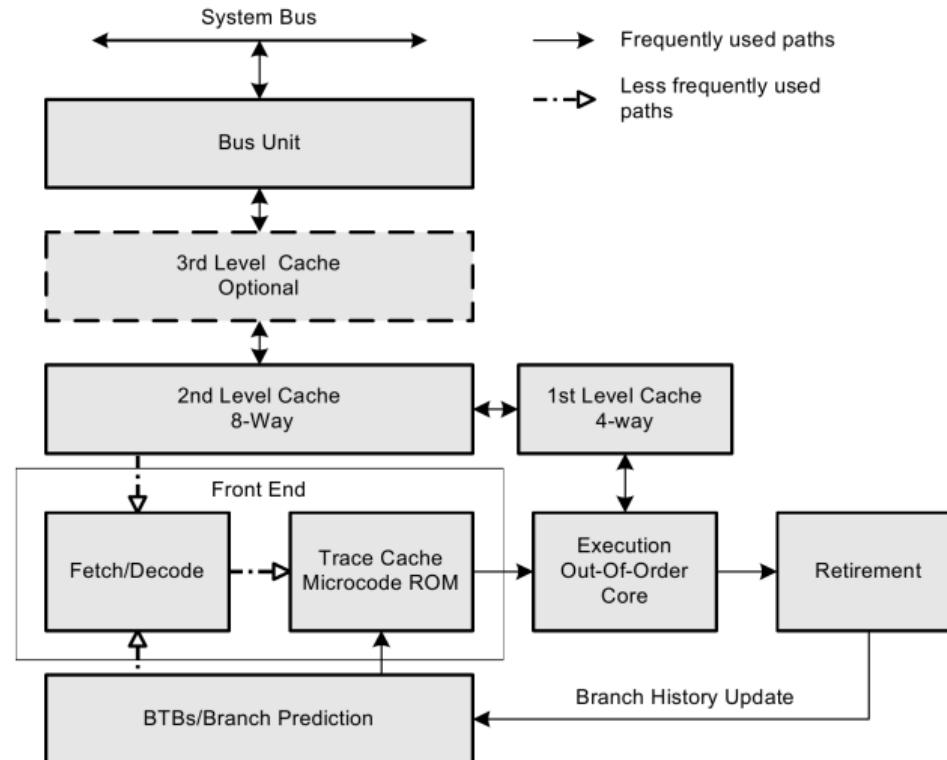


# 2do. Nivel On chip



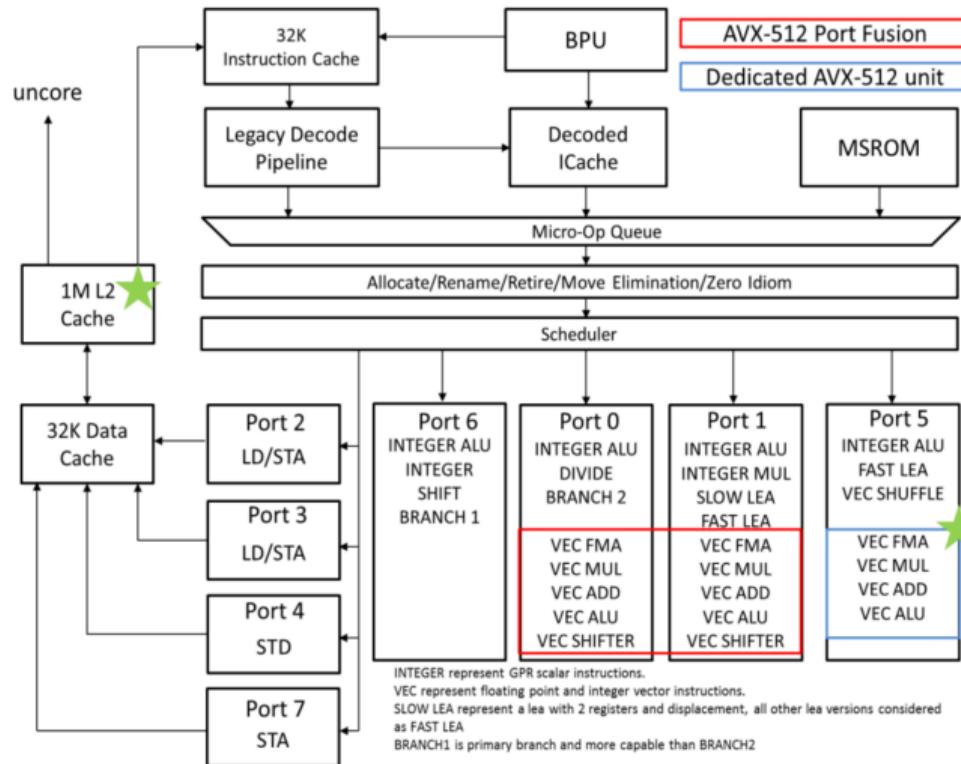
Cortex A8 Architecture

# 3er. Nivel On chip



Intel Netburst Architecture

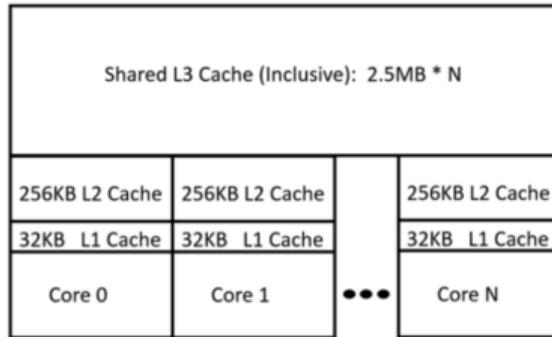
# 3er. Nivel On chip



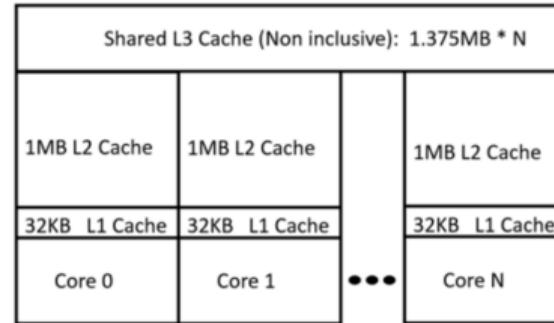
Intel Skylake Server Architecture

# 3er. Nivel On chip

Broadwell Server Cache Structure



Skylake Server Cache Structure



## Intel Skylake vs Broadwell Server Architecture

Level	Capacity / Associativity	Line Size (bytes)	Fastest Latency <sup>1</sup>	Peak Bandwidth (bytes/cyc)	Sustained Bandwidth (bytes/cyc)	Update Policy
First Level Data	32 KB/ 8	64	4 cycle	96 (2x32B Load + 1*32B Store)	~81	Writeback
Instruction	32 KB/8	64	N/A	N/A	N/A	N/A
Second Level	256KB/4	64	12 cycle	64	~29	Writeback
Third Level (Shared L3)	Up to 2MB per core/Up to 16 ways	64	44	32	~18	Writeback

## Intel Skylake Server Architecture - Parámetros de Cache

# Memorias Dinámicas.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Conceptos Básicos

# Conceptos Básicos

- A diferencia de las memorias cache que van generalmente en el mismo chip de la CPU, estas memoria al momento se ubican en uno o mas chips separados del de la CPU.

# Conceptos Básicos

- A diferencia de las memorias cache que van generalmente en el mismo chip de la CPU, estas memoria al momento se ubican en uno o mas chips separados del de la CPU.
- Por tal motivo se deben considerar en su diseño los siguientes aspectos:

# Conceptos Básicos

- A diferencia de las memorias cache que van generalmente en el mismo chip de la CPU, estas memoria al momento se ubican en uno o mas chips separados del de la CPU.
- Por tal motivo se deben considerar en su diseño los siguientes aspectos:
  - Pines No solo el pinout sino sus características eléctricas (inductancia, capacitancia, etc.)

# Conceptos Básicos

- A diferencia de las memorias cache que van generalmente en el mismo chip de la CPU, estas memoria al momento se ubican en uno o mas chips separados del de la CPU.
- Por tal motivo se deben considerar en su diseño los siguientes aspectos:
  - Pines No solo el pinout sino sus características eléctricas (inductancia, capacitancia, etc.)
  - Señalización Handshake con el hardware controlador

# Conceptos Básicos

- A diferencia de las memorias cache que van generalmente en el mismo chip de la CPU, estas memoria al momento se ubican en uno o mas chips separados del de la CPU.
- Por tal motivo se deben considerar en su diseño los siguientes aspectos:
  - Pines No solo el pinout sino sus características eléctricas (inductancia, capacitancia, etc.)
  - Señalización Handshake con el hardware controlador
  - Signal Integrity En función de la frecuencia de trabajo

# Conceptos Básicos

- A diferencia de las memorias cache que van generalmente en el mismo chip de la CPU, estas memoria al momento se ubican en uno o mas chips separados del de la CPU.
- Por tal motivo se deben considerar en su diseño los siguientes aspectos:
  - Pines No solo el pinout sino sus características eléctricas (inductancia, capacitancia, etc.)
  - Señalización Handshake con el hardware controlador
  - Signal Integrity En función de la frecuencia de trabajo
  - Encapsulado Define la manufacturabilidad.

# Conceptos Básicos

- A diferencia de las memorias cache que van generalmente en el mismo chip de la CPU, estas memoria al momento se ubican en uno o mas chips separados del de la CPU.
- Por tal motivo se deben considerar en su diseño los siguientes aspectos:

**Pines** No solo el pinout sino sus características eléctricas (inductancia, capacitancia, etc.)

**Señalización** Handshake con el hardware controlador

**Signal Integrity** En función de la frecuencia de trabajo

**Encapsulado** Define la manufacturabilidad.

**Clock y sincro** Define criterios de montaje en el PCB y adaptaciones eléctricas al bus

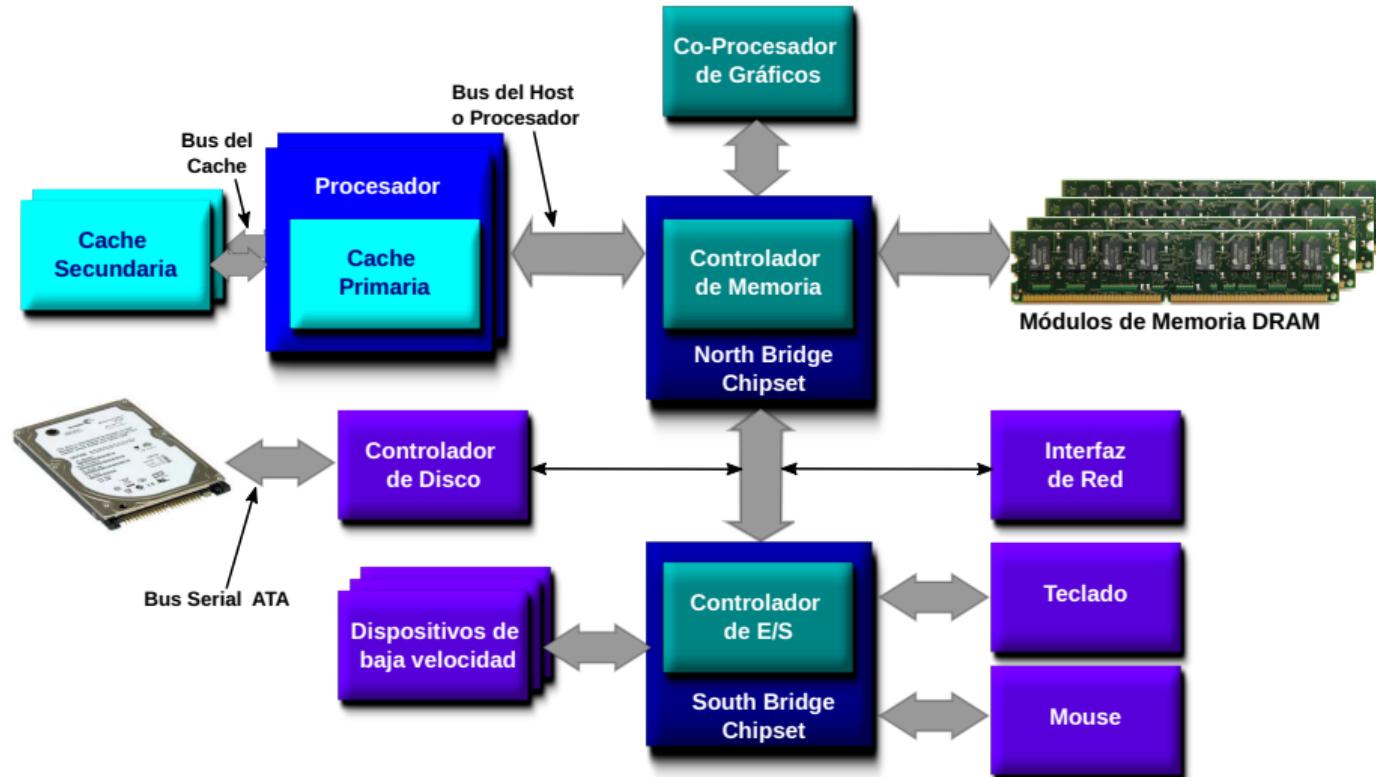
# Conceptos Básicos

- A diferencia de las memorias cache que van generalmente en el mismo chip de la CPU, estas memoria al momento se ubican en uno o mas chips separados del de la CPU.
- Por tal motivo se deben considerar en su diseño los siguientes aspectos:
  - Pines** No solo el pinout sino sus características eléctricas (inductancia, capacitancia, etc.)
  - Señalización** Handshake con el hardware controlador
  - Signal Integrity** En función de la frecuencia de trabajo
  - Encapsulado** Define la manufacturabilidad.
  - Clock y sincro** Define criterios de montaje en el PCB y adaptaciones eléctricas al bus
  - Timing** Refresco, tiempo de acceso para lectura y escritura

# Conceptos Básicos

- A diferencia de las memorias cache que van generalmente en el mismo chip de la CPU, estas memoria al momento se ubican en uno o mas chips separados del de la CPU.
- Por tal motivo se deben considerar en su diseño los siguientes aspectos:
  - Pines** No solo el pinout sino sus características eléctricas (inductancia, capacitancia, etc.)
  - Señalización** Handshake con el hardware controlador
  - Signal Integrity** En función de la frecuencia de trabajo
  - Encapsulado** Define la manufacturabilidad.
  - Clock y sincro** Define criterios de montaje en el PCB y adaptaciones eléctricas al bus
  - Timing** Refresco, tiempo de acceso para lectura y escritura
- Si no se respetan estos aspectos se puede caer en diseños no óptimos o no funcionales.

# Interacción dentro de un computador



Como ya hemos abordado, el Cache Secundario puede ser L2 o L3 y puede estar en el mismo chip del procesador.

# Interacción dentro de un computador

- En el gráfico anterior puede observarse la presencia de un Controlador de memoria Dinámica.

# Interacción dentro de un computador

- En el gráfico anterior puede observarse la presencia de un Controlador de memoria Dinámica.
- Este dispositivo media entre la CPU y el/los chip/s de DRAM.

# Interacción dentro de un computador

- En el gráfico anterior puede observarse la presencia de un Controlador de memoria Dinámica.
- Este dispositivo media entre la CPU y el/los chip/s de DRAM.
- Su presencia sugiere demoras en el tiempo de acceso (veremos que no es el único factor, pero evidentemente una mediación implica etapas lógicas).

# Interacción dentro de un computador

- En el gráfico anterior puede observarse la presencia de un Controlador de memoria Dinámica.
- Este dispositivo media entre la CPU y el/los chip/s de DRAM.
- Su presencia sugiere demoras en el tiempo de acceso (veremos que no es el único factor, pero evidentemente una mediación implica etapas lógicas).
- Sugiere además una complejidad en el sistema de DRAM que a esta altura no estamos en condiciones de valorar adecuadamente, pero que al cabo es estos slides podremos verificar.

# Interacción dentro de un computador

- En el gráfico anterior puede observarse la presencia de un Controlador de memoria Dinámica.
- Este dispositivo media entre la CPU y el/los chip/s de DRAM.
- Su presencia sugiere demoras en el tiempo de acceso (veremos que no es el único factor, pero evidentemente una mediación implica etapas lógicas).
- Sugiere además una complejidad en el sistema de DRAM que a esta altura no estamos en condiciones de valorar adecuadamente, pero que al cabo es estos slides podremos verificar.
- Pero también, como veremos, aporta independencia a la CPU de los detalles de la DRAM (timing, refresco, manejo de la estructura interna, por citar los mas obvios)

# Cacárcetísticas de una celda

# Cacárcetísticas de una celda

- Un bit se implementa con un solo transistor que en los hechos actúa como capacitor.

# Cacárcetísticas de una celda

- Un bit se implementa con un solo transistor que en los hechos actúa como capacitor.
- Una implicancia directa de ello es que al leerlo se destruye su información ya que para proveer el estado de la celda debe cederse la carga (o no carga) del capacitor.

# Características de una celda

- Un bit se implementa con un solo transistor que en los hechos actúa como capacitor.
- Una implicancia directa de ello es que al leerlo se destruye su información ya que para proveer el estado de la celda debe cederse la carga (o no carga) del capacitor.
- Por lo tanto cada bit leído debe ser vuelto a escribir. Si bien esto ocurre cuando el dato ya está disponible retarda el inicio del siguiente ciclo de lectura eventualmente.

## Cacárcetísticas de una celda

- Un bit se implementa con un solo transistor que en los hechos actúa como capacitor.
- Una implicancia directa de ello es que al leerlo se destruye su información ya que para proveer el estado de la celda debe cederse la carga (o no carga) del capacitor.
- Por lo tanto cada bit leído debe ser vuelto a escribir. Si bien esto ocurre cuando el dato ya está disponible retarda el inicio del siguiente ciclo de lectura eventualmente.
- El término dinámica proviene del hecho de que al no implementarse con transistores ideales las corrientes de fuga hacen que se pierda la carga del capacitor.

## Cacárcetísticas de una celda

- Un bit se implementa con un solo transistor que en los hechos actúa como capacitor.
- Una implicancia directa de ello es que al leerlo se destruye su información ya que para proveer el estado de la celda debe cederse la carga (o no carga) del capacitor.
- Por lo tanto cada bit leído debe ser vuelto a escribir. Si bien esto ocurre cuando el dato ya está disponible retarda el inicio del siguiente ciclo de lectura eventualmente.
- El término dinámica proviene del hecho de que al no implementarse con transistores ideales las corrientes de fuga hacen que se pierda la carga del capacitor.
- Este hecho hace que deba ser refrescada periódicamente para que no se pierda la información almacenada.

# Características de una celda

- Desde el punto de vista eléctrico leer un bit no resulta trivial.

# Cacárcetísticas de una celda

- Desde el punto de vista eléctrico leer un bit no resulta trivial.
- Hay que poder sensar el estado de carga durante un intervalo muy pequeño de tiempo ya que una vez cedida en forma de corriente eléctrica si no logra detectarse, la carga se habrá perdido irremediablemente y con ella el estado lógico de la celda.

# Cacárcetísticas de una celda

- Desde el punto de vista eléctrico leer un bit no resulta trivial.
- Hay que poder sensar el estado de carga durante un intervalo muy pequeño de tiempo ya que una vez cedida en forma de corriente eléctrica si no logra detectarse, la carga se habrá perdido irremediablemente y con ella el estado lógico de la celda.
- Para ello los conductores que transportan la señal se precargan en un estado intermedio entre '0' y '1'.

# Cacárcetísticas de una celda

- Desde el punto de vista eléctrico leer un bit no resulta trivial.
- Hay que poder sensar el estado de carga durante un intervalo muy pequeño de tiempo ya que una vez cedida en forma de corriente eléctrica si no logra detectarse, la carga se habrá perdido irremediablemente y con ella el estado lógico de la celda.
- Para ello los conductores que transportan la señal se precargan en un estado intermedio entre '0' y '1'.
- La carga de la capacidad de entrada del transistor desbalancea ese estado de precarga en mas o en menos.

## Cacárcetísticas de una celda

- Desde el punto de vista eléctrico leer un bit no resulta trivial.
- Hay que poder sensar el estado de carga durante un intervalo muy pequeño de tiempo ya que una vez cedida en forma de corriente eléctrica si no logra detectarse, la carga se habrá perdido irremediablemente y con ella el estado lógico de la celda.
- Para ello los conductores que transportan la señal se precargan en un estado intermedio entre '0' y '1'.
- La carga de la capacidad de entrada del transistor desbalancea ese estado de precarga en mas o en menos.
- Por medio de amplificadores de detección se detecta el desbalance "empujando" la celda al estado lógico inicial.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Organización de una DRAM genérica

# Organización de una DRAM genérica

- La organización de las celdas de una memoria DRAM es como una matriz rectangular.

# Organización de una DRAM genérica

- La organización de las celdas de una memoria DRAM es como una matriz rectangular.
- Originalmente la cantidad de pines de un chip era un problema serio desde el punto de vista mecánico (cantidad de pines, su ancho y separación).

# Organización de una DRAM genérica

- La organización de las celdas de una memoria DRAM es como una matriz rectangular.
- Originalmente la cantidad de pines de un chip era un problema serio desde el punto de vista mecánico (cantidad de pines, su ancho y separación).
- Por ello se organizaba en una matriz de celdas de  $n \times m$  con  $n$  y  $m$  en lo posible del mismo valor o con una unidad de diferencia. Se multiplexaban los terminales y se seleccionaba en primer lugar la fila y luego la columna.

# Organización de una DRAM genérica

- La organización de las celdas de una memoria DRAM es como una matriz rectangular.
- Originalmente la cantidad de pines de un chip era un problema serio desde el punto de vista mecánico (cantidad de pines, su ancho y separación).
- Por ello se organizaba en una matriz de celdas de  $n \times m$  con  $n$  y  $m$  en lo posible del mismo valor o con una unidad de diferencia. Se multiplexaban los terminales y se seleccionaba en primer lugar la fila y luego la columna.
- La cantidad de terminales de address se reduce a la mitad

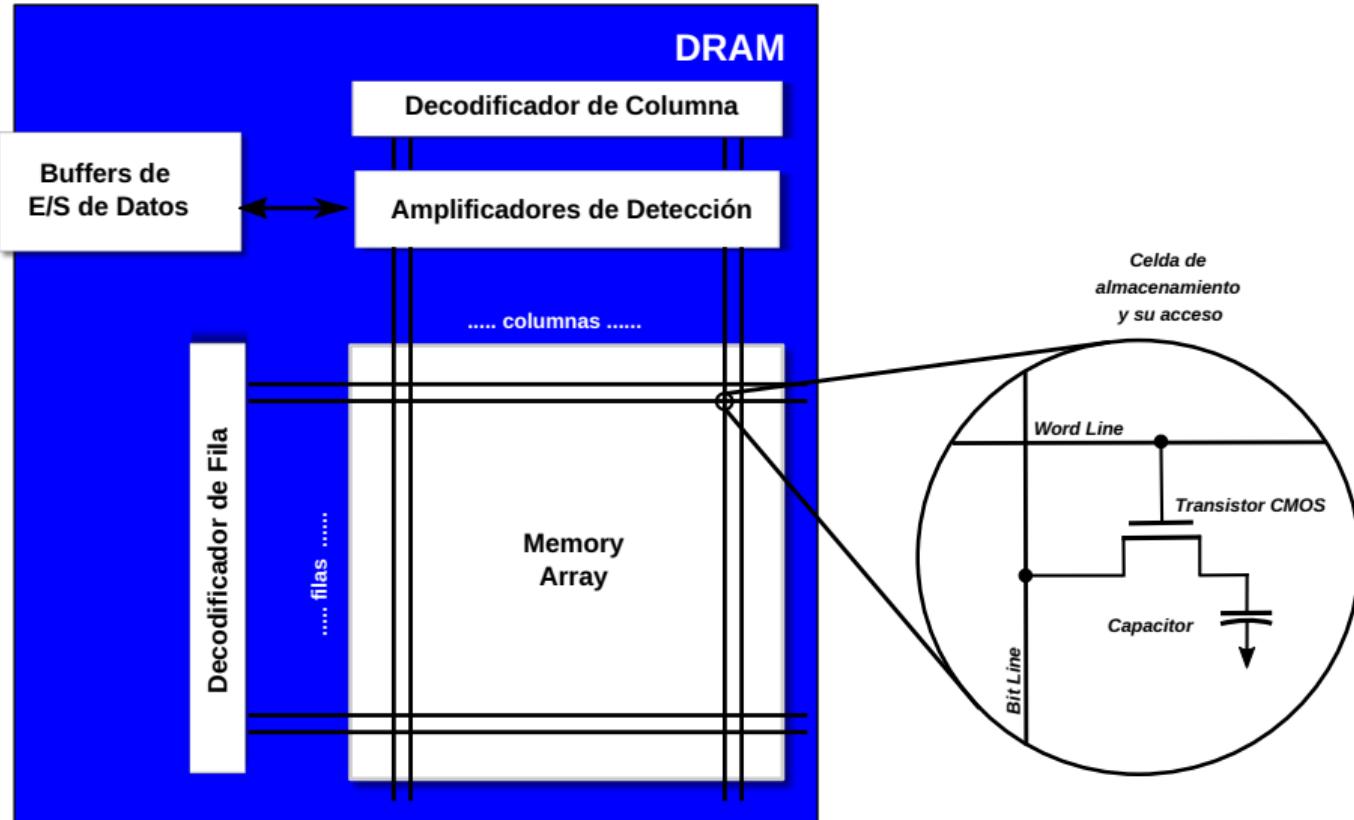
# Organización de una DRAM genérica

- La organización de las celdas de una memoria DRAM es como una matriz rectangular.
- Originalmente la cantidad de pines de un chip era un problema serio desde el punto de vista mecánico (cantidad de pines, su ancho y separación).
- Por ello se organizaba en una matriz de celdas de  $n \times m$  con  $n$  y  $m$  en lo posible del mismo valor o con una unidad de diferencia. Se multiplexaban los terminales y se seleccionaba en primer lugar la fila y luego la columna.
- La cantidad de terminales de address se reduce a la mitad
- La señal RAS (Row Address Strobe) selecciona la fila a leer y CAS (Column Address Strobe) los elementos de esa fila que se copiarán en el buffer de lectura.

# Organización de una DRAM genérica

- La organización de las celdas de una memoria DRAM es como una matriz rectangular.
- Originalmente la cantidad de pines de un chip era un problema serio desde el punto de vista mecánico (cantidad de pines, su ancho y separación).
- Por ello se organizaba en una matriz de celdas de  $n \times m$  con  $n$  y  $m$  en lo posible del mismo valor o con una unidad de diferencia. Se multiplexaban los terminales y se seleccionaba en primer lugar la fila y luego la columna.
- La cantidad de terminales de address se reduce a la mitad
- La señal RAS (Row Address Strobe) selecciona la fila a leer y CAS (Column Address Strobe) los elementos de esa fila que se copiarán en el buffer de lectura.
- Por esta razón se adoptó éste tipo de organización. Y se mantuvo hasta ahora.

# Organización de una DRAM genérica



# Organización de una DRAM genérica

- Cada *die* de DRAM, contiene uno o varios arrays de  $n \times m$  celdas.

# Organización de una DRAM genérica

- Cada *die* de DRAM, contiene uno o varios arrays de  $n \times m$  celdas.
- Cada celda es un bit de almacenamiento (transistor / capacitor)

# Organización de una DRAM genérica

- Cada *die* de DRAM, contiene uno o varios arrays de  $n \times m$  celdas.
- Cada celda es un bit de almacenamiento (transistor / capacitor)
- Los arrays dentro de cada celda pueden trabajar completamente asociados, completamente disociados, o en configuraciones intermedias entre las opciones anteriores.

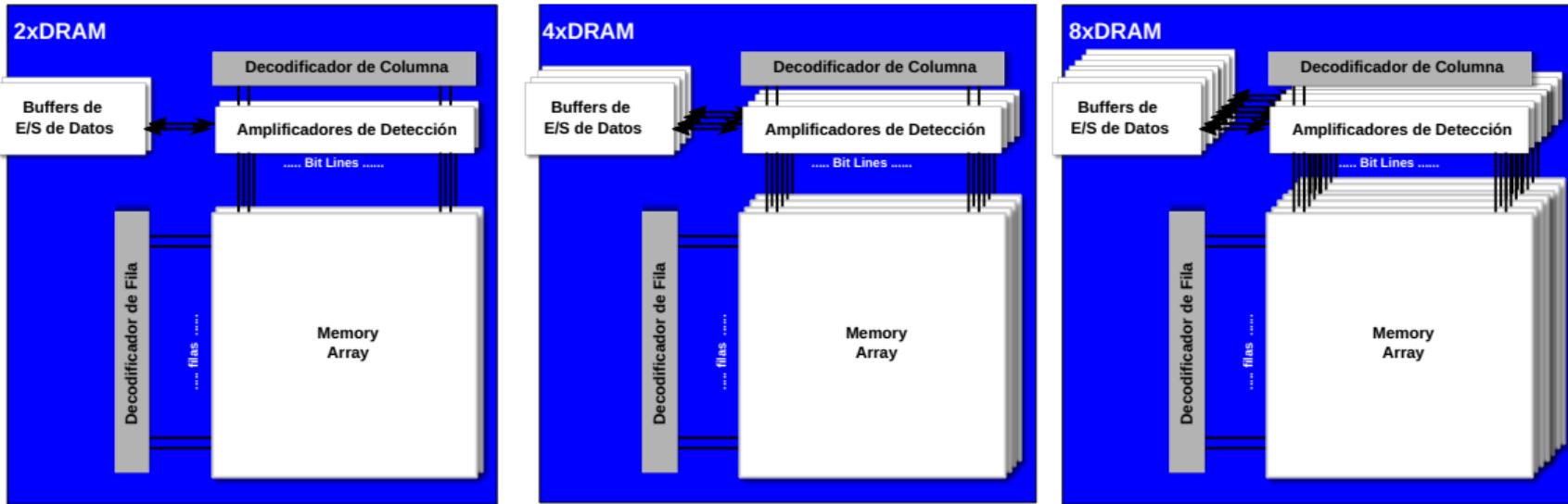
# Organización de una DRAM genérica

- Cada *die* de DRAM, contiene uno o varios arrays de  $n \times m$  celdas.
- Cada celda es un bit de almacenamiento (transistor / capacitor)
- Los arrays dentro de cada celda pueden trabajar completamente asociados, completamente disociados, o en configuraciones intermedias entre las opciones anteriores.
- Si trabajan en forma asociada, todos los arrays transmiten o reciben el bit correspondiente al valor de fila y columna, proveyendo acceso a un número de tantos bits como arrays haya en el die.

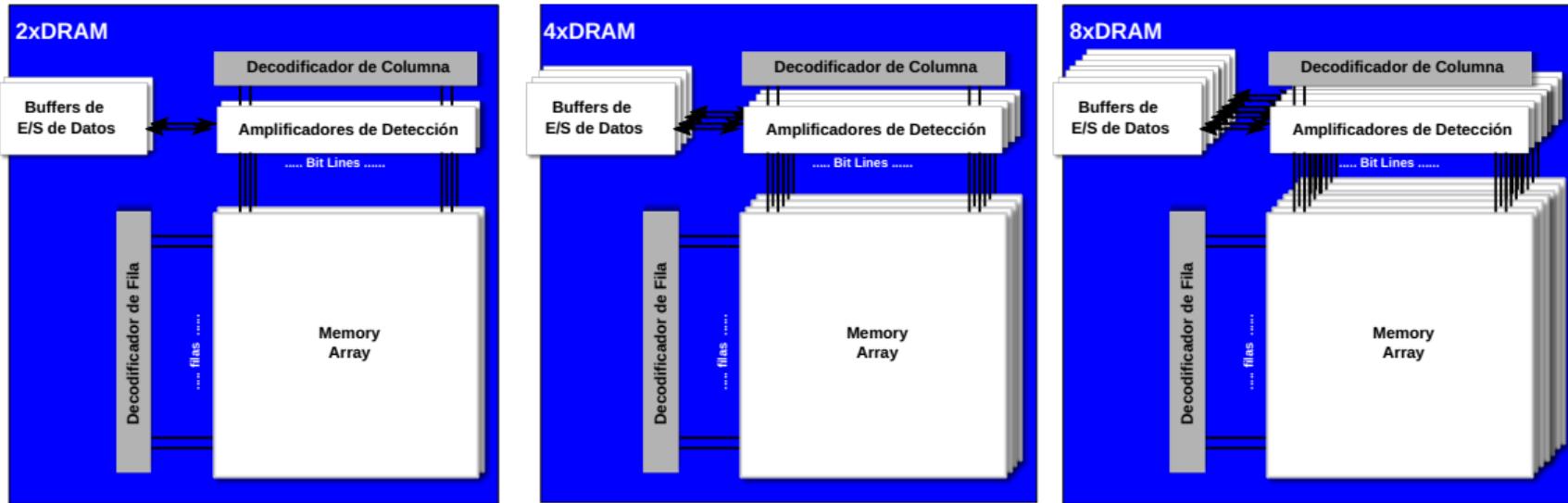
# Organización de una DRAM genérica

- Cada *die* de DRAM, contiene uno o varios arrays de  $n \times m$  celdas.
- Cada celda es un bit de almacenamiento (transistor / capacitor)
- Los arrays dentro de cada celda pueden trabajar completamente asociados, completamente disociados, o en configuraciones intermedias entre las opciones anteriores.
- Si trabajan en forma asociada, todos los arrays transmiten o reciben el bit correspondiente al valor de fila y columna, proveyendo acceso a un número de tantos bits como arrays haya en el die.
- Por ejemplo una x4 DRAM (se dice “por cuatro”), entrega un nibble por cada valor de fila y columna que se le provee.

# Organización de una DRAM genérica

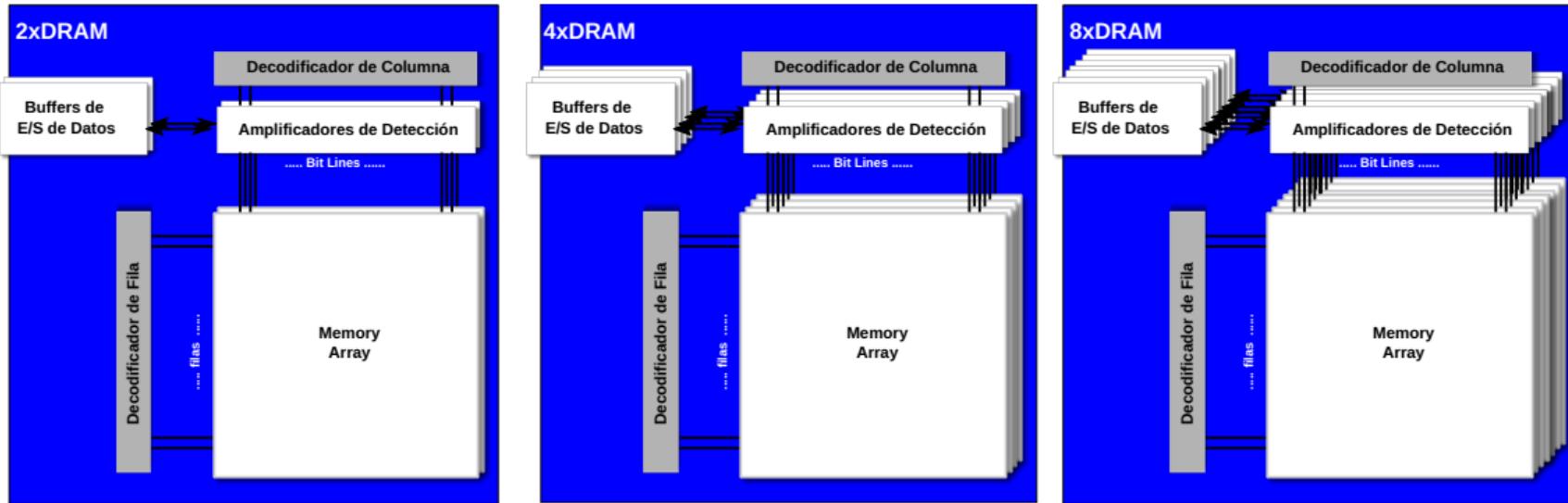


# Organización de una DRAM genérica



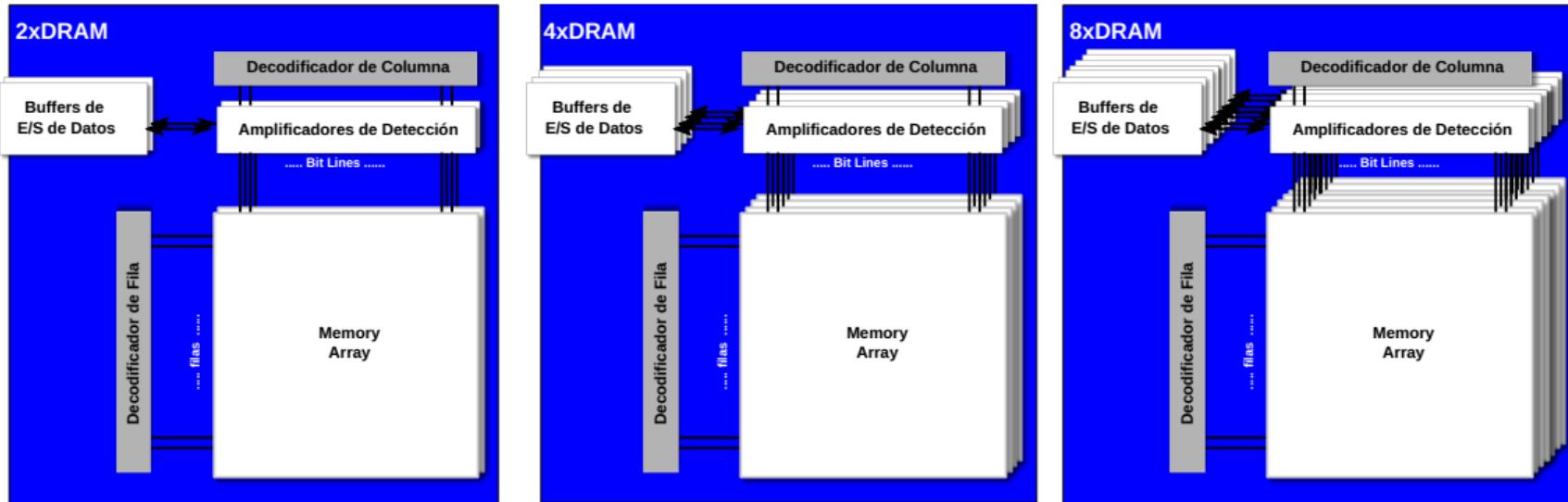
- Las configuraciones habituales para sistemas de media capacidad son x2, x4, y x8.

# Organización de una DRAM genérica



- Las configuraciones habituales para sistemas de media capacidad son x2, x4, y x8.
- Para las PCs actuales y servidores de mayor capacidad en los 90 comenzaron a producirse x16 y x32

# Organización de una DRAM genérica



- Las configuraciones habituales para sistemas de media capacidad son x2, x4, y x8.
- Para las PCs actuales y servidores de mayor capacidad en los 90 comenzaron a producirse x16 y x32
- Estos arrays conforman bancos que operarán independientemente de otros bancos.

# Bancos de DRAM

# Bancos de DRAM

- Operan en forma independiente como set de arrays con algunas restricciones:

# Bancos de DRAM

- Operan en forma independiente como set de arrays con algunas restricciones:
- Deben ser activados, precargados, leídos, etc. al mismo tiempo que otros bancos, estén todos dentro del mismo dispositivo DRAM o en dispositivos diferentes.

# Bancos de DRAM

- Operan en forma independiente como set de arrays con algunas restricciones:
- Deben ser activados, precargados, leídos, etc. al mismo tiempo que otros bancos, estén todos dentro del mismo dispositivo DRAM o en dispositivos diferentes.
- El uso de bancos independientes permite aumentar la velocidad de acceso a estos dispositivos, entrelazando los accesos a bancos diferentes de memoria.

# Bancos de DRAM

- Operan en forma independiente como set de arrays con algunas restricciones:
- Deben ser activados, precargados, leídos, etc. al mismo tiempo que otros bancos, estén todos dentro del mismo dispositivo DRAM o en dispositivos diferentes.
- El uso de bancos independientes permite aumentar la velocidad de acceso a estos dispositivos, entrelazando los accesos a bancos diferentes de memoria.
- Esto se consigue con hardware de mediación entre la memoria y el procesador.

# Bancos de DRAM

- Operan en forma independiente como set de arrays con algunas restricciones:
- Deben ser activados, precargados, leídos, etc. al mismo tiempo que otros bancos, estén todos dentro del mismo dispositivo DRAM o en dispositivos diferentes.
- El uso de bancos independientes permite aumentar la velocidad de acceso a estos dispositivos, entrelazando los accesos a bancos diferentes de memoria.
- Esto se consigue con hardware de mediación entre la memoria y el procesador.
- Si los bancos tienen un tiempo de acceso de 10 ns, leer en forma entrelazada dos bancos permite acceder a 5 ns

# Bancos de DRAM

- Operan en forma independiente como set de arrays con algunas restricciones:
- Deben ser activados, precargados, leídos, etc. al mismo tiempo que otros bancos, estén todos dentro del mismo dispositivo DRAM o en dispositivos diferentes.
- El uso de bancos independientes permite aumentar la velocidad de acceso a estos dispositivos, entrelazando los accesos a bancos diferentes de memoria.
- Esto se consigue con hardware de mediación entre la memoria y el procesador.
- Si los bancos tienen un tiempo de acceso de 10 ns, leer en forma entrelazada dos bancos permite acceder a 5 ns
- Si se hace round robin con 4 bancos, el tiempo de acceso percibido por el procesador es de 2,5 ns.

# Dual In Module Memory: DIMM

# Dual In Module Memory: DIMM

- En sistemas que requieren alta capacidad de almacenamiento las DRAM Se organizan en DIMM's.

# Dual In Module Memory: DIMM

- En sistemas que requieren alta capacidad de almacenamiento las DRAM Se organizan en DIMM's.
- Los dispositivos DRAM se montan a ambos lados del PCB

# Dual In Module Memory: DIMM

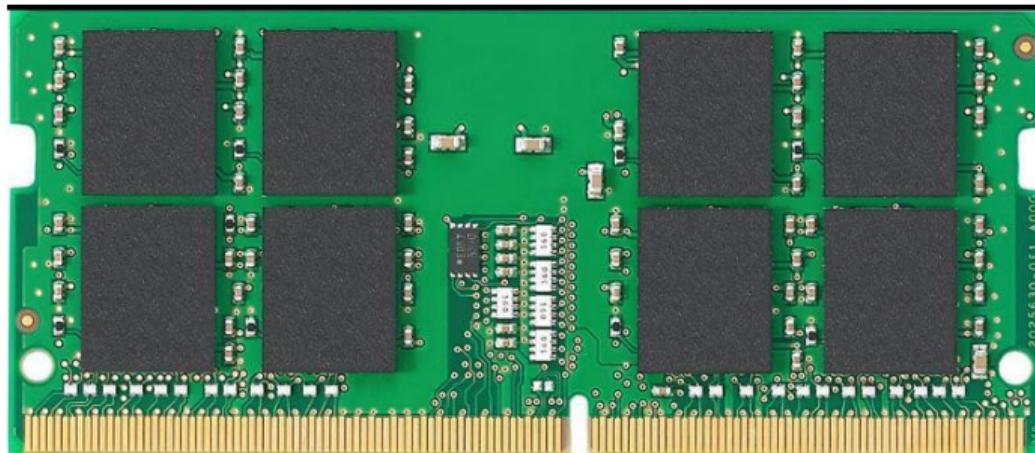
- En sistemas que requieren alta capacidad de almacenamiento las DRAM Se organizan en DIMM's.
- Los dispositivos DRAM se montan a ambos lados del PCB
- Cada DIMM puede pensarse como un banco independiente.

# Dual In Module Memory: DIMM

- En sistemas que requieren alta capacidad de almacenamiento las DRAM Se organizan en DIMM's.
- Los dispositivos DRAM se montan a ambos lados del PCB
- Cada DIMM puede pensarse como un banco independiente.
- O bien, cada dispositivo o grupo de dispositivos DRAM de un DIMM se puede asignar a un banco independiente.

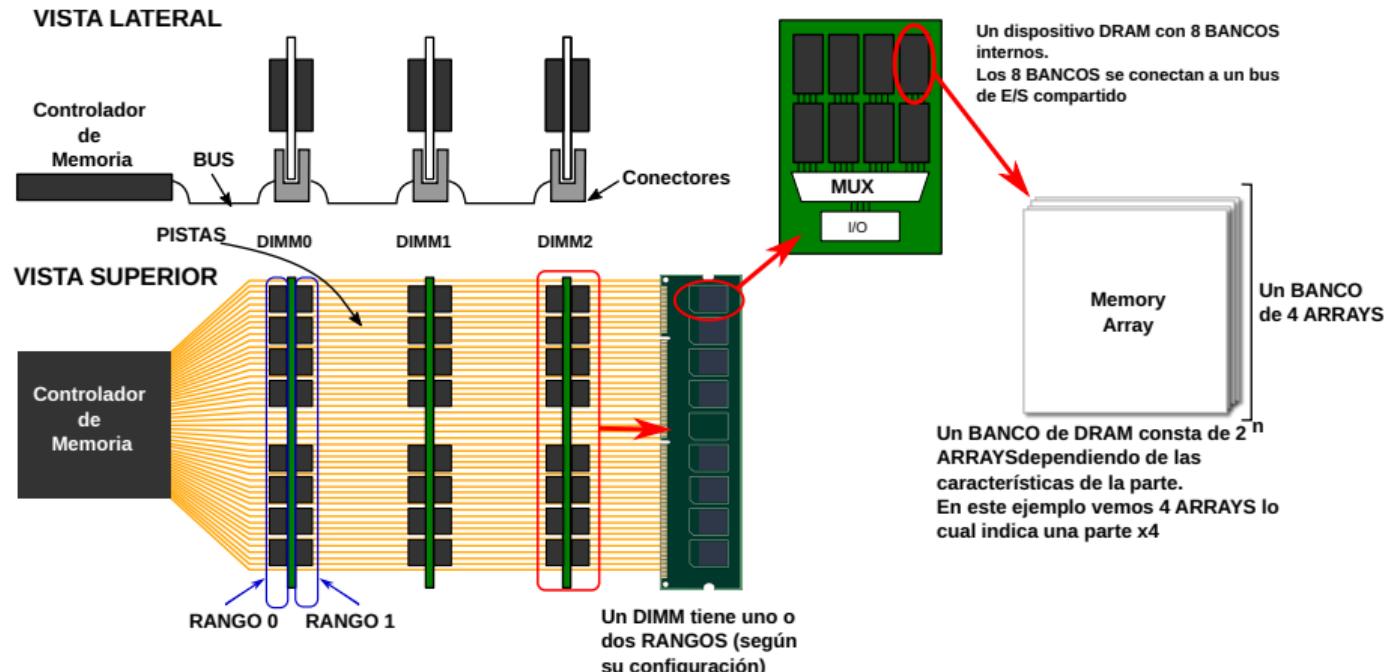
# Dual In Module Memory: DIMM

- En sistemas que requieren alta capacidad de almacenamiento las DRAM Se organizan en DIMM's.
- Los dispositivos DRAM se montan a ambos lados del PCB
- Cada DIMM puede pensarse como un banco independiente.
- O bien, cada dispositivo o grupo de dispositivos DRAM de un DIMM se puede asignar a un banco independiente.



Memoria Ram Sodimm Kingston 32GB 3200 Mhz DDR4

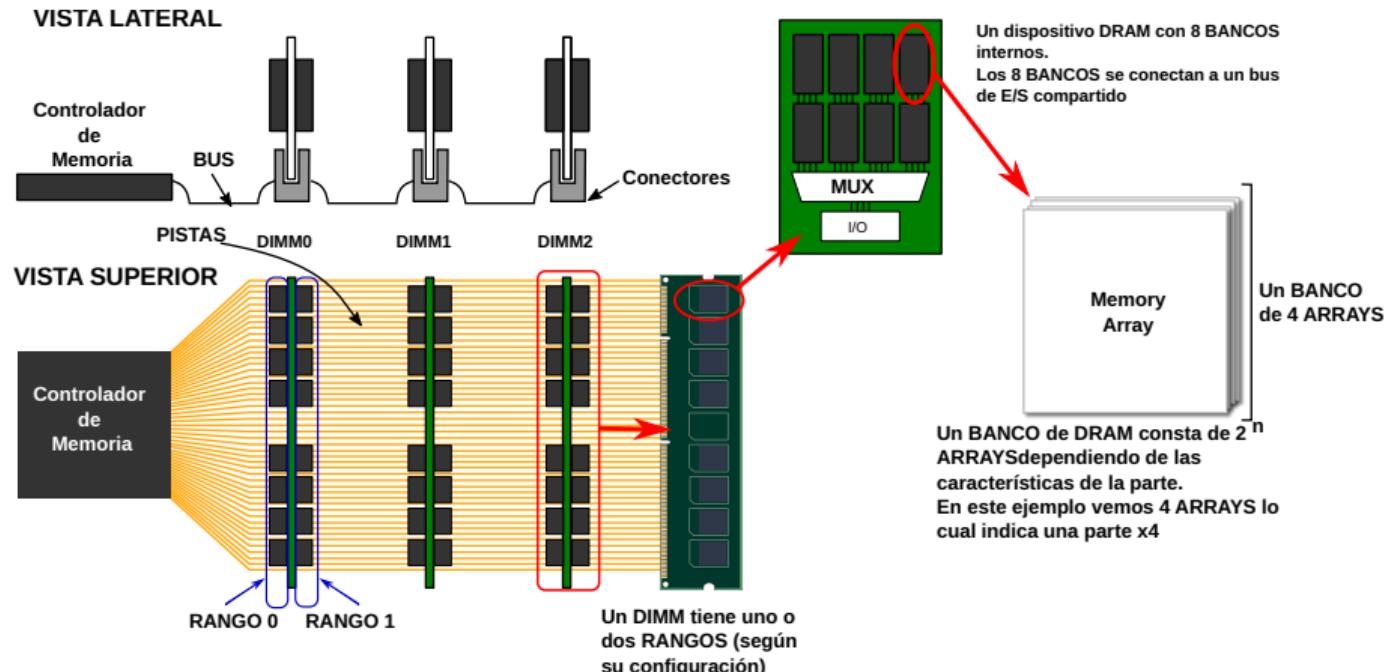
# Organización general de un DIMM



## Rango (“rank”)

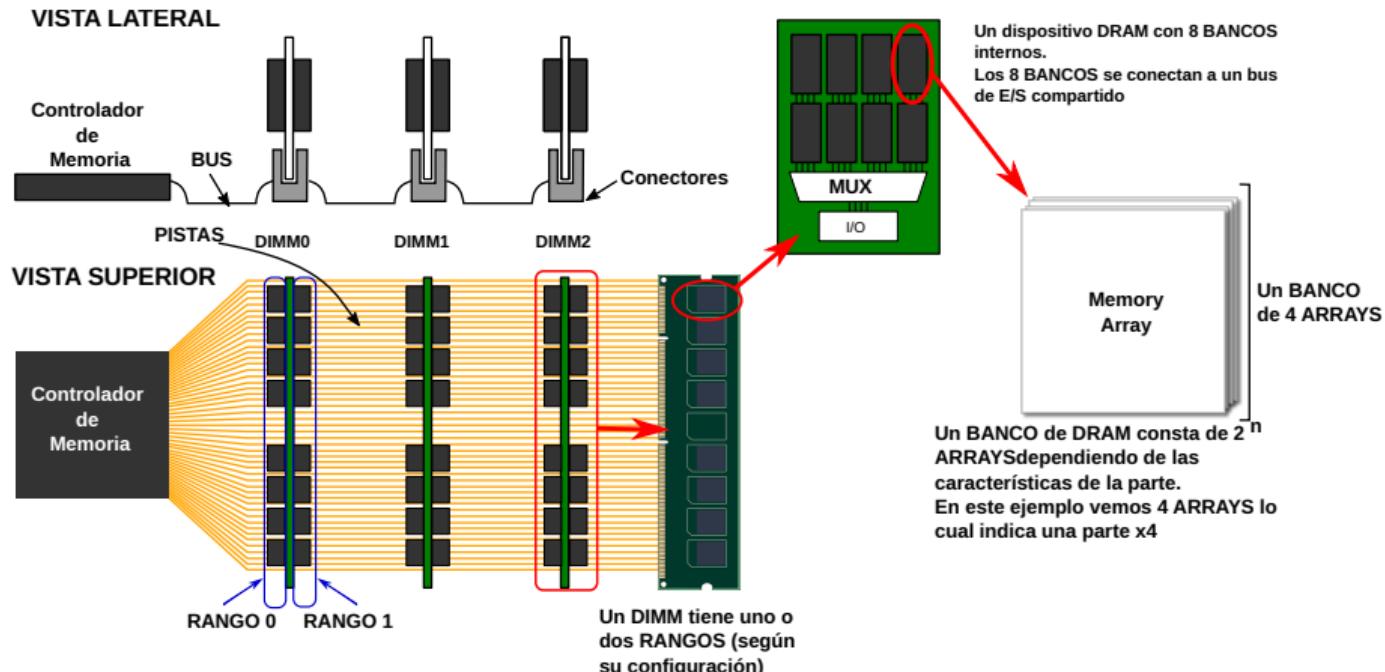
Es un set de dispositivos DRAM (todos los de un DIMM o una parte de ellos), que operan en forma conjunta.

# Organización general de un DIMM



Cada dispositivo DRAM implementa internamente uno o mas bancos independientes que operan de forma independiente entre sí.

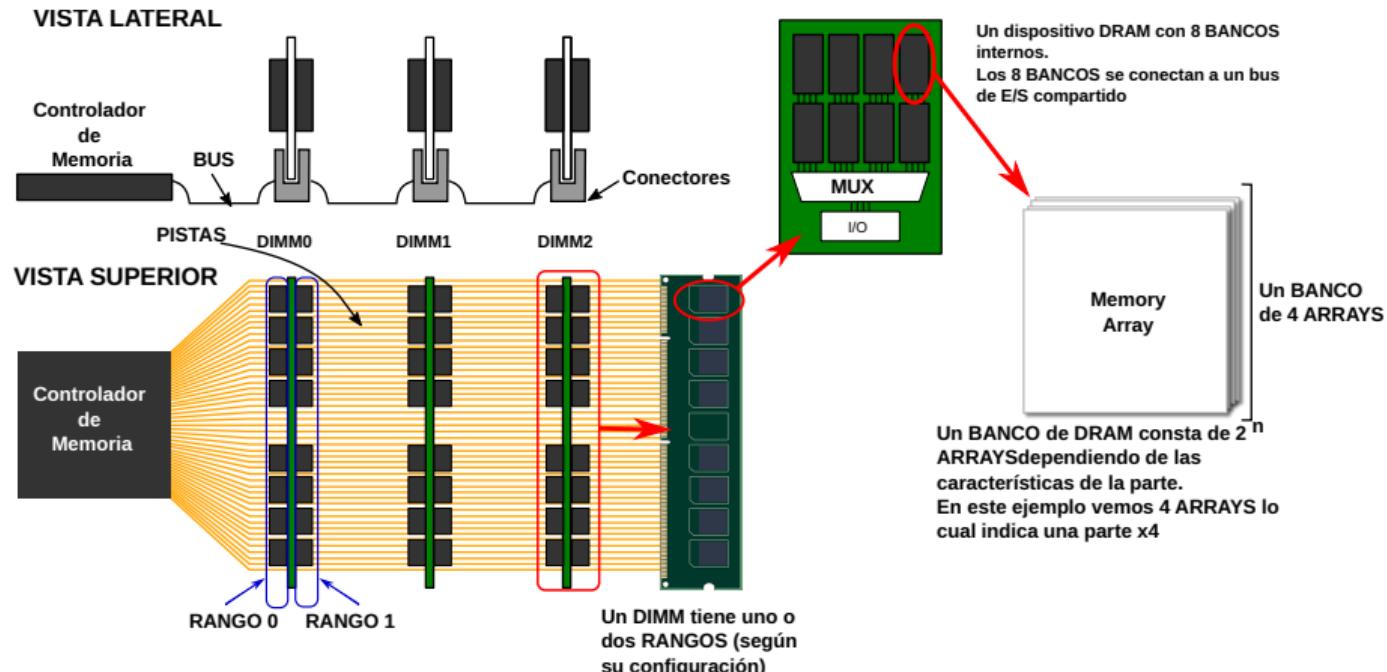
# Organización general de un DIMM



## Array

Cada banco de un dispositivo DRAM se compone de un conjunto de arrays esclavos, cuyo número determina el ancho del dispositivo DRAM (x2, x4, etc.).

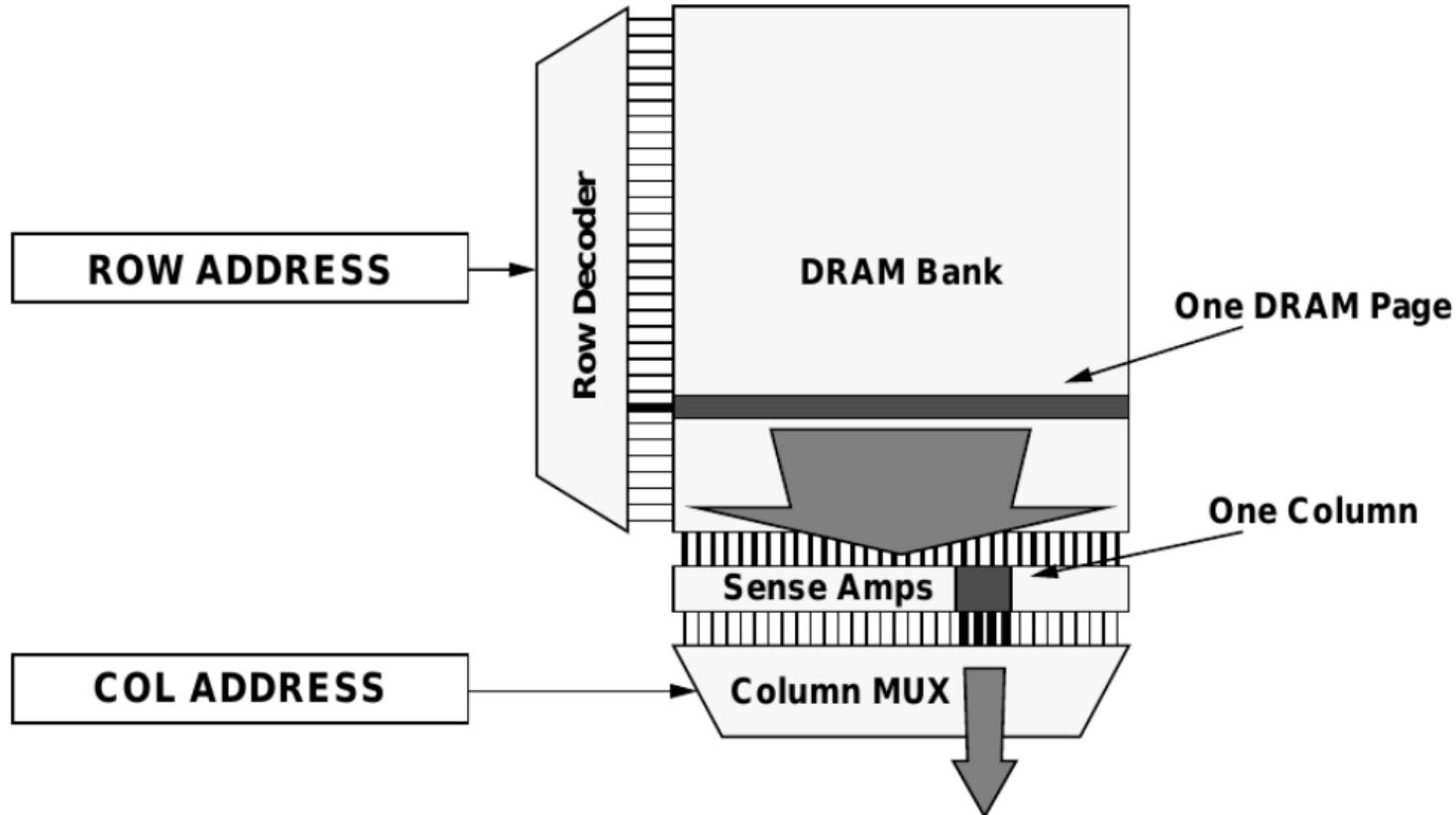
# Organización general de un DIMM



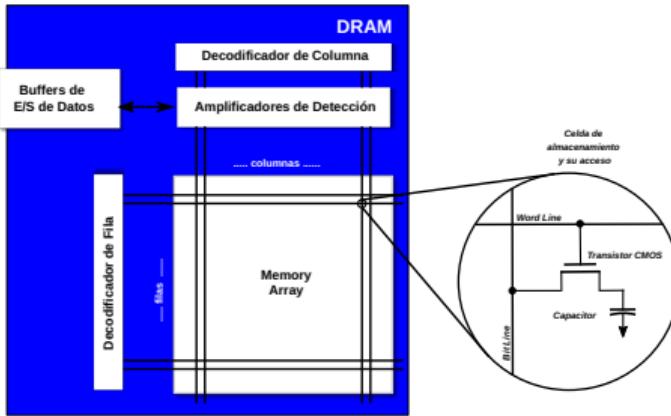
Concurrencia = paralelismo = Bandwidth

Accesos concurrentes a bancos y rangos con un “request pipeline” aumenta el ancho de banda, ya que operan en paralelo múltiples DRAMs a nivel de rango y múltiples arrays a nivel de banco.

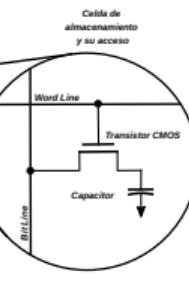
# Acceso multi-fase a una DRAM



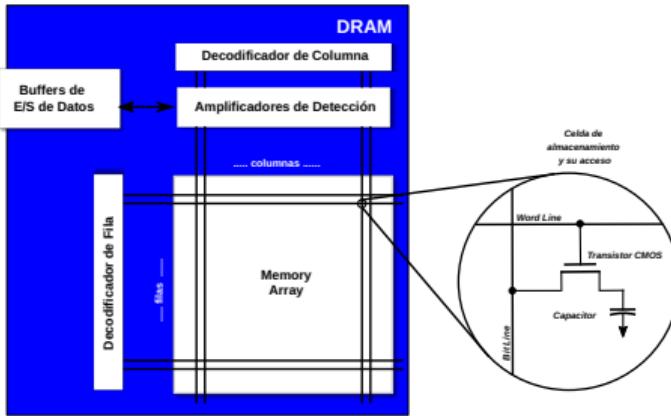
# Dentro del DRAM Device



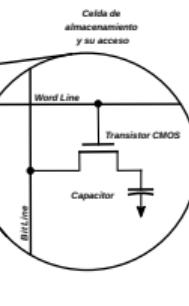
- Cada capacitor se ubica en la intersección entre un word-line y un bitline (decodificación de fila y columna).



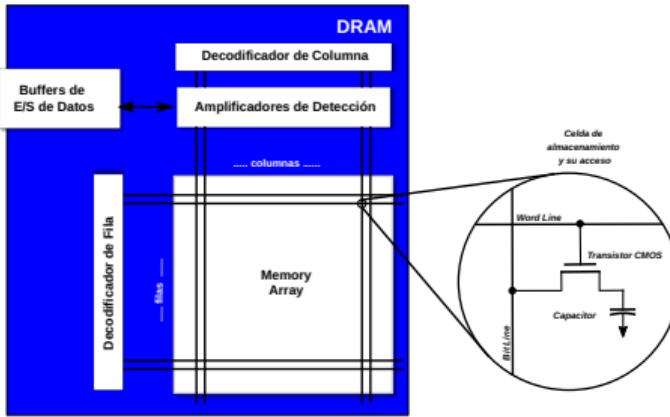
# Dentro del DRAM Device



- Cada capacitor se ubica en la intersección entre un word-line y un bitline (decodificación de fila y columna).
- Se conecta al bitline mediante un transistor controlado por la wordline.

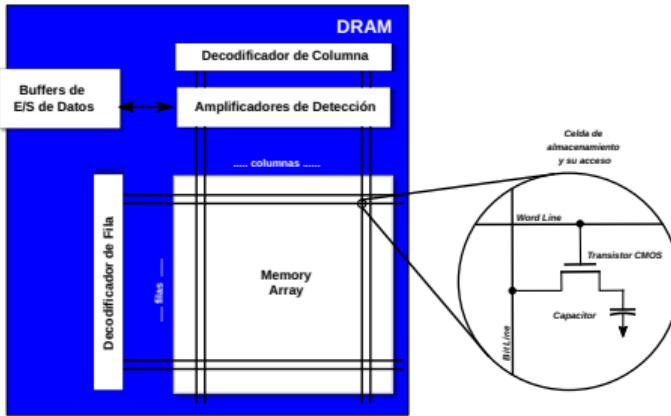


# Dentro del DRAM Device



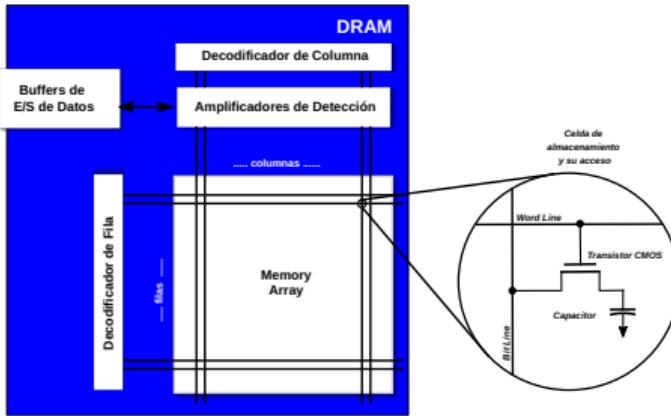
- Cada capacitor se ubica en la intersección entre un word-line y un bitline (decodificación de fila y columna).
- Se conecta al bitline mediante un transistor controlado por la wordline.
- La wordline que reciba un nivel de tensión alto, satura a todos los transistores cuyo gate tenga conectado (como un switch cerrado).

# Dentro del DRAM Device



- Cada capacitor se ubica en la intersección entre un word-line y un bitline (decodificación de fila y columna).
  - Se conecta al bitline mediante un transistor controlado por la wordline.
  - La wordline que reciba un nivel de tensión alto, satura a todos los transistores cuyo gate tenga conectado (como un switch cerrado).
- 
- En las escalas de integración actuales, la cantidad de electrones que puede almacenar el capacitor no puede producir una señal detectable en el bitline, debido a la diferencia de las escalas físicas.

# Dentro del DRAM Device



- Cada capacitor se ubica en la intersección entre un word-line y un bitline (decodificación de fila y columna).
- Se conecta al bitline mediante un transistor controlado por la wordline.
- La wordline que reciba un nivel de tensión alto, satura a todos los transistores cuyo gate tenga conectado (como un switch cerrado).

- En las escalas de integración actuales, la cantidad de electrones que puede almacenar el capacitor no puede producir una señal detectable en el bitline, debido a la diferencia de las escalas físicas.
- Para detectar los valores eléctricos almacenados en los capacitores se requiere un conjunto de “amplificadores de detección” conectados a las bitlines.

# Dentro del DRAM Device

- El amplificador de detección, en principio, *precarga* las bitlines con un valor eléctrico intermedio entre '0' y '1'.

## Dentro del DRAM Device

- El amplificador de detección, en principio, *precarga* las bitlines con un valor eléctrico intermedio entre '0' y '1'.
- Al habilitarse una wordline, el capacitor de la bitline habilitada produce una alteración muy pequeña en el valor de tensión precargado, en mas o en menos dependiendo del estado de su carga.

# Dentro del DRAM Device

- El amplificador de detección, en principio, *precarga* las bitlines con un valor eléctrico intermedio entre '0' y '1'.
- Al habilitarse una wordline, el capacitor de la bitline habilitada produce una alteración muy pequeña en el valor de tensión precargado, en mas o en menos dependiendo del estado de su carga.
- Esa diferencia es muy pequeña, pero el amplificador de detección está diseñado para detectarla.

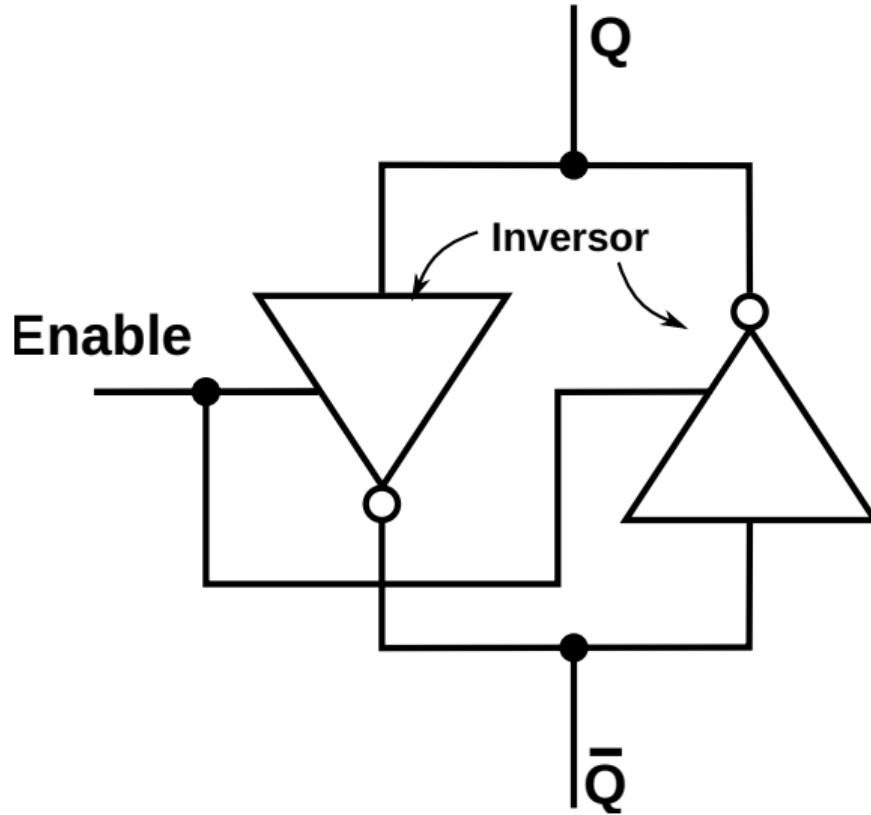
## Dentro del DRAM Device

- El amplificador de detección, en principio, *precarga* las bitlines con un valor eléctrico intermedio entre '0' y '1'.
- Al habilitarse una wordline, el capacitor de la bitline habilitada produce una alteración muy pequeña en el valor de tensión precargado, en mas o en menos dependiendo del estado de su carga.
- Esa diferencia es muy pequeña, pero el amplificador de detección está diseñado para detectarla.
- En función del sentido de la variación de tensión detectada el amplificador lleva la bitline a un '0' o a un '1'.

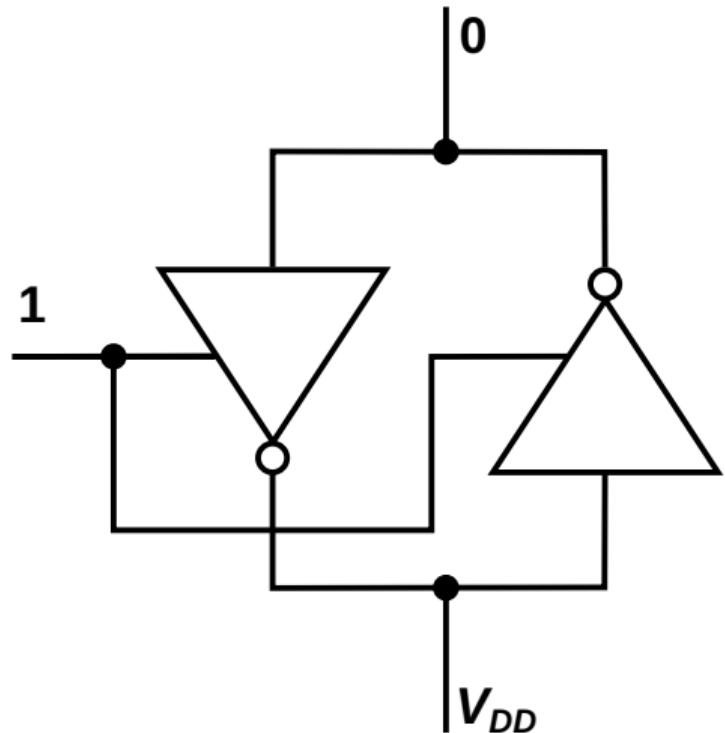
## Dentro del DRAM Device

- El amplificador de detección, en principio, *precarga* las bitlines con un valor eléctrico intermedio entre '0' y '1'.
- Al habilitarse una wordline, el capacitor de la bitline habilitada produce una alteración muy pequeña en el valor de tensión precargado, en mas o en menos dependiendo del estado de su carga.
- Esa diferencia es muy pequeña, pero el amplificador de detección está diseñado para detectarla.
- En función del sentido de la variación de tensión detectada el amplificador lleva la bitline a un '0' o a un '1'.
- Este pull a '0' o a '1' por parte del amplificador de detección, por otra parte restituye el valor de carga de los capacitores cuya carga se había perdido al modificar el nivel de tensión precargada en cada bitline.

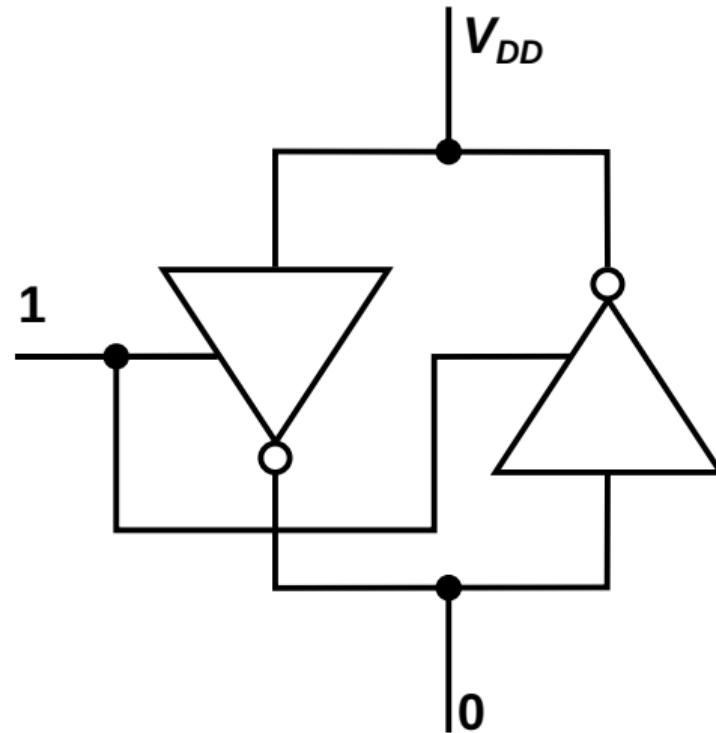
# El amplificador de detección



# El amplificador de detección

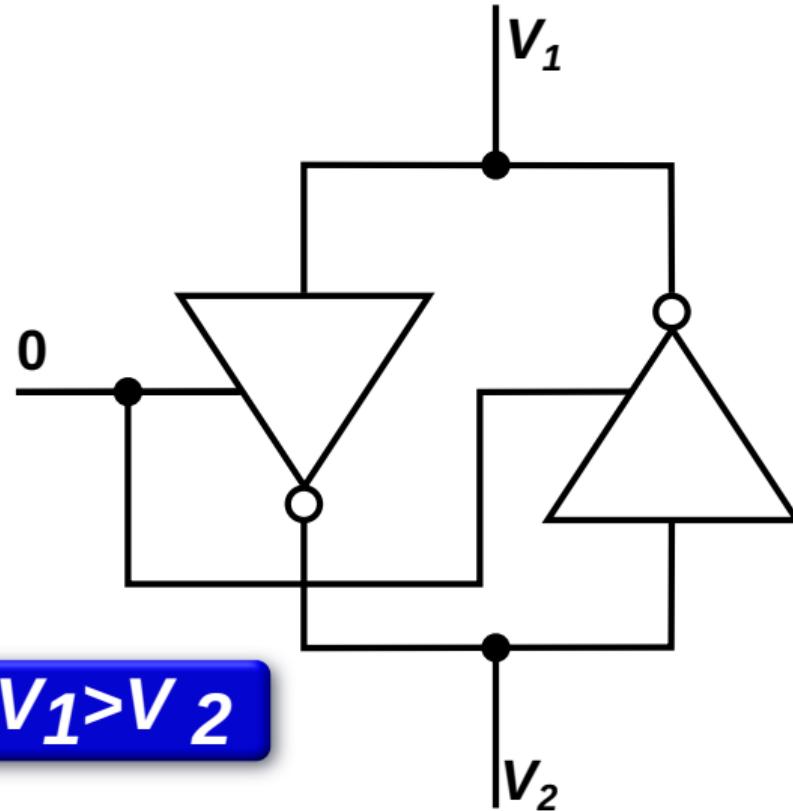


'0' Lógico

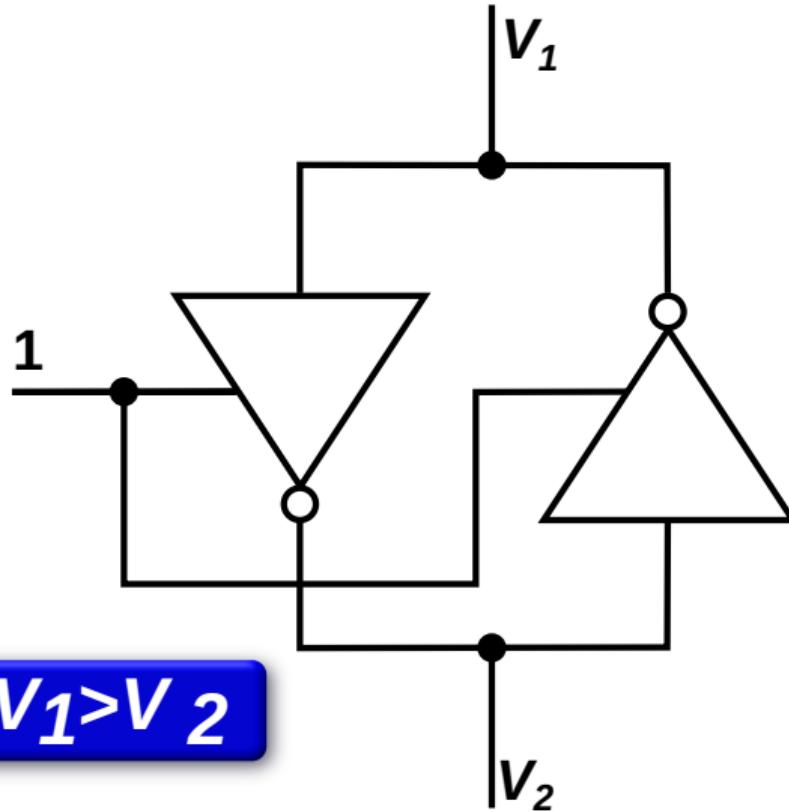


'1' Lógico

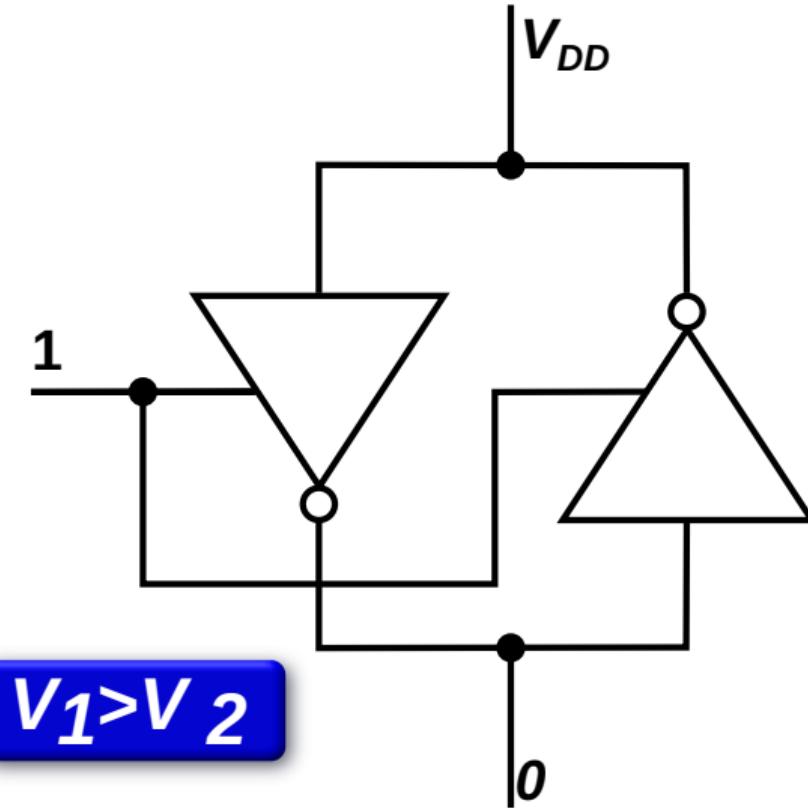
# El amplificador de detección



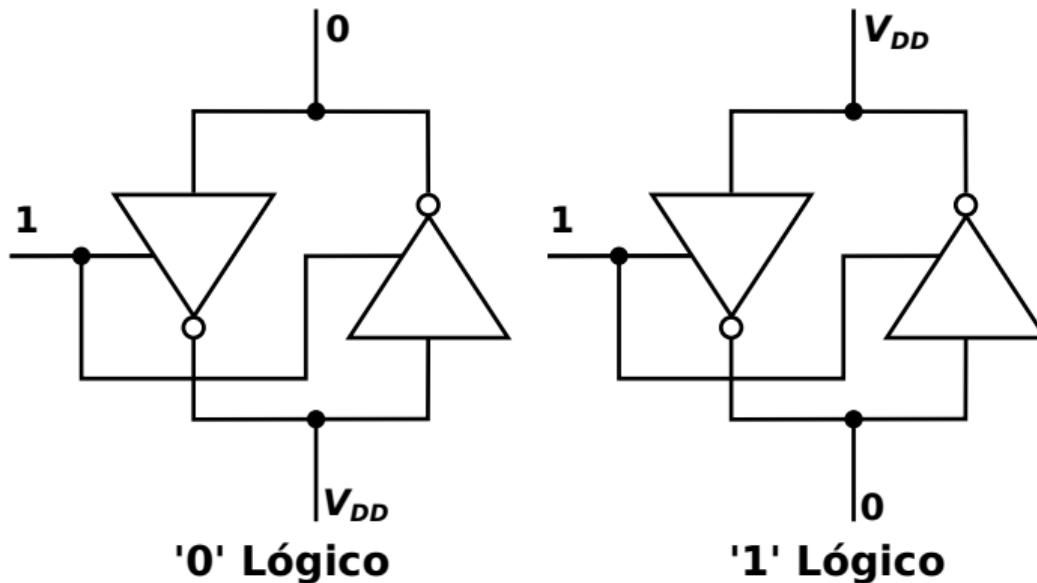
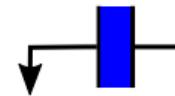
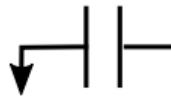
# El amplificador de detección



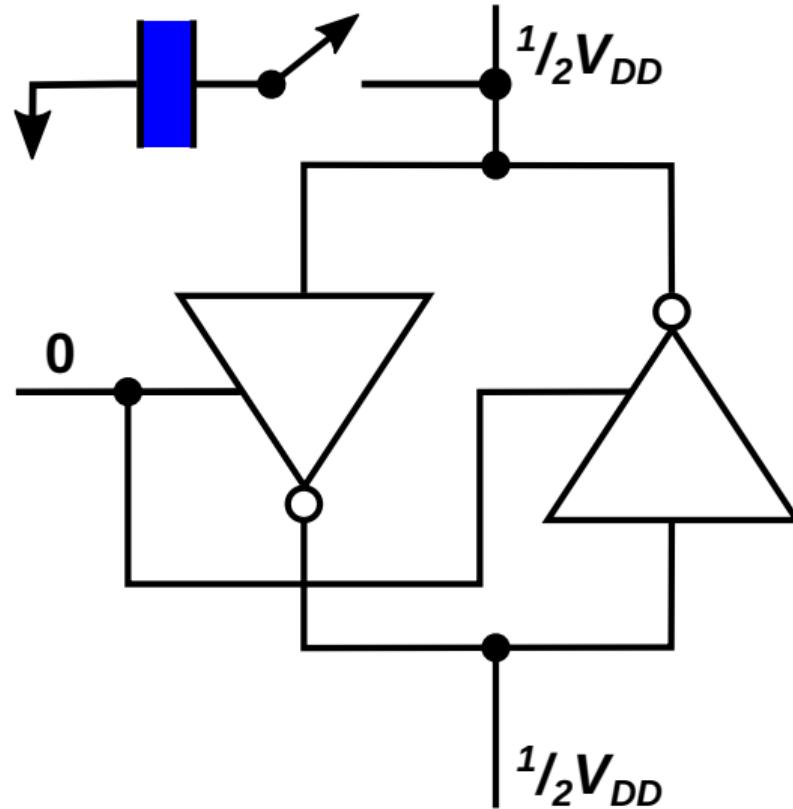
# El amplificador de detección



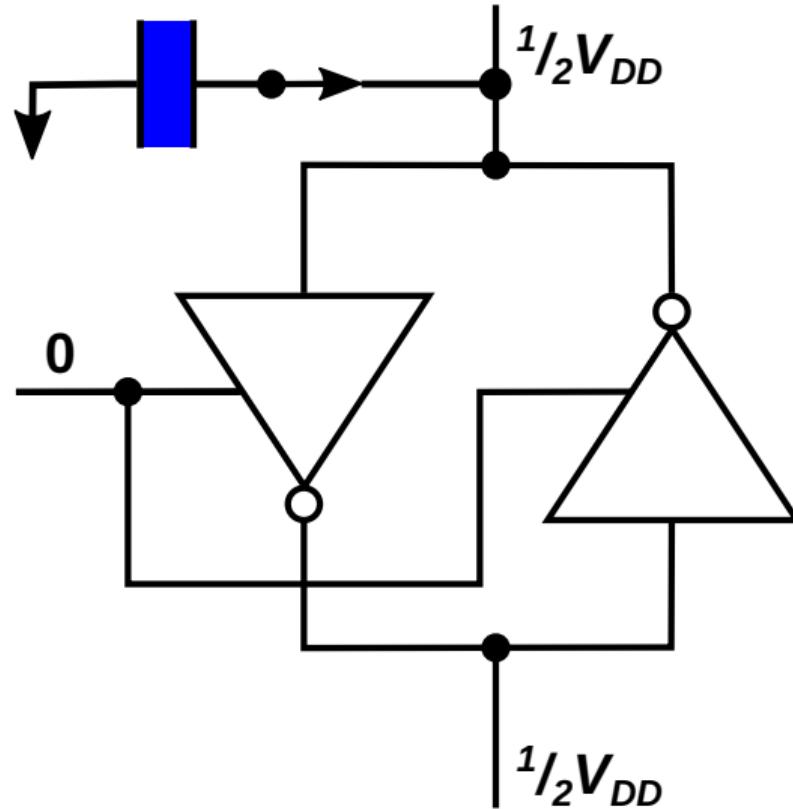
# El amplificador de detección



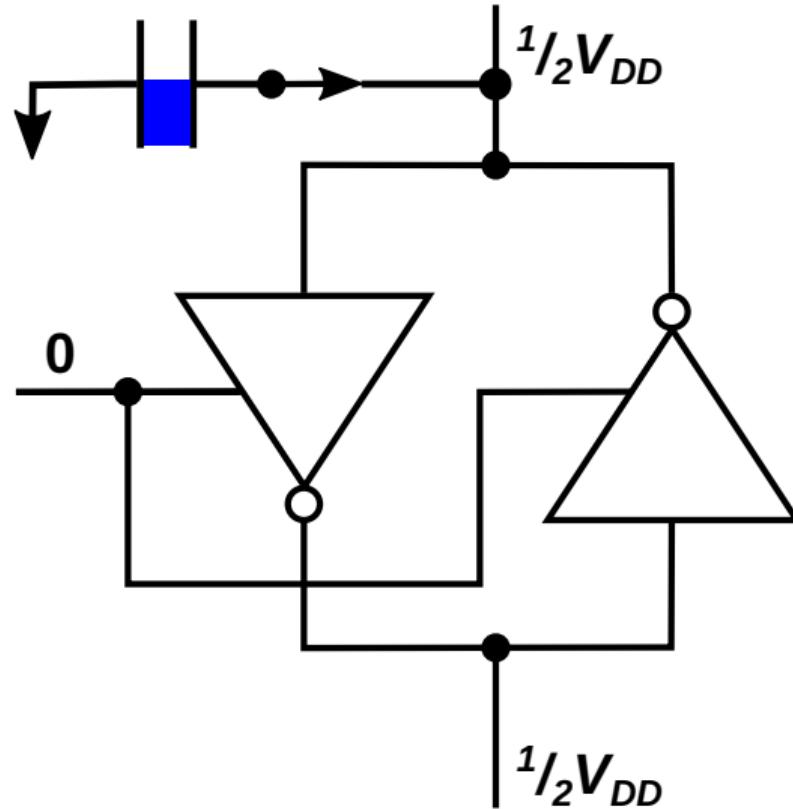
# El amplificador de detección



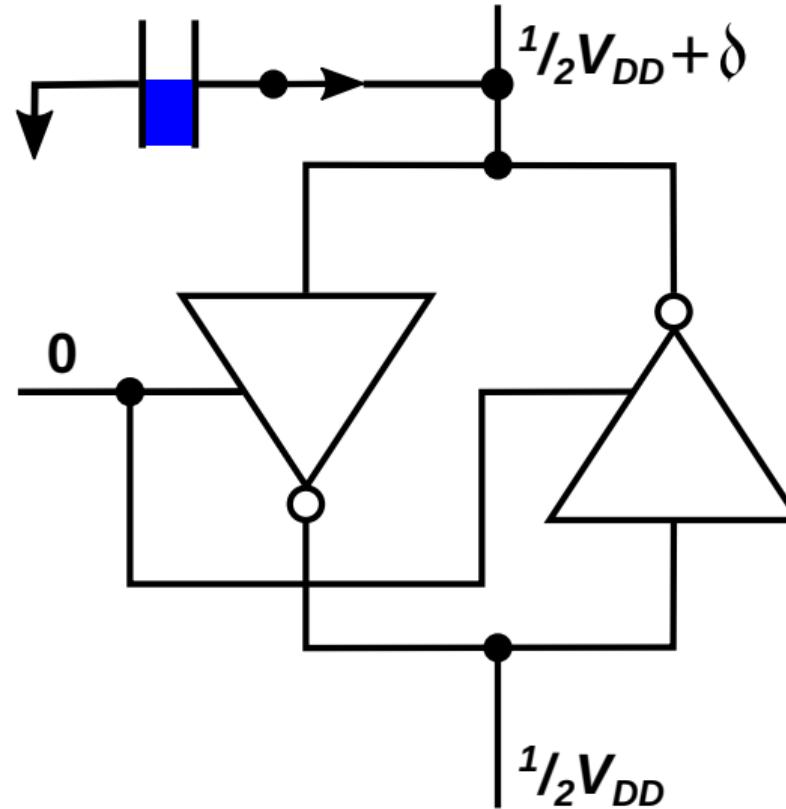
# El amplificador de detección



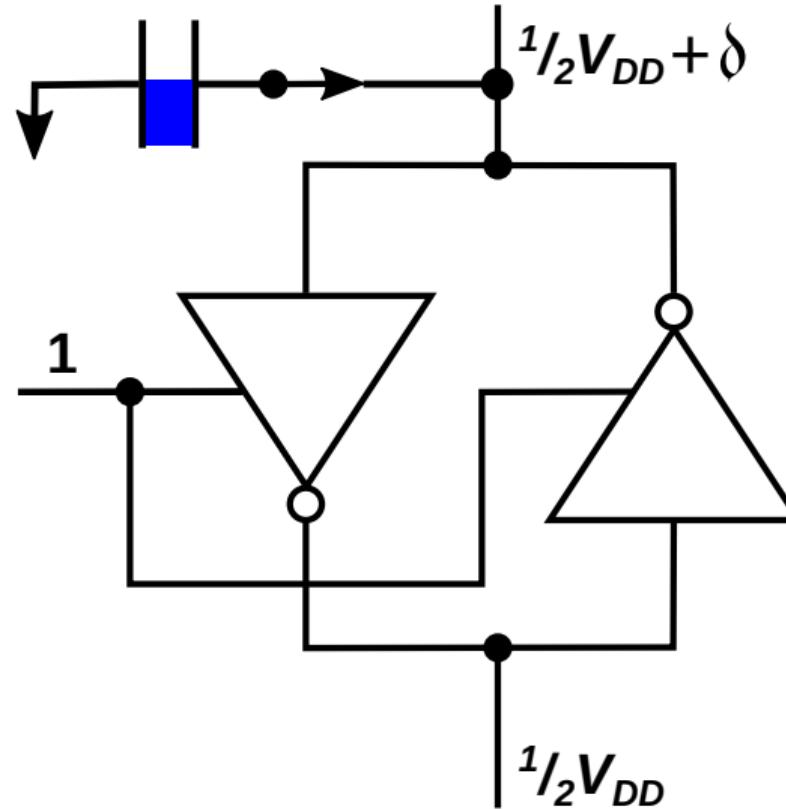
## El amplificador de detección



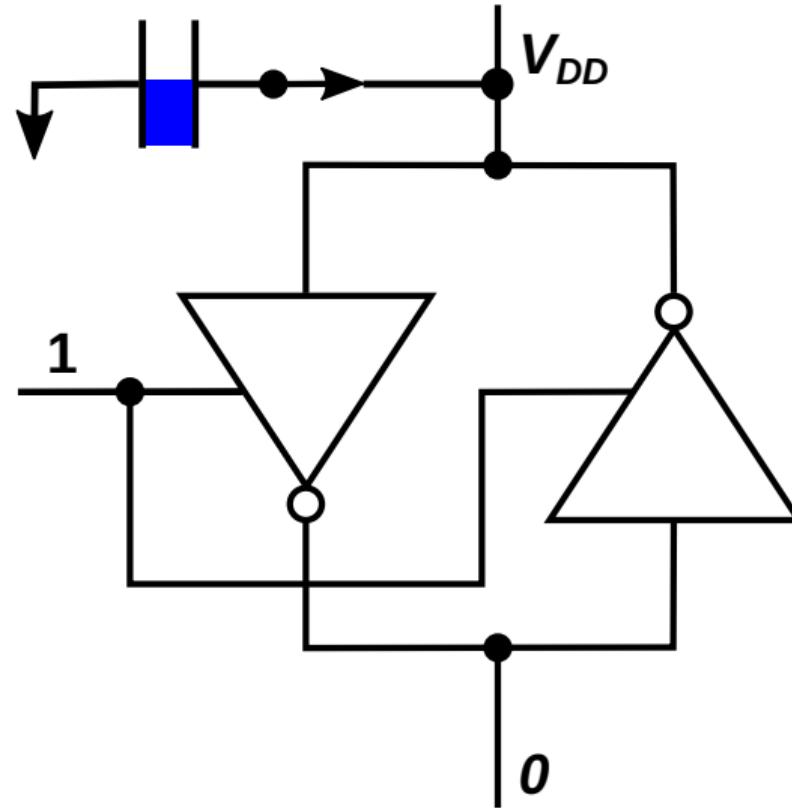
# El amplificador de detección



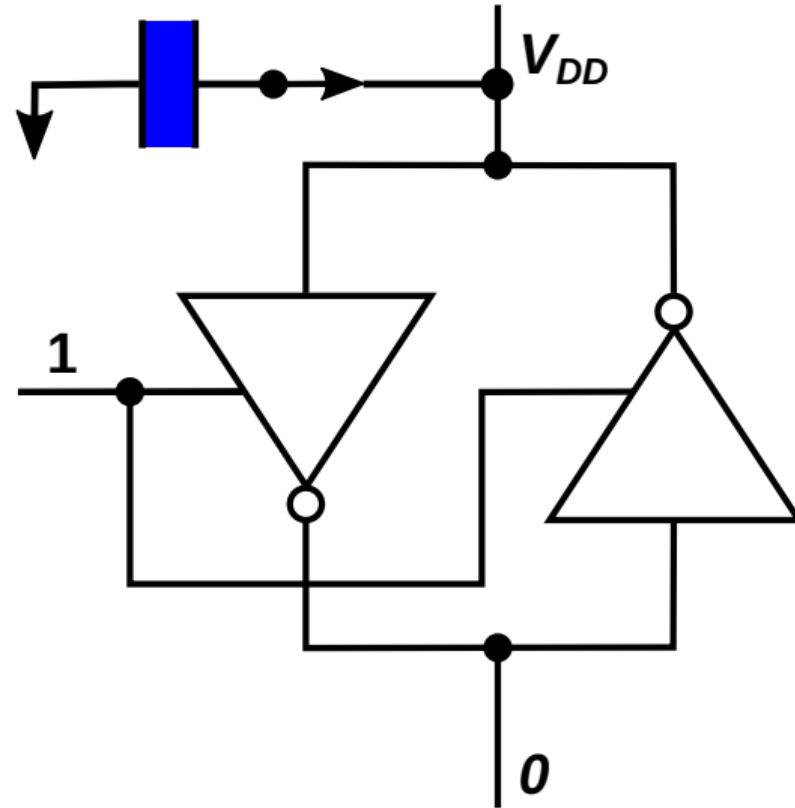
# El amplificador de detección



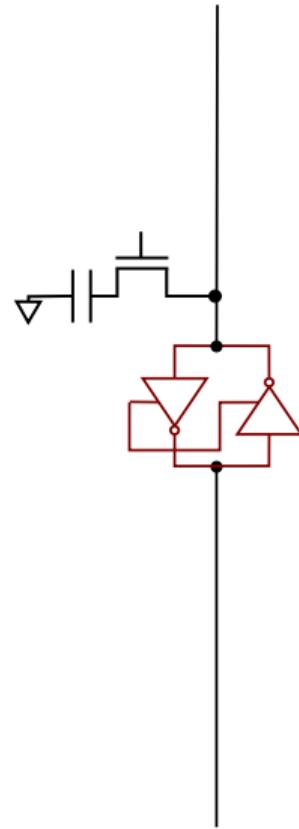
# El amplificador de detección



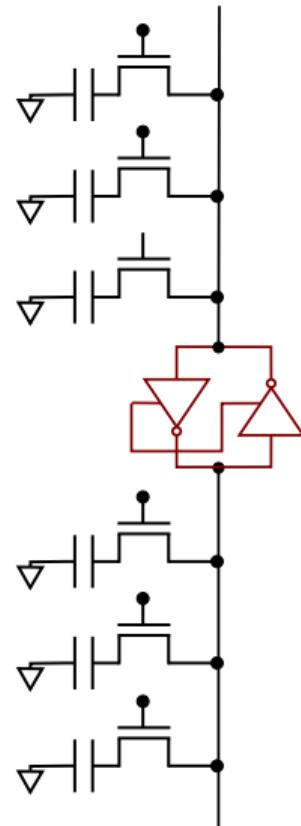
# El amplificador de detección



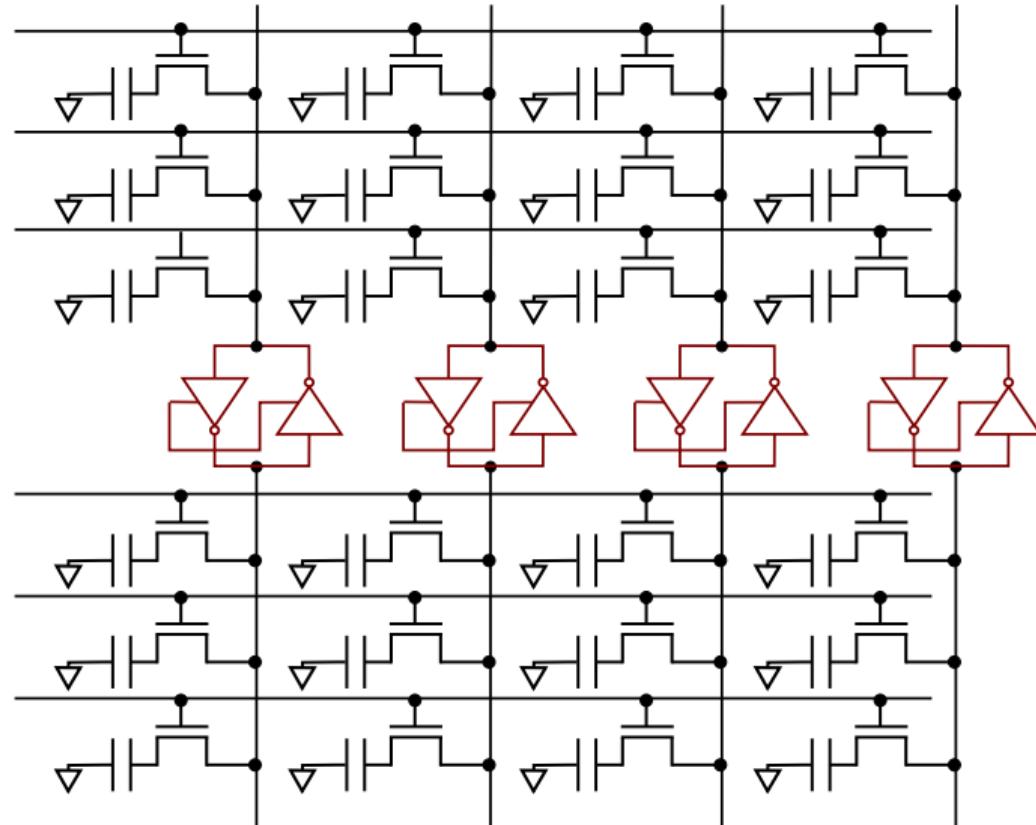
# Operación de la celda



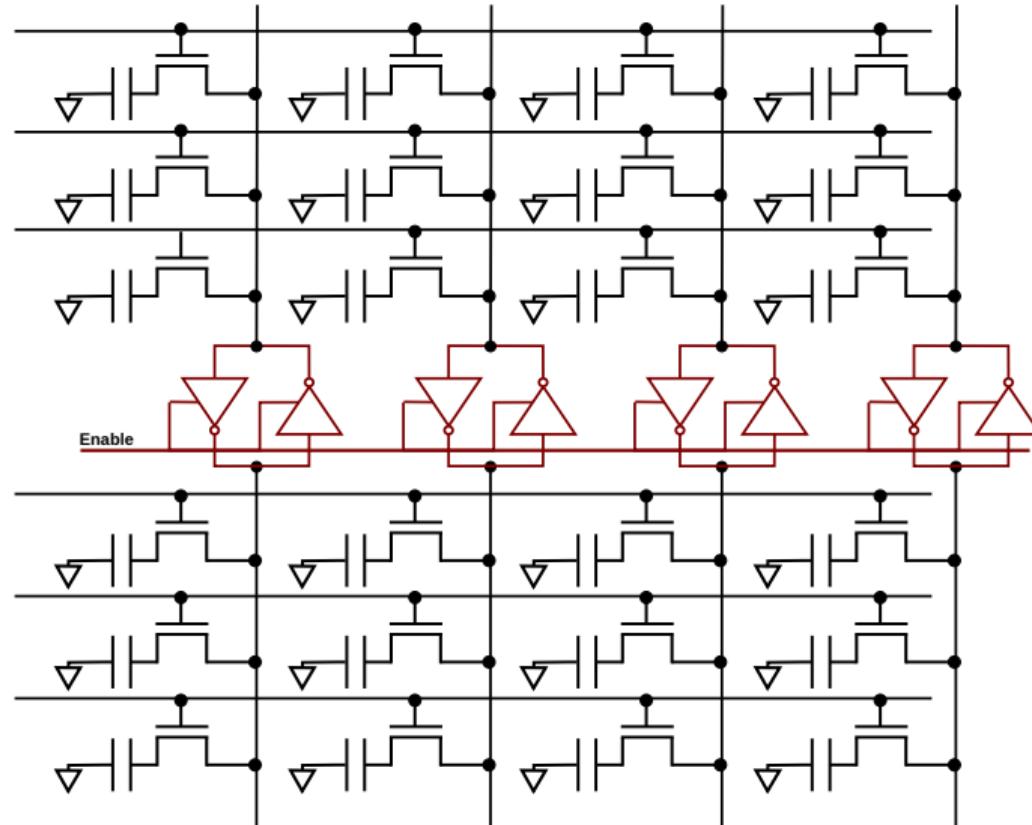
# Operación de la celda



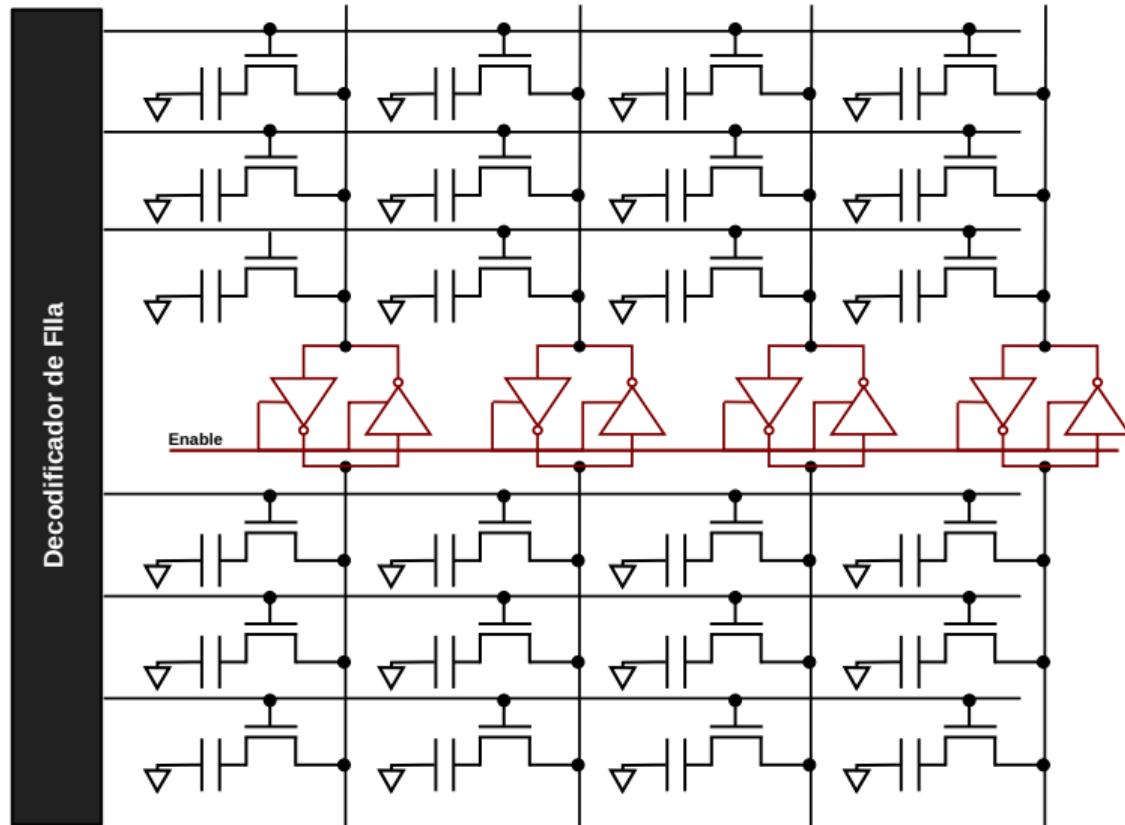
# Operación de la celda



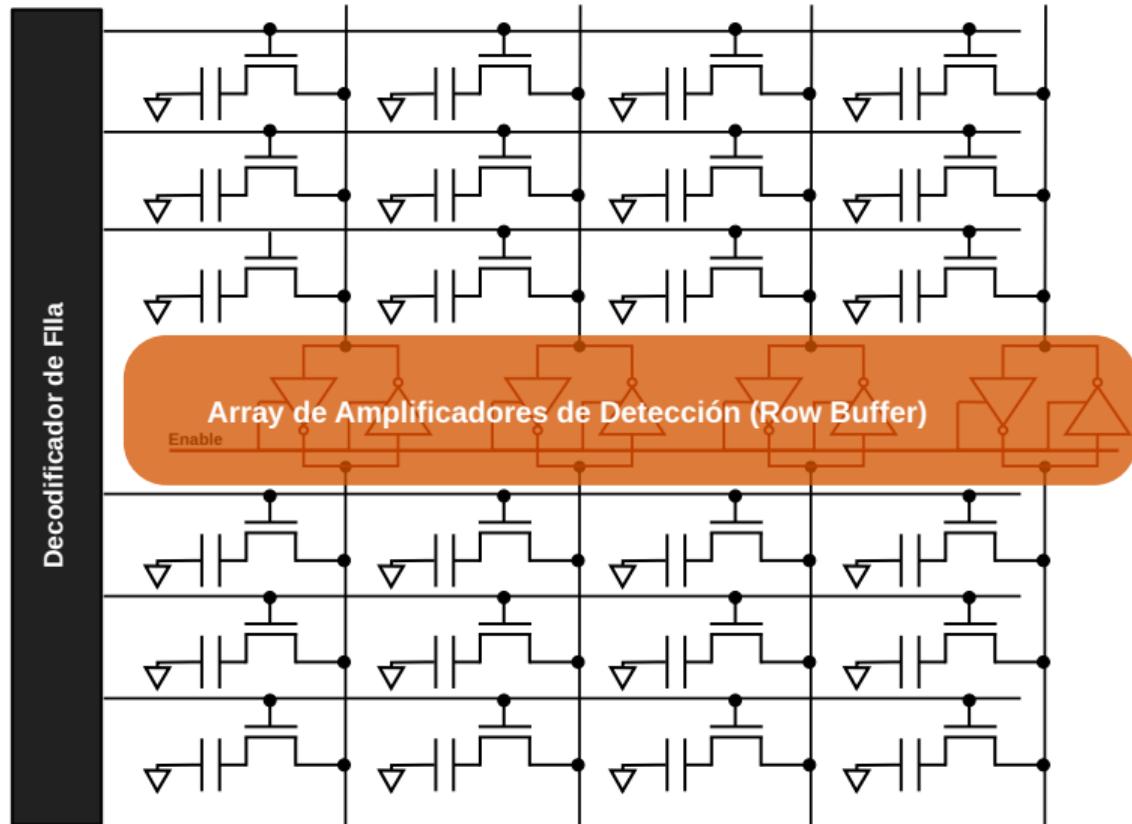
# Operación de la celda



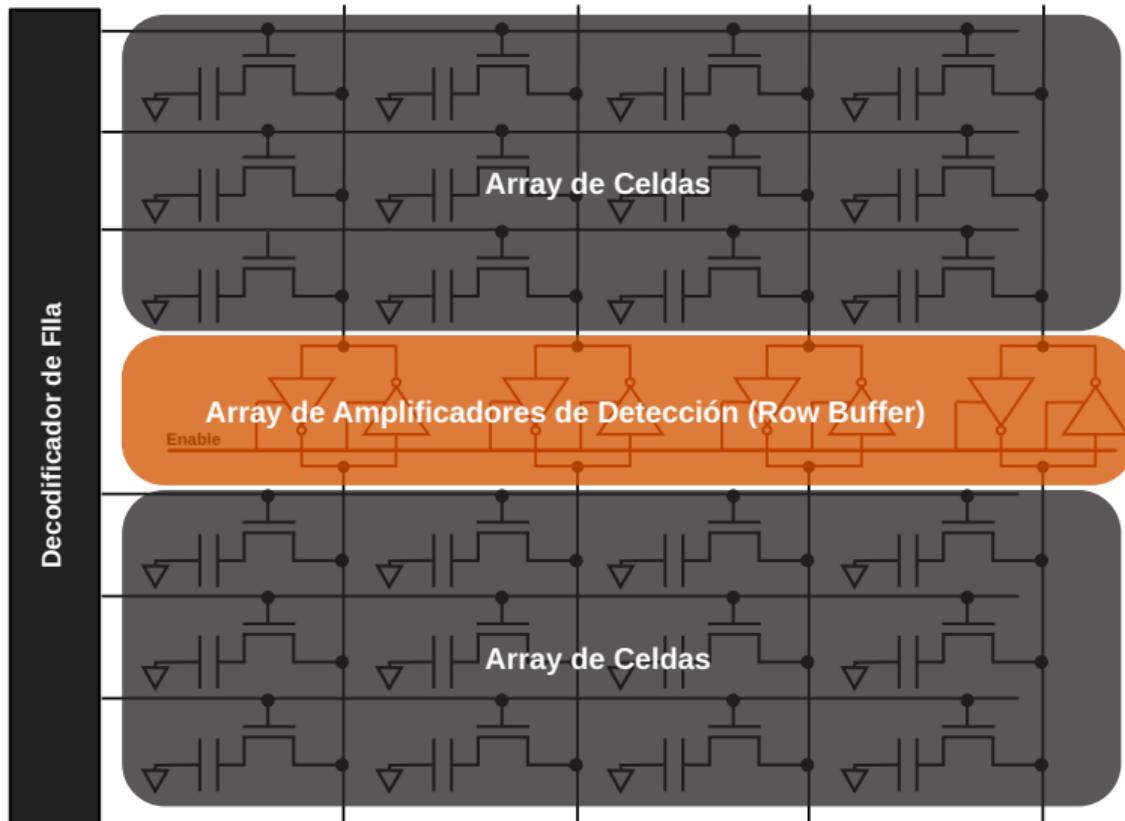
# Operación de la celda



# Operación de la celda



# Operación de la celda



# Acceso

- Con la dirección física recibida, el controlador de memoria decodifica el rango de memoria, el banco dentro de ese rango y fila y columna dentro del banco.

# Acceso

- Con la dirección física recibida, el controlador de memoria decodifica el rango de memoria, el banco dentro de ese rango y fila y columna dentro del banco.
- Los valores de fila y columna son ***row address*** y ***column address***.

# Acceso

- Con la dirección física recibida, el controlador de memoria decodifica el rango de memoria, el banco dentro de ese rango y fila y columna dentro del banco.
- Los valores de fila y columna son ***row address*** y ***column address***.
- Con el banco genera un grupo de bits adicionales a la dirección.

# Acceso

- Con la dirección física recibida, el controlador de memoria decodifica el rango de memoria, el banco dentro de ese rango y fila y columna dentro del banco.
- Los valores de fila y columna son ***row address*** y ***column address***.
- Con el banco genera un grupo de bits adicionales a la dirección.
- Con el rango genera el valor a colocar en el bus de chip select .

# Acceso

- Con la dirección física recibida, el controlador de memoria decodifica el rango de memoria, el banco dentro de ese rango y fila y columna dentro del banco.
- Los valores de fila y columna son ***row address*** y ***column address***.
- Con el banco genera un grupo de bits adicionales a la dirección.
- Con el rango genera el valor a colocar en el bus de chip select .
- Al recibir la información de direcciones el DRAM device precarga (es decir, pullea mediante sus amplificadores de detección a un valor intermedio entre '0' y '1') las bitlines del banco seleccionado.

# Acceso

- Con la dirección física recibida, el controlador de memoria decodifica el rango de memoria, el banco dentro de ese rango y fila y columna dentro del banco.
- Los valores de fila y columna son ***row address*** y ***column address***.
- Con el banco genera un grupo de bits adicionales a la dirección.
- Con el rango genera el valor a colocar en el bus de chip select .
- Al recibir la información de direcciones el DRAM device precarga (es decir, pullea mediante sus amplificadores de detección a un valor intermedio entre '0' y '1') las bitlines del banco seleccionado.
- Se activa el chip select seleccionando los DRAM devices correspondientes de acuerdo con la información puesta en el bus, y se activa la fila seleccionada con la ***row address*** activando el terminal RAS del DRAM Device.

# Acceso

- Con la dirección física recibida, el controlador de memoria decodifica el rango de memoria, el banco dentro de ese rango y fila y columna dentro del banco.
- Los valores de fila y columna son ***row address*** y ***column address***.
- Con el banco genera un grupo de bits adicionales a la dirección.
- Con el rango genera el valor a colocar en el bus de chip select .
- Al recibir la información de direcciones el DRAM device precarga (es decir, pullea mediante sus amplificadores de detección a un valor intermedio entre '0' y '1') las bitlines del banco seleccionado.
- Se activa el chip select seleccionando los DRAM devices correspondientes de acuerdo con la información puesta en el bus, y se activa la fila seleccionada con la ***row address*** activando el terminal RAS del DRAM Device.
- El DRAM Device selecciona la fila entera (cientos de celdas)

# Acceso

- Cada fila ataca a su correspondiente amplificador de detección dentro del array de amplificadores de detección,

# Acceso

- Cada fila ataca a su correspondiente amplificador de detección dentro del array de amplificadores de detección,
- Cada amplificador detecta el  $\delta$  de tensión aplicada por el capacitor conectado a su bitline a través del transistor, y fuerza (“pull”) el valor lógico definitivo. Esto lleva en las tecnologías actuales típicamente decenas de nanosegundos.

# Acceso

- Cada fila ataca a su correspondiente amplificador de detección dentro del array de amplificadores de detección,
- Cada amplificador detecta el  $\delta$  de tensión aplicada por el capacitor conectado a su bitline a través del transistor, y fuerza (“pull”) el valor lógico definitivo. Esto lleva en las tecnologías actuales típicamente decenas de nanosegundos.
- Se activa la señal CAS, que activa la wordline deseada del banco seleccionado, y selecciona a los amplificadores de detección que deben ser conectados al row buffer de salida.

# Acceso

- Cada fila ataca a su correspondiente amplificador de detección dentro del array de amplificadores de detección,
- Cada amplificador detecta el  $\delta$  de tensión aplicada por el capacitor conectado a su bitline a través del transistor, y fuerza (“pull”) el valor lógico definitivo. Esto lleva en las tecnologías actuales típicamente decenas de nanosegundos.
- Se activa la señal CAS, que activa la wordline deseada del banco seleccionado, y selecciona a los amplificadores de detección que deben ser conectados al row buffer de salida.
- El controlador de memoria recibe los datos y los retransmite al procesador.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

### ● Estado del arte

#### ● JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

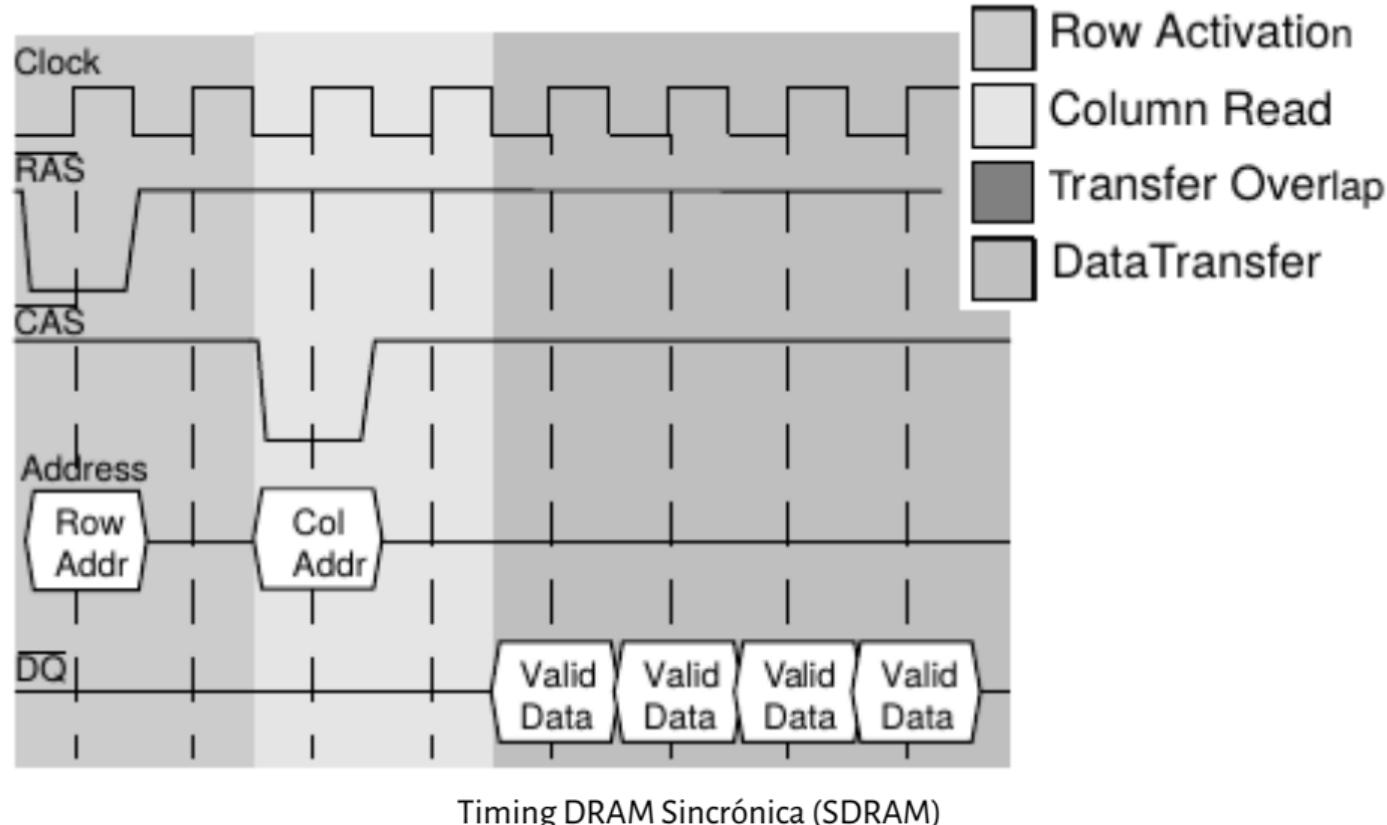
## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

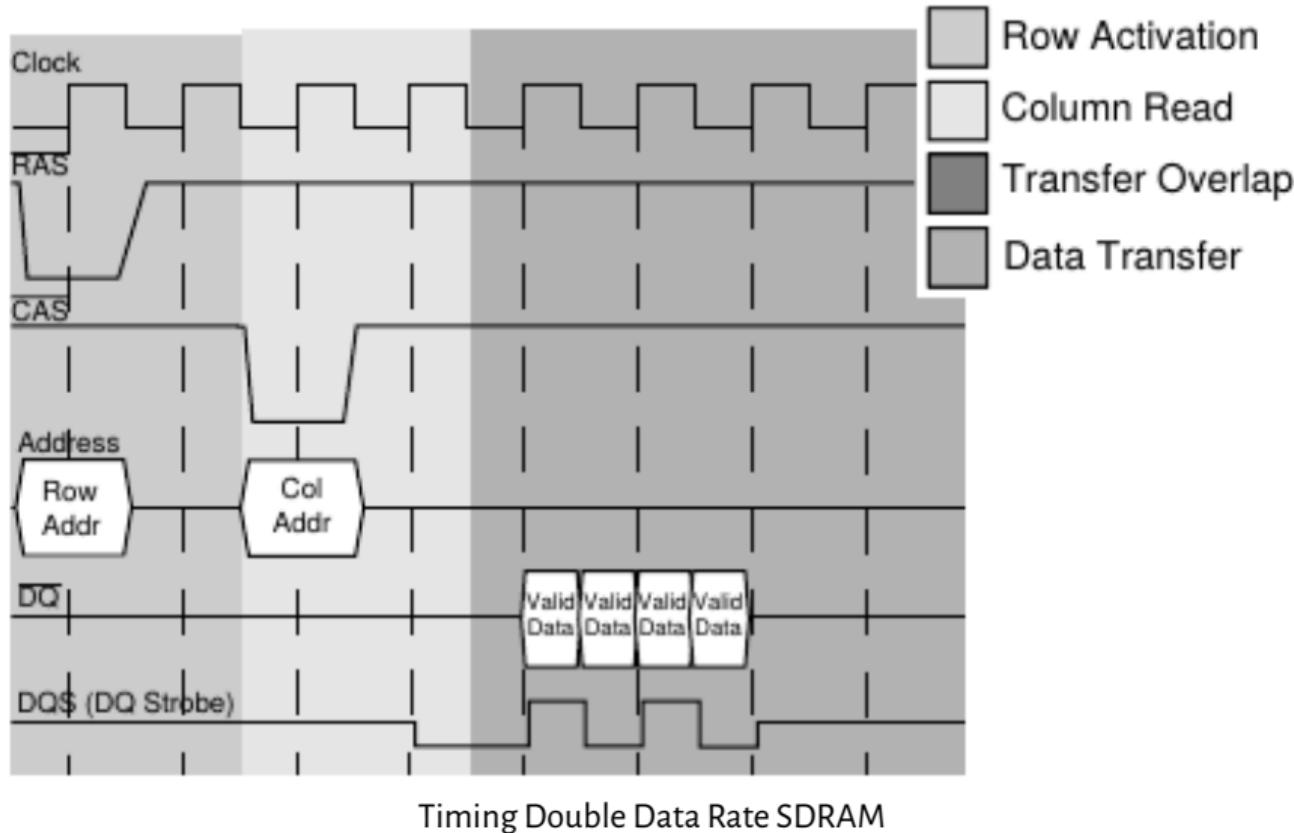
# Tecnologías orientadas al Throughput

- **SDRAM Synchronous DRAM:** Todas las generaciones anteriores eran asincrónicas. Sincronizar hace posible aplicar CAS y RAS al mismo tiempo, ya que se dispone de latches internos que aseguran la información que se aplica a la matriz. Esto deriva en que el tiempo de acceso es mas predecible, por independizar el acceso cada dispositivo DRAM del resto (de-skewing), y minimiza el tiempo de conmutación de un DIMM al próximo (cuando se tienen varios DIMMs). Conserva el burst mode introducido por BEDO. Los SDRAM devices tienen un registro para programar el números de accesos burst. Esto evita enviar un CAS por cada lectura reduciendo el ancho de banda de bus dedicado a handshake aumentando el porcentaje relativo dedicado a transferir datos.

# Tecnologías orientadas al Throughput



# Interfaces orientadas al Throughput



# Interfaces orientadas al Throughput

# Interfaces orientadas al Throughput

- **DDR-SDRAM Double Data Rate SDRAM:** Utiliza ambos flancos de la señal para transferir datos.

# Interfaces orientadas al Throughput

- **DDR-SDRAM Double Data Rate SDRAM:** Utiliza ambos flancos de la señal para transferir datos.
- En este caso al ser una memoria Sincrónica, se utilizan ambos flancos de la señal de clock.

# Interfaces orientadas al Throughput

- **DDR-SDRAM Double Data Rate SDRAM:** Utiliza ambos flancos de la señal para transferir datos.
- En este caso al ser una memoria Síncrona, se utilizan ambos flancos de la señal de clock.
- Sus restantes características son similares a la SDRAM: utiliza la misma tecnología de señalización, la misma especificación de interfaz, y los mismos pinouts en los DIMM carriers. Sin embargo tienen el doble de velocidad de transferencia que las SDRAM.

# Interfaces orientadas al Throughput

- **DDR-SDRAM Double Data Rate SDRAM:** Utiliza ambos flancos de la señal para transferir datos.
- En este caso al ser una memoria Sincrónica, se utilizan ambos flancos de la señal de clock.
- Sus restantes características son similares a la SDRAM: utiliza la misma tecnología de señalización, la misma especificación de interfaz, y los mismos pinouts en los DIMM carriers. Sin embargo tienen el doble de velocidad de transferencia que las SDRAM.
- Durante las escrituras se prescinde de la señal de clock y se utiliza una señal DQS en cuyos flancos ascendente y descendente se escriben los datos.

# Interfaces orientadas al Throughput

- **DDR-SDRAM Double Data Rate SDRAM:** Utiliza ambos flancos de la señal para transferir datos.
- En este caso al ser una memoria Sincrónica, se utilizan ambos flancos de la señal de clock.
- Sus restantes características son similares a la SDRAM: utiliza la misma tecnología de señalización, la misma especificación de interfaz, y los mismos pinouts en los DIMM carriers. Sin embargo tienen el doble de velocidad de transferencia que las SDRAM.
- Durante las escrituras se prescinde de la señal de clock y se utiliza una señal DQS en cuyos flancos ascendente y descendente se escriben los datos.
- Esto contradice lo estipulado por JEDEC para SDRAMs y hace que las DDR se asimilen al standard IBM Toggle.

# Standard JEDEC

- Las DRAM actualmente son un comodity.

# Standard JEDEC

- Las DRAM actualmente son un comodity.
- Cualquier DRAM o DIMM actual tiene especificaciones equivalentes: ancho de bus, capacidad, velocidad, interfaz, por citar las mas importantes.

# Standard JEDEC

- Las DRAM actualmente son un comodity.
- Cualquier DRAM o DIMM actual tiene especificaciones equivalentes: ancho de bus, capacidad, velocidad, interfaz, por citar las mas importantes.
- Esta compatibilidad está gobernada por JEDEC (**J**oint **E**lectron **D**evice **E**ngineering **C**ouncil).

## Standard JEDEC

- Las DRAM actualmente son un commodity.
- Cualquier DRAM o DIMM actual tiene especificaciones equivalentes: ancho de bus, capacidad, velocidad, interfaz, por citar las mas importantes.
- Esta compatibilidad está gobernada por JEDEC (**J**oint **E**lectron **D**evice **E**ngineering **C**ouncil).
- Hemos mencionado ya la presencia de un controlador de memoria.

# Standard JEDEC

- Las DRAM actualmente son un commodity.
- Cualquier DRAM o DIMM actual tiene especificaciones equivalentes: ancho de bus, capacidad, velocidad, interfaz, por citar las mas importantes.
- Esta compatibilidad está gobernada por JEDEC (**J**oint **E**lectron **D**evice **E**ngineering **C**ouncil).
- Hemos mencionado ya la presencia de un controlador de memoria.
- Entre otros aspectos se definen cuatro buses independientes para una organización de memoria: *data, address, control, y chip select*.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte

## ● JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

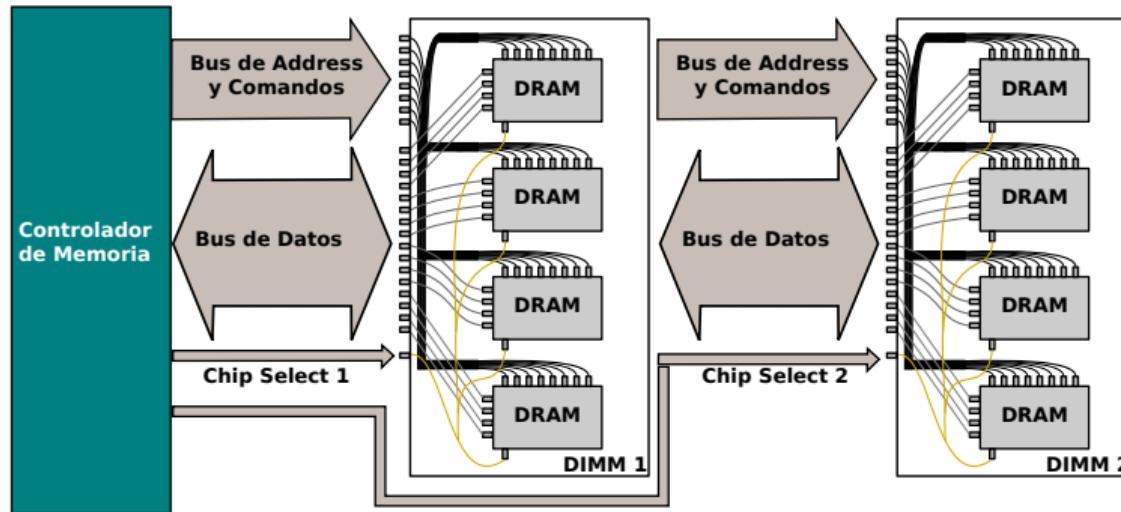
## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

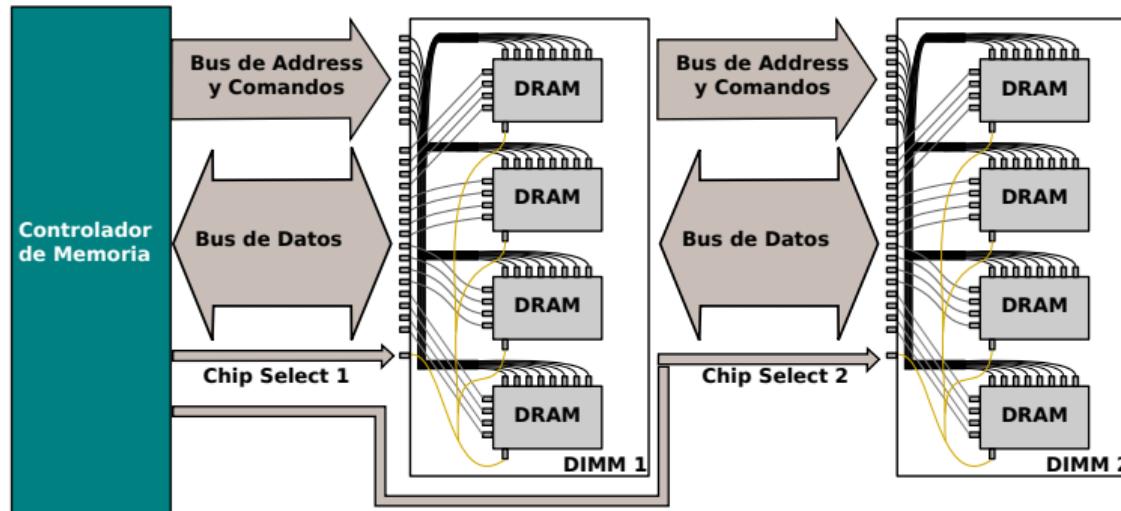
- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Estándares: JEDEC



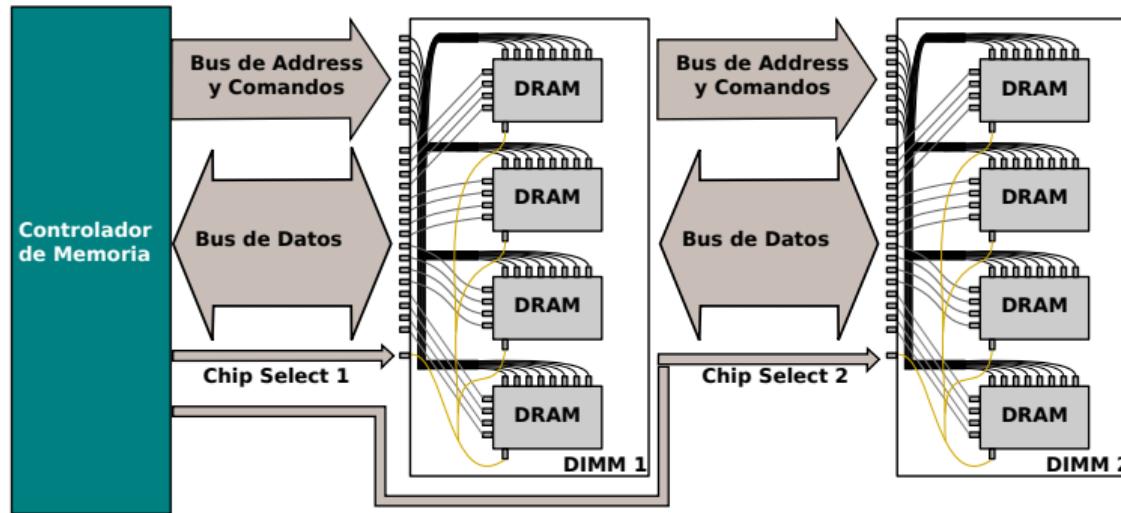
- El *data bus* por razones de optimización del ancho de banda se definió de 64 bits.

# Estándares: JEDEC



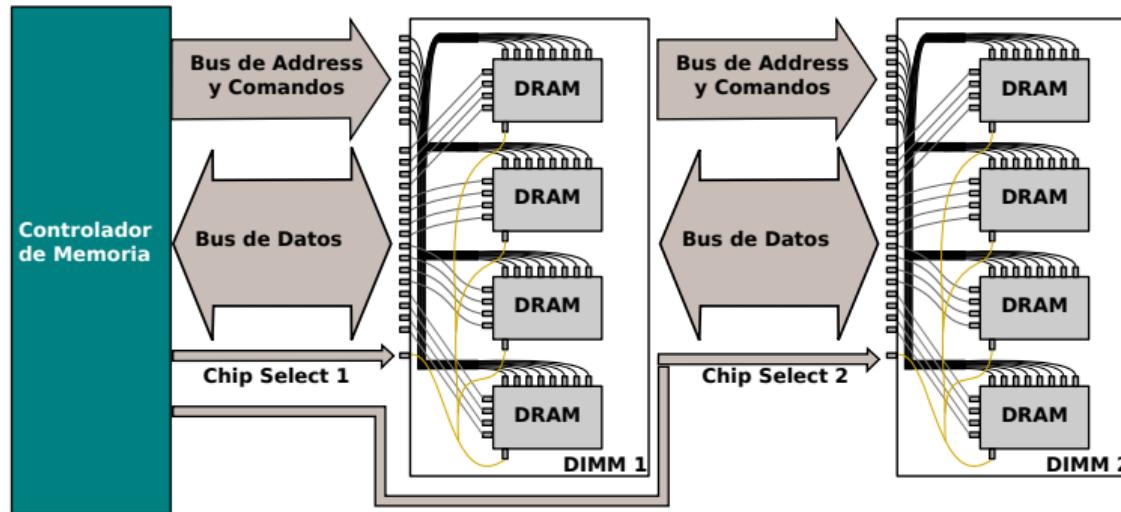
- El *data bus* por razones de optimización del ancho de banda se definió de 64 bits.
- En sistemas de alto rendimiento (HBM por ejemplo) puede tener hasta 256 bits.

# Estándares: JEDEC



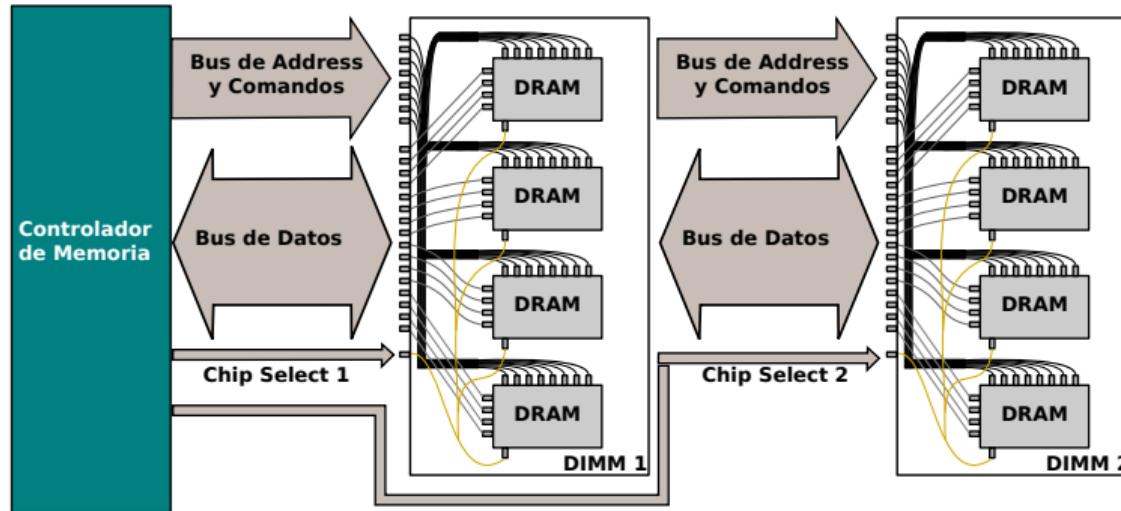
- El *address* bus envía la información de row y col a los DIMMs. Su ancho es proporcional a la capacidad de DRAM instalada

# Estándares: JEDEC



- El *address* bus envía la información de row y col a los DIMMs. Su ancho es proporcional a la capacidad de DRAM instalada
- El *control* bus lleva los strobes de row y col, output enable, clock enable, clock, etc.

# Estándares: JEDEC



- El *address* bus envía la información de row y col a los DIMMs. Su ancho es proporcional a la capacidad de DRAM instalada
- El *control* bus lleva los strobes de row y col, output enable, clock enable, clock, etc.
- El *chip select* bus es individual para cada rango. Ej: 2 rangos por DIMM, implican dos *chip select* bus por cada DIMM.

# Topología básica

- Los procesadores un sistema no se conectan directamente a los dispositivos de memoria DRAM, sino que intermedian uno o mas controladores de memoria.

# Topología básica

- Los procesadores un sistema no se conectan directamente a los dispositivos de memoria DRAM, sino que intermedian uno o mas controladores de memoria.
- Un procesador puede conectarse a uno o mas controladores, o varios procesadores pueden utilizar un único controlador.

# Topología básica

- Los procesadores un sistema no se conectan directamente a los dispositivos de memoria DRAM, sino que intermedian uno o mas controladores de memoria.
- Un procesador puede conectarse a uno o mas controladores, o varios procesadores pueden utilizar un único controlador.
- Las redes de conexionado (bus, crossbar, etc.) entre el procesador y el controlador de memoria son diferentes de las que conectan al controlador con la DRAM (maneján diferentes señales).

# Topología básica

- Los procesadores un sistema no se conectan directamente a los dispositivos de memoria DRAM, sino que intermedian uno o mas controladores de memoria.
- Un procesador puede conectarse a uno o mas controladores, o varios procesadores pueden utilizar un único controlador.
- Las redes de conexionado (bus, crossbar, etc.) entre el procesador y el controlador de memoria son diferentes de las que conectan al controlador con la DRAM (manejan diferentes señales).
- De este modo el procesador se independiza de las características físicas de los DRAM devices.

# Topología básica

- Los procesadores un sistema no se conectan directamente a los dispositivos de memoria DRAM, sino que intermedian uno o mas controladores de memoria.
- Un procesador puede conectarse a uno o mas controladores, o varios procesadores pueden utilizar un único controlador.
- Las redes de conexionado (bus, crossbar, etc.) entre el procesador y el controlador de memoria son diferentes de las que conectan al controlador con la DRAM (manejan diferentes señales).
- De este modo el procesador se independiza de las características físicas de los DRAM devices.
- Un controlador de memoria se conecta a uno o mas DIMMs, o lo que es lo mismo, controla varios DRAM devices.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

### ● Introducción General

- Arquitectura
- Configuración
- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Porque usamos un controlador de memoria

- Maneja la transferencia de datos hacia y desde la memoria asegurando la compatibilidad con el protocolo.

# Porque usamos un controlador de memoria

- Maneja la transferencia de datos hacia y desde la memoria asegurando la compatibilidad con el protocolo.
- Independiza al procesador de las características eléctricas de los DRAM Devices que conforman el sistema de memoria.

## Porque usamos un controlador de memoria

- Maneja la transferencia de datos hacia y desde la memoria asegurando la compatibilidad con el protocolo.
- Independiza al procesador de las características eléctricas de los DRAM Devices que conforman el sistema de memoria.
- Asegura el cumplimiento de los requerimientos de timing de los DRAM Devices independiente-mente del procesador base del sistema.

## Porque usamos un controlador de memoria

- Maneja la transferencia de datos hacia y desde la memoria asegurando la compatibilidad con el protocolo.
- Independiza al procesador de las características eléctricas de los DRAM Devices que conforman el sistema de memoria.
- Asegura el cumplimiento de los requerimientos de timing de los DRAM Devices independiente-mente del procesador base del sistema.
- Permite arbitrar el acceso a un único subsistema de memoria por parte de múltiples cores de pro-cesamiento, aun si estos son heterogéneos.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General

## ● Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

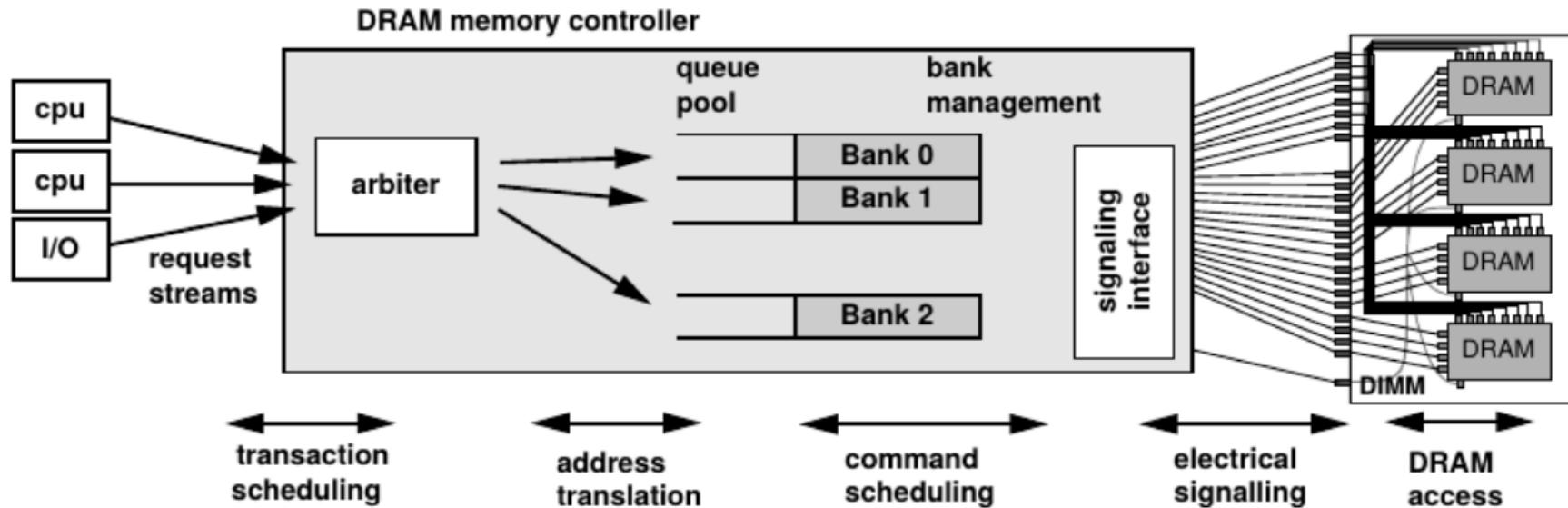
## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Arquitectura de un Controlador Genérico



# Arbitración de requerimientos

- Los costos en acceso a memoria siguen aumentando relativamente los costos computacionales de muchos algoritmos.

# Arbitración de requerimientos

- Los costos en acceso a memoria siguen aumentando relativamente los costos computacionales de muchos algoritmos.
- La仲裁ación de requerimientos es un aspecto crítico para la performance del sistema de memoria.

# Arbitración de requerimientos

- Los costos en acceso a memoria siguen aumentando relativamente los costos computacionales de muchos algoritmos.
- La仲裁ación de requerimientos es un aspecto crítico para la performance del sistema de memoria.
- Resuelve en base a las prioridades de los requerimientos de memoria de parte de los cores y de dispositivos de E/S como un DMA controller por ejemplo, cual es el que debe transmitirse a los DRAM devices.

# Arbitración de requerimientos

- Los costos en acceso a memoria siguen aumentando relativamente los costos computacionales de muchos algoritmos.
- La仲裁ación de requerimientos es un aspecto crítico para la performance del sistema de memoria.
- Resuelve en base a las prioridades de los requerimientos de memoria de parte de los cores y de dispositivos de E/S como un DMA controller por ejemplo, cual es el que debe transmitirse a los DRAM devices.
- No siempre pasa el de mayor prioridad. Puede darse que un requerimiento de baja prioridad arribe al controlador en el momento en que la fila que contiene sus datos está activa. En tal caso algunos controladores resuelven primero este requerimiento a pesar de su prioridad menor en virtud de la performance general.

# Traducción de Direcciones

- Cuando una transacción gana la arbitración, pasa al controlador que traduce la dirección física del bus del procesador, a la dirección del Bus del controlador de memoria: Row & Column.

# Traducción de Direcciones

- Cuando una transacción gana la arbitración, pasa al controlador que traduce la dirección física del bus del procesador, a la dirección del Bus del controlador de memoria: Row & Column.
- En función de la operación requerida desde el core o desde el device de E/S sobre esa dirección, la etapa de traducción arma la secuencia de comandos necesaria para enviar a la DRAM.

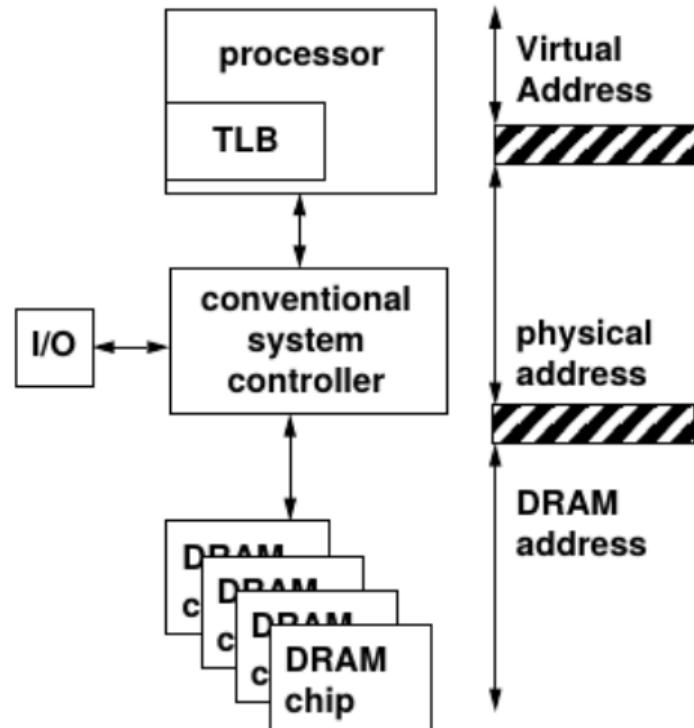
# Traducción de Direcciones

- Cuando una transacción gana la arbitración, pasa al controlador que traduce la dirección física del bus del procesador, a la dirección del Bus del controlador de memoria: Row & Column.
- En función de la operación requerida desde el core o desde el device de E/S sobre esa dirección, la etapa de traducción arma la secuencia de comandos necesaria para enviar a la DRAM.
- Los DRAM devices relativamente modernos tienen buffers de comandos como etapa previa a los amplificadores de detección.

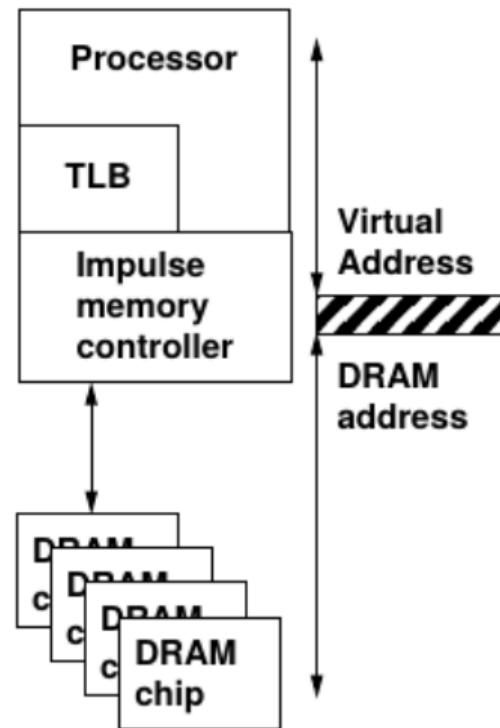
# Traducción de Direcciones

- Cuando una transacción gana la arbitración, pasa al controlador que traduce la dirección física del bus del procesador, a la dirección del Bus del controlador de memoria: Row & Column.
- En función de la operación requerida desde el core o desde el device de E/S sobre esa dirección, la etapa de traducción arma la secuencia de comandos necesaria para enviar a la DRAM.
- Los DRAM devices relativamente modernos tienen buffers de comandos como etapa previa a los amplificadores de detección.
- La política de buffering es crítica en el diseño de cada DRAM Device en virtud de su efecto sobre la performance, o sobre el consumo. El análisis del trade on y trade off de estos aspectos son moneda corriente entre los diseñadores de estos dispositivos.

# Traducción de direcciones



Conventional System Architecture



Impulse Memory System Architecture

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Lógica de control de un SDRAM Device

- En esencia los DRAM Devices son circuitos analógicos cuya temporización es naturalmente asincrónica.

# Lógica de control de un SDRAM Device

- En esencia los DRAM Devices son circuitos analógicos cuya temporización es naturalmente asincrónica.
- Los pasos que le toma a la circuitería de una DRAM para almacenar y recuperar datos en forma de carga en un capacitor a través de amplificadores de detección, le insumen una latencia considerable, la cual se expresa normalmente en nseg., y no en ciclos de clock.

# Lógica de control de un SDRAM Device

- En esencia los DRAM Devices son circuitos analógicos cuya temporización es naturalmente asincrónica.
- Los pasos que le toma a la circuitería de una DRAM para almacenar y recuperar datos en forma de carga en un capacitor a través de amplificadores de detección, le insumen una latencia considerable, la cual se expresa normalmente en nseg., y no en ciclos de clock.
- Por eso de acuerdo al diseño y al proceso de fabricación cada DRAM tiene un juego de parámetros de timing diferentes.

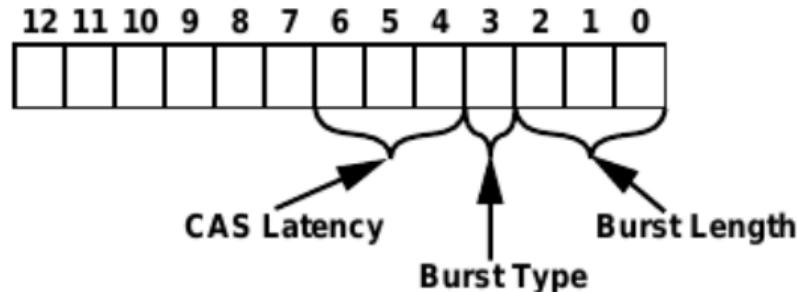
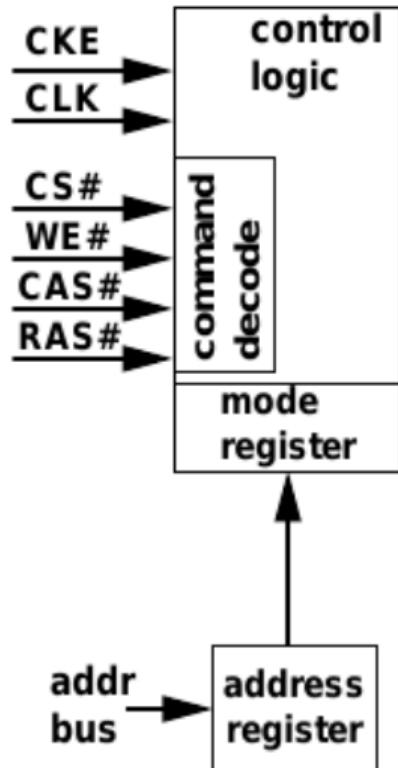
# Lógica de control de un SDRAM Device

- En esencia los DRAM Devices son circuitos analógicos cuya temporización es naturalmente asincrónica.
- Los pasos que le toma a la circuitería de una DRAM para almacenar y recuperar datos en forma de carga en un capacitor a través de amplificadores de detección, le insumen una latencia considerable, la cual se expresa normalmente en nseg., y no en ciclos de clock.
- Por eso de acuerdo al diseño y al proceso de fabricación cada DRAM tiene un juego de parámetros de timing diferentes.
- La solución que encontró la industria es proveerles una interfaz sincrónica.

# Lógica de control de un SDRAM Device

- En esencia los DRAM Devices son circuitos analógicos cuya temporización es naturalmente asincrónica.
- Los pasos que le toma a la circuitería de una DRAM para almacenar y recuperar datos en forma de carga en un capacitor a través de amplificadores de detección, le insumen una latencia considerable, la cual se expresa normalmente en nseg., y no en ciclos de clock.
- Por eso de acuerdo al diseño y al proceso de fabricación cada DRAM tiene un juego de parámetros de timing diferentes.
- La solución que encontró la industria es proveerles una interfaz sincrónica.
- Para lidiar con los diferentes juegos de temporización se incluyen uno o varios registro de modo (dependiendo del device).

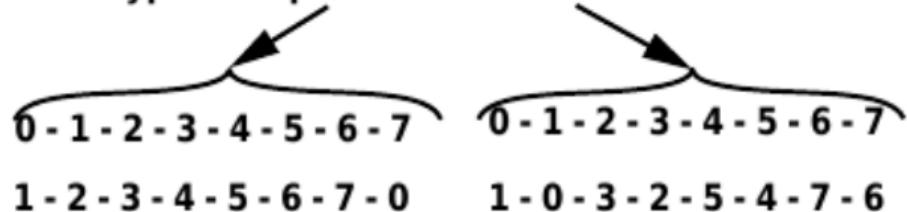
# Registro de modo de un SDRAM Device



**Burst Length = 1, 2, 4, 8, or Page mode**

**CAS Latency = 2, 3 (4, 5, etc. in special versions)**

**Burst Type = Sequential or Interleaved**



# Registro de modo de un SDRAM Device

- Posee tres campos

# Registro de modo de un SDRAM Device

- Posee tres campos
  - ➊ **CAS latency**: En las hojas de datos se lo llama **CL**. De acuerdo al valor de este campo de bits, el SDRAM Device devolverá el dato dos o tres ciclos de clock luego de activada CAS

# Registro de modo de un SDRAM Device

- Posee tres campos
  - ➊ **CAS latency**: En las hojas de datos se lo llama **CL**. De acuerdo al valor de este campo de bits, el SDRAM Device devolverá el dato dos o tres ciclos de clock luego de activada CAS
  - ➋ **Burst Type**: Determina el orden en el que el SDRAM Device devuelve los datos.

# Registro de modo de un SDRAM Device

- Posee tres campos

- ➊ **CAS latency**: En las hojas de datos se lo llama **CL**. De acuerdo al valor de este campo de bits, el SDRAM Device devolverá el dato dos o tres ciclos de clock luego de activada CAS
- ➋ **Burst Type**: Determina el orden en el que el SDRAM Device devuelve los datos.
- ➌ **Burst Length**: Determina el número de columnas de una fila entera que el SDRAM Device devuelve por cada comando CAS: 1, 2, 4, u 8 bits.

# Registro de modo de un SDRAM Device

- Posee tres campos
  - 1 **CAS latency**: En las hojas de datos se lo llama **CL**. De acuerdo al valor de este campo de bits, el SDRAM Device devolverá el dato dos o tres ciclos de clock luego de activada CAS
  - 2 **Burst Type**: Determina el orden en el que el SDRAM Device devuelve los datos.
  - 3 **Burst Length**: Determina el número de columnas de una fila entera que el SDRAM Device devuelve por cada comando CAS: 1, 2, 4, u 8 bits.
- Los DDR SDRAM Devices y los D-RDRAM Devices, tienen mas Registros de Modos en los que se pueden configurar mas parámetros de control.

# Diferentes configuraciones para el mismo tamaño

## Diferentes configuraciones para el mismo tamaño

- Las SDRAM se clasifican de acuerdo al número de bits contenidos en el Device. Pero independientemente de este dato básico se puede organizar de diferentes maneras, de acuerdo con el tamaño de la palabra de datos que se pretenda leer o escribir en el Device.

Configuración	64Mbit x 4	32Mbit x 8	16Mbit x 16
Nº Bancos	4	4	4
Nº Filas	8192	8192	8192
Nº Columnas	2048	1024	512
Ancho Data Bus	4	8	16

## Diferentes configuraciones para el mismo tamaño

- Las SDRAM se clasifican de acuerdo al número de bits contenidos en el Device. Pero independientemente de este dato básico se puede organizar de diferentes maneras, de acuerdo con el tamaño de la palabra de datos que se pretenda leer o escribir en el Device.

Configuración	64Mbit x 4	32Mbit x 8	16Mbit x 16
Nº Bancos	4	4	4
Nº Filas	8192	8192	8192
Nº Columnas	2048	1024	512
Ancho Data Bus	4	8	16

- A pesar de los cambios de configuración, el número de filas de la SDRAM del Device, se mantiene constante, mientras que el número de columnas por fila decrece a medida que se agranda la cantidad de bits que entrega el Device en un mismo CAS.

# Diferentes configuraciones para el mismo tamaño

## Diferentes configuraciones para el mismo tamaño

- Para capacidades mayores de memoria no siempre se cumple que el tamaño de fila se mantenga constante. De hecho para una DRAM de 1 Gbit normalmente se tiene una configuración como la siguiente

Configuración	256Mbit x 4	128Mbit x 8	64Mbit x 16
Nº Bancos	8	8	8
Nº Filas	16384	16384	8192
Nº Columnas	2048	1024	1024
Ancho Data Bus	4	8	16

## Diferentes configuraciones para el mismo tamaño

- Para capacidades mayores de memoria no siempre se cumple que el tamaño de fila se mantenga constante. De hecho para una DRAM de 1 Gbit normalmente se tiene una configuración como la siguiente

Configuración	256Mbit x 4	128Mbit x 8	64Mbit x 16
Nº Bancos	8	8	8
Nº Filas	16384	16384	8192
Nº Columnas	2048	1024	1024
Ancho Data Bus	4	8	16

- Estas diferencias en las configuraciones implican diferente cantidad de bits por línea, lo cual deriva en diferentes cantidades de circuitos que se activan en un comando CAS variando las características de performance y consumo para diferentes configuraciones en una misma generación de Dispositivos.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas

Multiprocesador

## ● Coherencia de un cache

## ● Protocolos de Coherencia para Multicore

## ● Casos del Mundo real

## 5 Memorias Dinámicas

### ● Introducción

### ● Organización interna

## 6 Standards

### ● Estado del arte

### ● JEDEC SDRAM

## 7 Controladores de Memoria

### ● Introducción General

### ● Arquitectura

## 8 Configuración

### ● Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

● Evitar cuellos de botella es la clave

## 10 Casos Prácticos

### ● Beagle Bone Black

### ● Memorias DDR en la BBB

### ● Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

### ● Vistazo introductorio

### ● Decodificación

### ● Implementación

## 12 DRAM Detalles de implementación

### ● Acceso a las celdas

### ● JEDEC DDR SDRAM

### ● Protocolo de acceso

# Longitud y orden de ráfagas (burst)

## Longitud y orden de ráfagas (burst)

- Hemos visto que en el Registro de Modo podemos programar al DRAM Device el tamaño del burst como 1, 2, 4, u 8 columnas de datos a transferir en respuesta a un solo comando CAS.

## Longitud y orden de ráfagas (burst)

- Hemos visto que en el Registro de Modo podemos programar al DRAM Device el tamaño del burst como 1, 2, 4, u 8 columnas de datos a transferir en respuesta a un solo comando CAS.
- En el caso de los SDRAM y DDR SDRAM es necesario re ordenar los datos de manera tal que el dato de la columna requerida esté en primer lugar en la transferencia.

## Longitud y orden de ráfagas (burst)

- Hemos visto que en el Registro de Modo podemos programar al DRAM Device el tamaño del burst como 1, 2, 4, u 8 columnas de datos a transferir en respuesta a un solo comando CAS.
- En el caso de los SDRAM y DDR SDRAM es necesario re ordenar los datos de manera tal que el dato de la columna requerida esté en primer lugar en la transferencia.
- Por ejemplo, una SDRAM x8, que recibe un comando read en la dirección de columna 17, debe proveer los 8 datos abarcados desde la columna 16 a la 23, ya que la salida es por grupos de 8.

## Longitud y orden de ráfagas (burst)

- Hemos visto que en el Registro de Modo podemos programar al DRAM Device el tamaño del burst como 1, 2, 4, u 8 columnas de datos a transferir en respuesta a un solo comando CAS.
- En el caso de los SDRAM y DDR SDRAM es necesario re ordenar los datos de manera tal que el dato de la columna requerida esté en primer lugar en la transferencia.
- Por ejemplo, una SDRAM x8, que recibe un comando read en la dirección de columna 17, debe proveer los 8 datos abarcados desde la columna 16 a la 23, ya que la salida es por grupos de 8.
- De este modo de acuerdo a como se programa el registro de modo, la salida puede ser 17-18-19-20-21-22-23-16 o 17-16-19-18-21-20-23-22.

# Prebúsqueda de n-bits

## Prebúsqueda de n-bits

- En un SDRAM device, cada vez que se envía un comando **READ**, la lógica de control determina la duración y el orden de las ráfagas (burst) de datos.

## Prebúsqueda de n-bits

- En un SDRAM device, cada vez que se envía un comando **READ**, la lógica de control determina la duración y el orden de las ráfagas (burst) de datos.
- Cada columna se mueve en forma separada desde el amplificador de detección hasta el bus de datos a través del bus de datos interno.

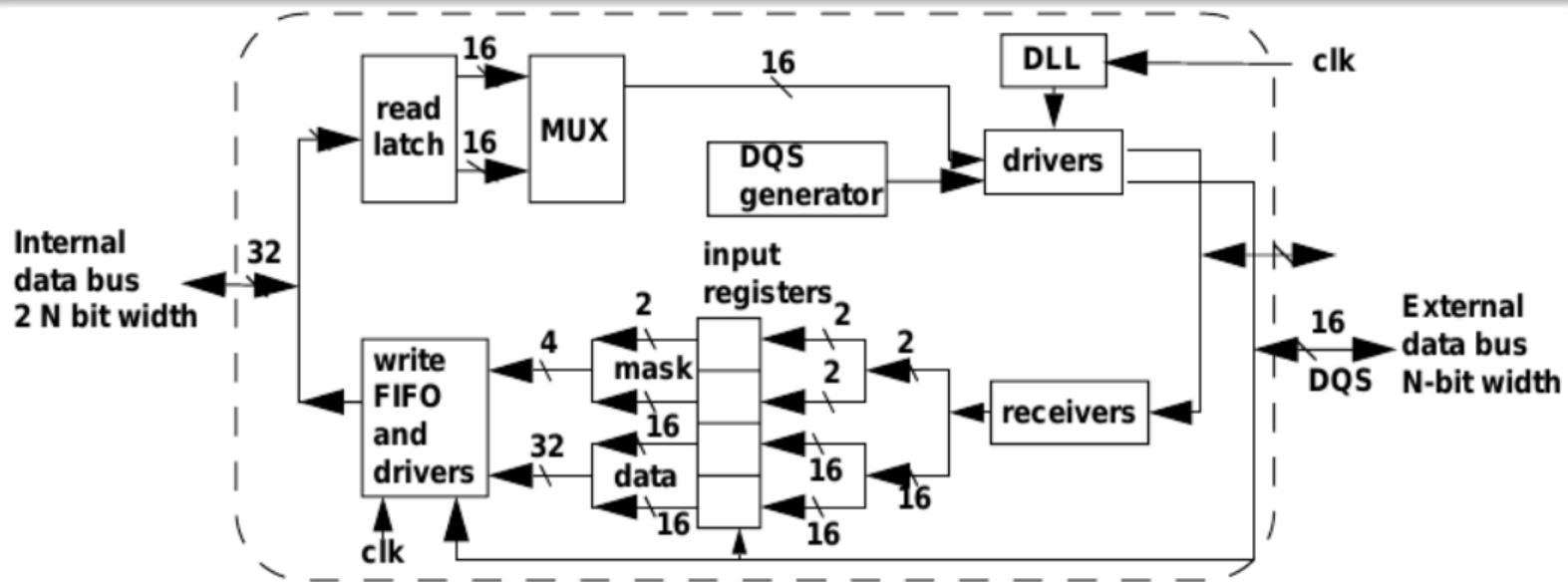
## Prebúsqueda de n-bits

- En un SDRAM device, cada vez que se envía un comando **READ**, la lógica de control determina la duración y el orden de las ráfagas (burst) de datos.
- Cada columna se mueve en forma separada desde el amplificador de detección hasta el bus de datos a través del bus de datos interno.
- La posibilidad de controlar de manera separada cada columna limita la tasa de datos de operación del DRAM device.

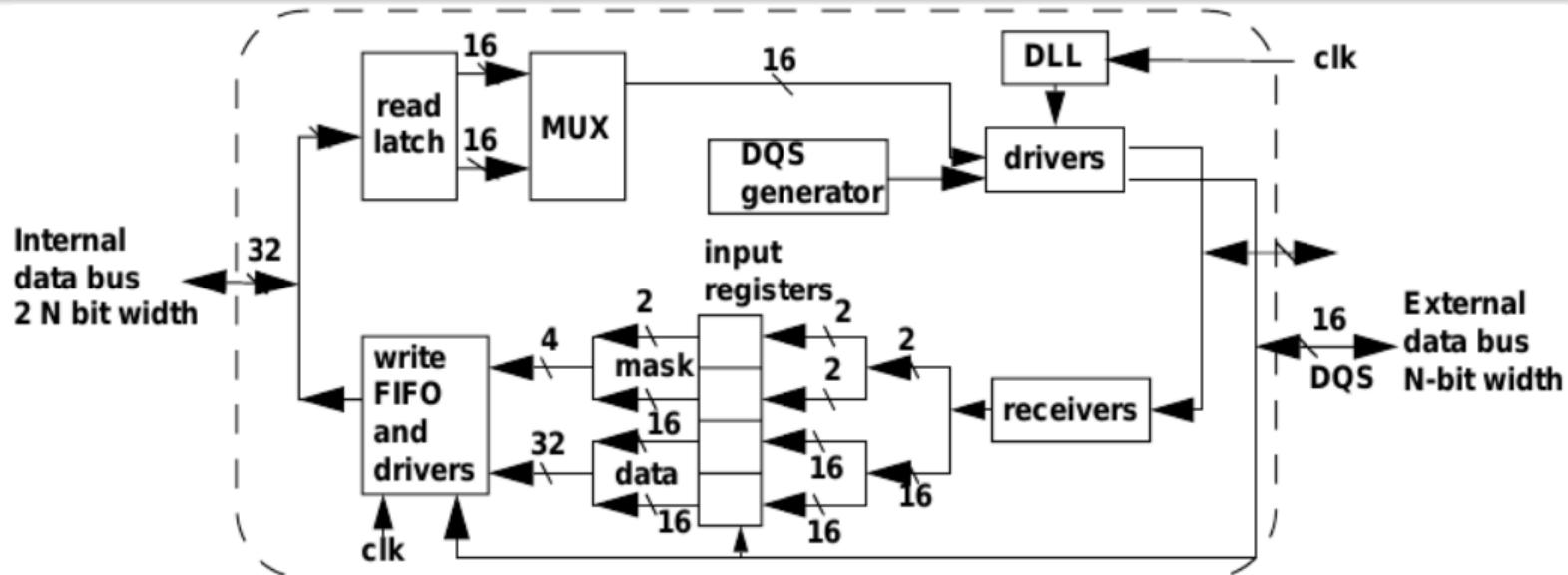
## Prebúsqueda de n-bits

- En un SDRAM device, cada vez que se envía un comando **READ**, la lógica de control determina la duración y el orden de las ráfagas (burst) de datos.
- Cada columna se mueve en forma separada desde el amplificador de detección hasta el bus de datos a través del bus de datos interno.
- La posibilidad de controlar de manera separada cada columna limita la tasa de datos de operación del DRAM device.
- En los DDRx SDRAM devices, de manera sucesiva se fueron moviendo mas cantidad de bits en paralelo desde los amplificadores de detección hasta el latch de lectura, y los datos se envían en un pipeline desde un multiplexor al bus de datos externo.

## Prebúsqueda de $2n$ -bits

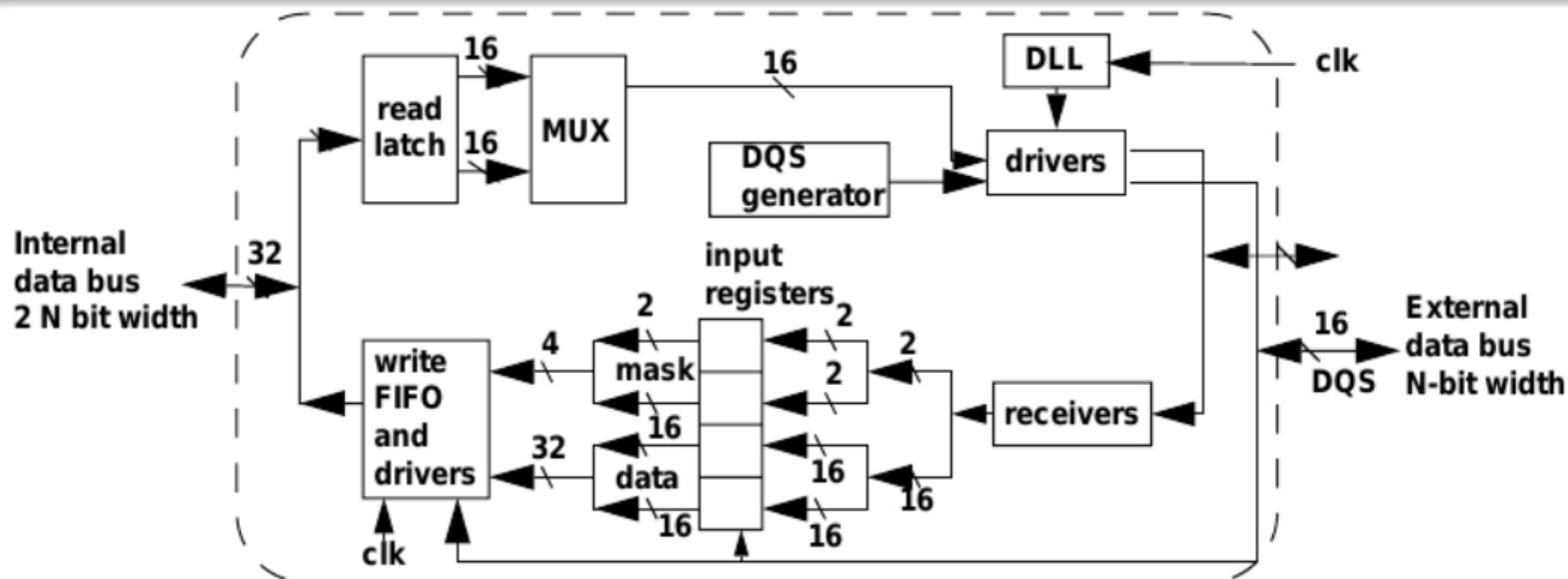


# Prebúsqueda de $2n$ -bits



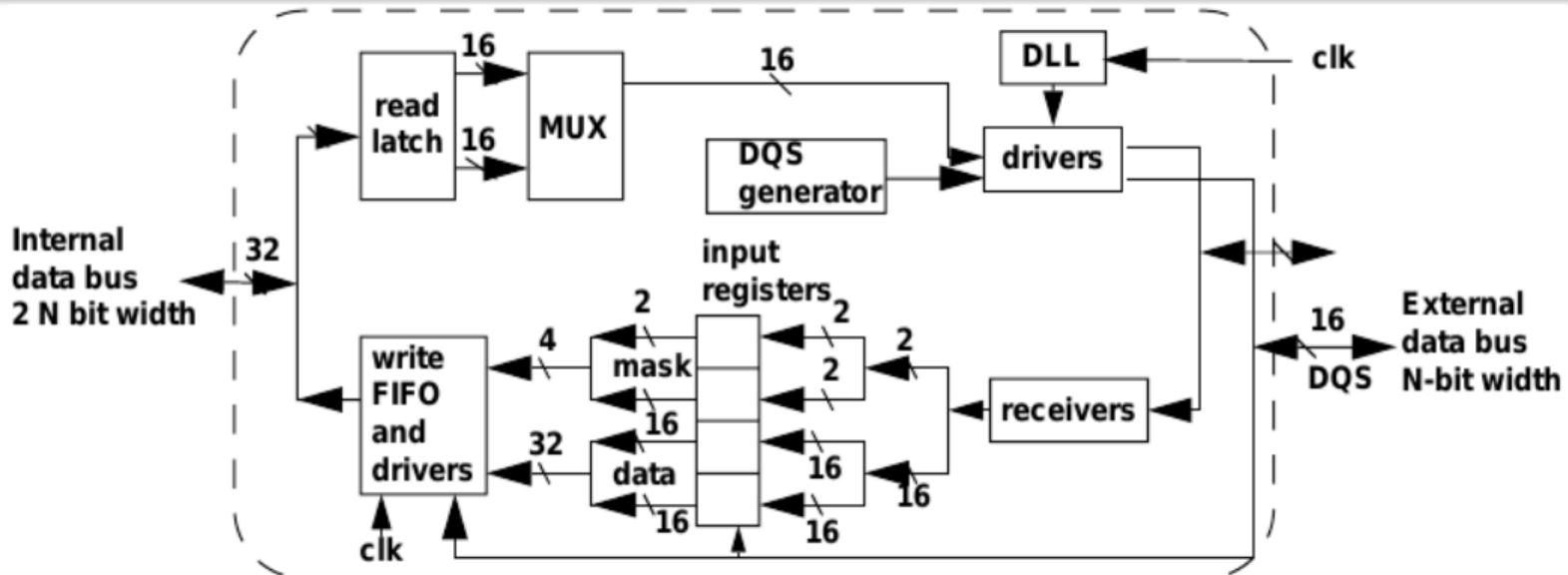
- La figura muestra como se transfiere un dato de  $N$  bits al bus externo hacia el controlador de DRAM desde un sistema que prebusca  $2N$  bits, utilizando un Multiplexor como pipeline.

# Prebúsqueda de $2n$ -bits



- La figura muestra como se transfiere un dato de  $N$  bits al bus externo hacia el controlador de DRAM desde un sistema que prebusca  $2N$  bits, utilizando un Multiplexor como pipeline.
- Una de las características de los DDR2 es que usan prebúsquedas de 4 bits

# Prebúsqueda de $2n$ -bits



- La figura muestra como se transfiere un dato de  $N$  bits al bus externo hacia el controlador de DRAM desde un sistema que prebusca  $2N$  bits, utilizando un Multiplexor como pipeline.
- Una de las características de los DDR2 es que usan prebúsquedas de 4 bits
- La velocidad se incrementa pero el chip no puede ser menor de 4x (4 bits de datos)

# Temario

1

## El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

2

## Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

3

## Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

4

## Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
  - Coherencia de un cache
  - Protocolos de Coherencia para Multicore
  - Casos del Mundo real
- 5      **Memorias Dinámicas**
- Introducción
  - Organización interna
- 6      **Standards**
- Estado del arte
  - JEDEC SDRAM
- 7      **Controladores de Memoria**
- Introducción General
  - Arquitectura
- 8      **Configuración**
- Configuración del DRAM Device

- Entrada Salida de Datos

9

## Integración con el sistema Cache

- Evitar cuellos de botella es la clave

10

## Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

11

## SRAM Cuestiones de Implementación

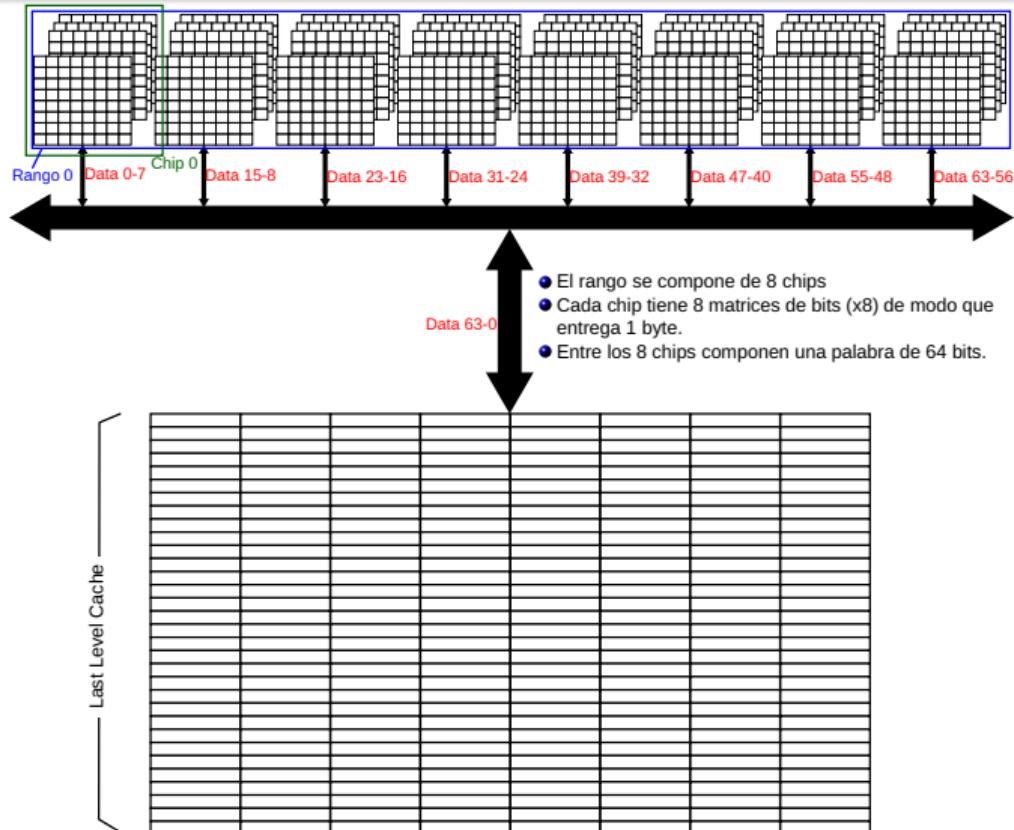
- Vistazo introductorio
- Decodificación
- Implementación

12

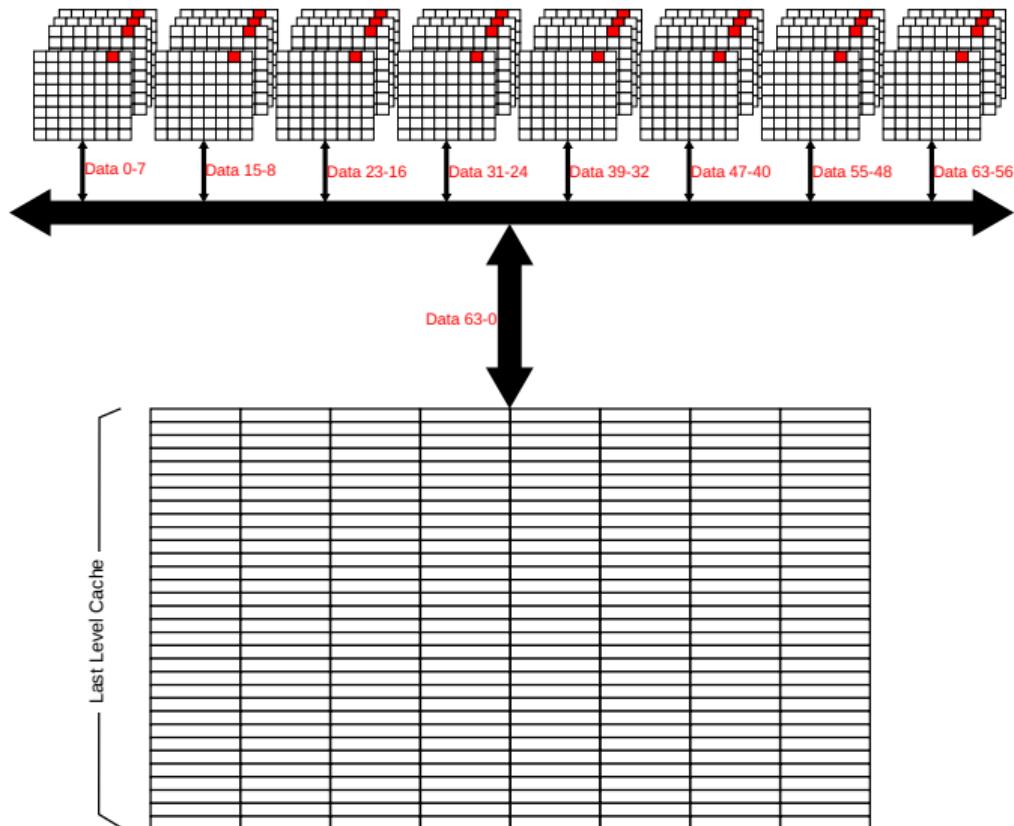
## DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

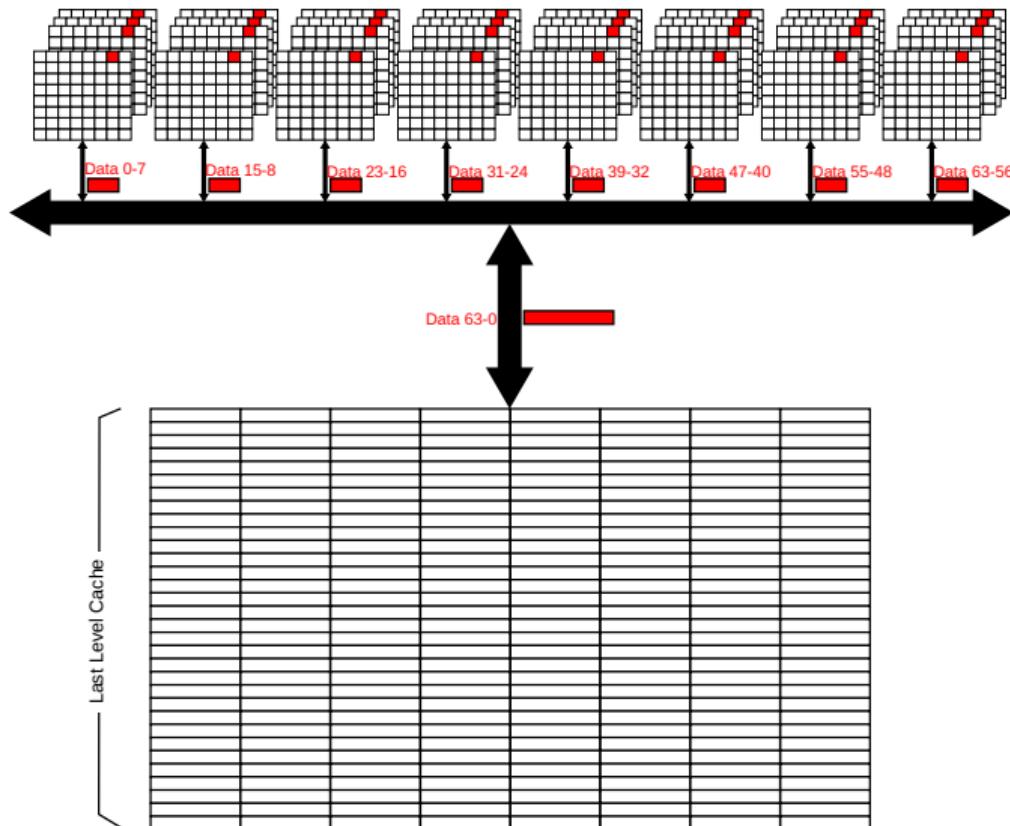
# Interfaz DRAM LLC (Last Level Cache)



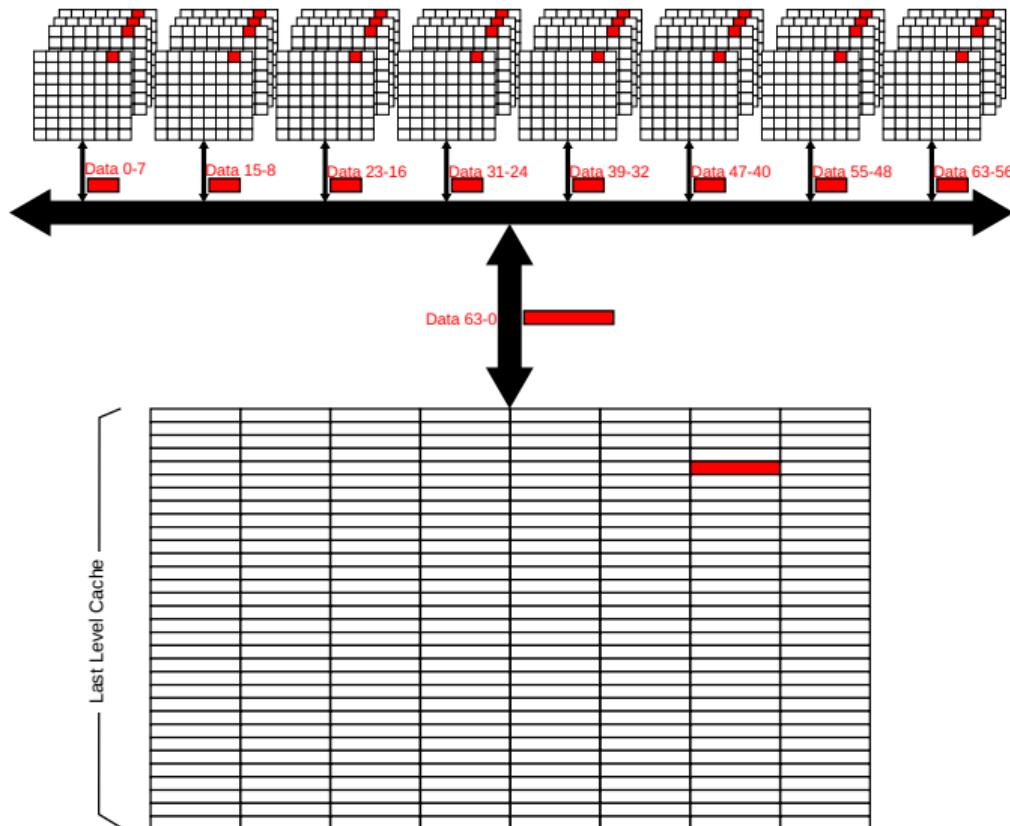
# Interfaz DRAM LLC (Last Level Cache)



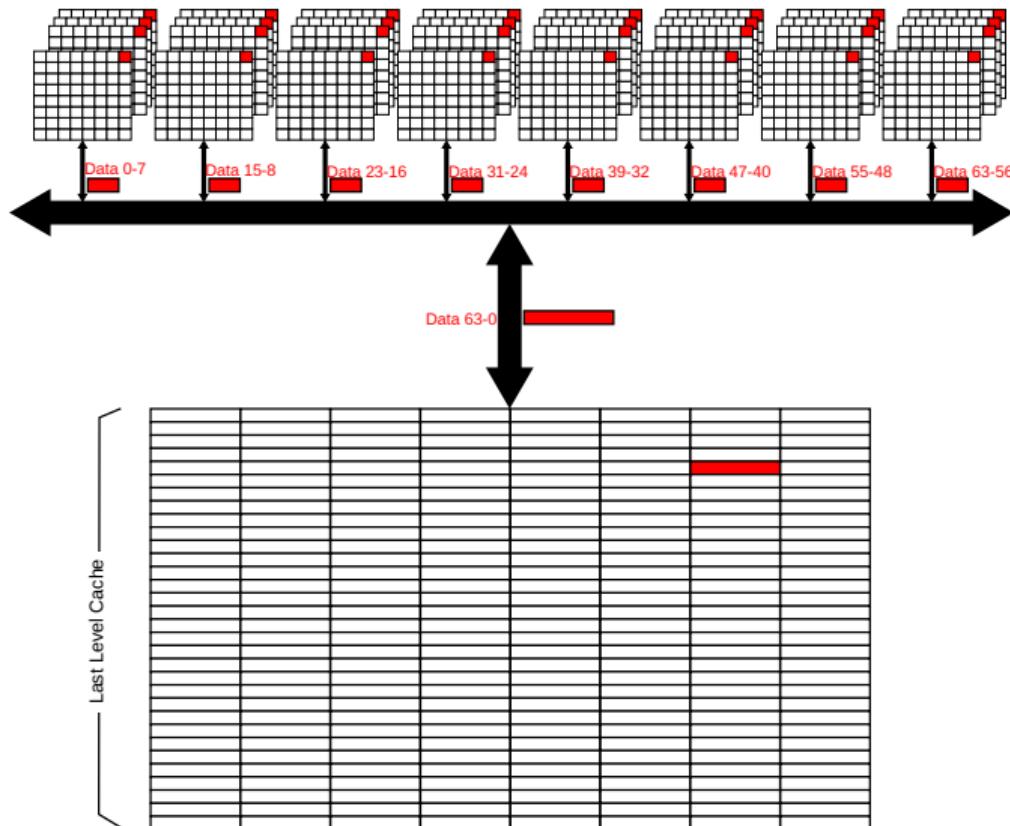
# Interfaz DRAM LLC (Last Level Cache)



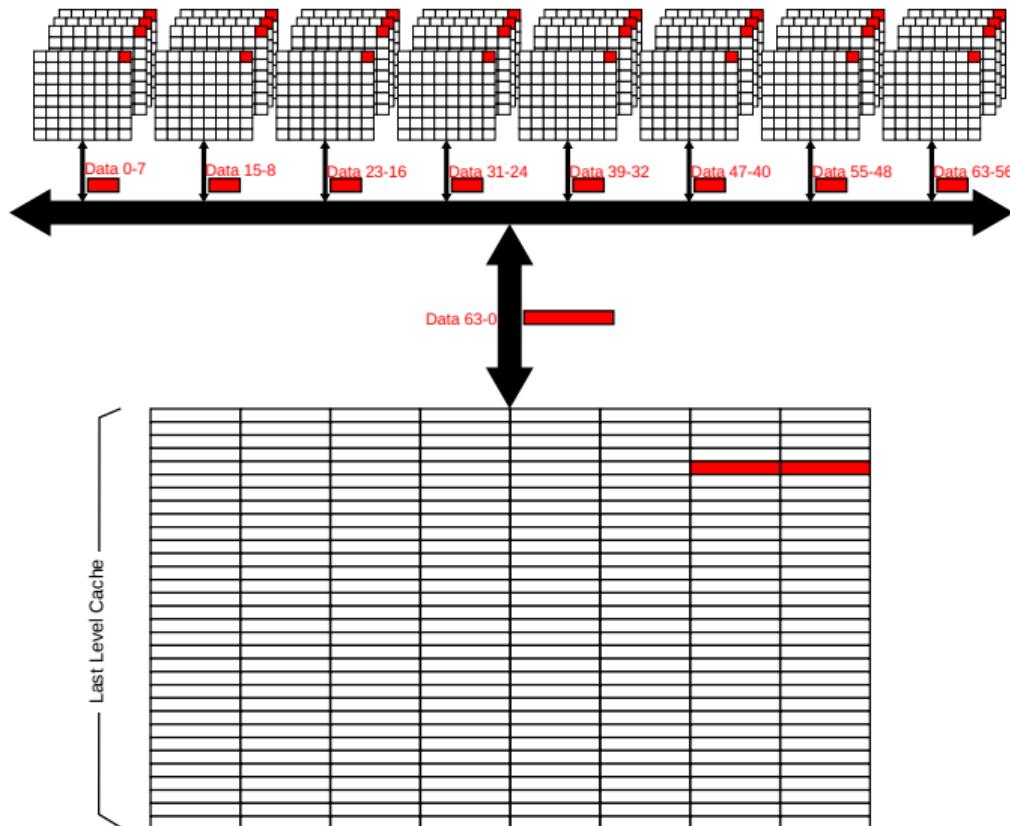
# Interfaz DRAM LLC (Last Level Cache)



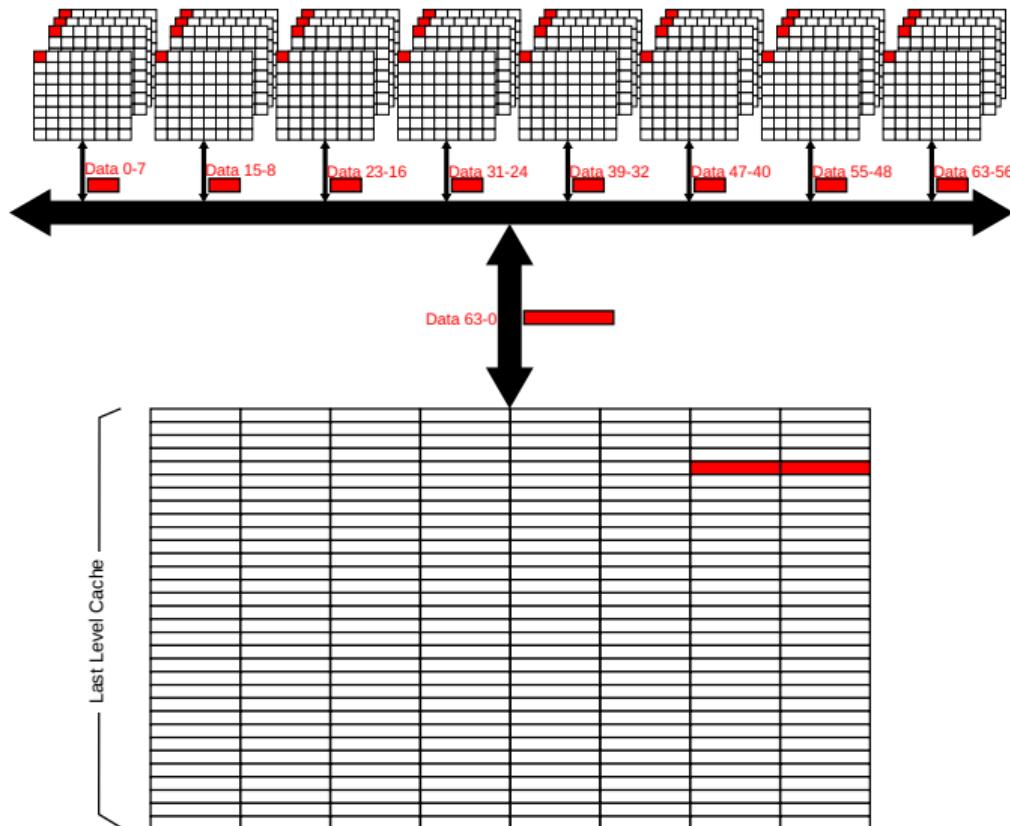
# Interfaz DRAM LLC (Last Level Cache)



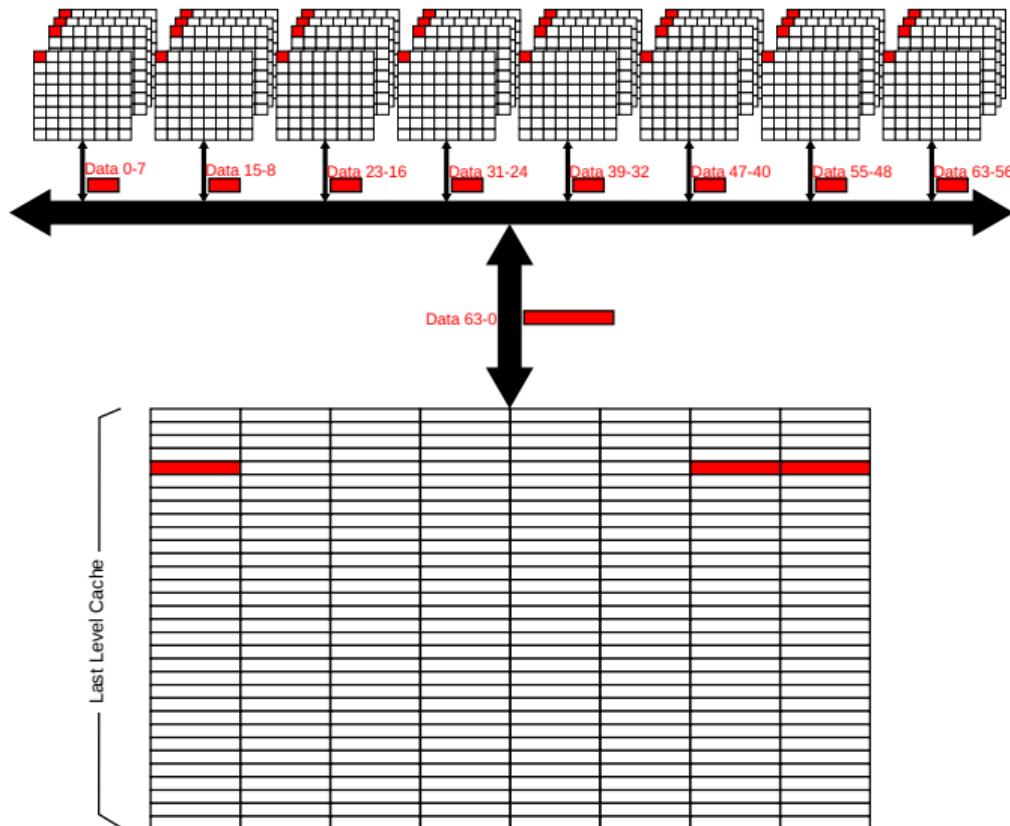
# Interfaz DRAM LLC (Last Level Cache)



# Interfaz DRAM LLC (Last Level Cache)



# Interfaz DRAM LLC (Last Level Cache)



# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

### ● Beagle Bone Black

- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

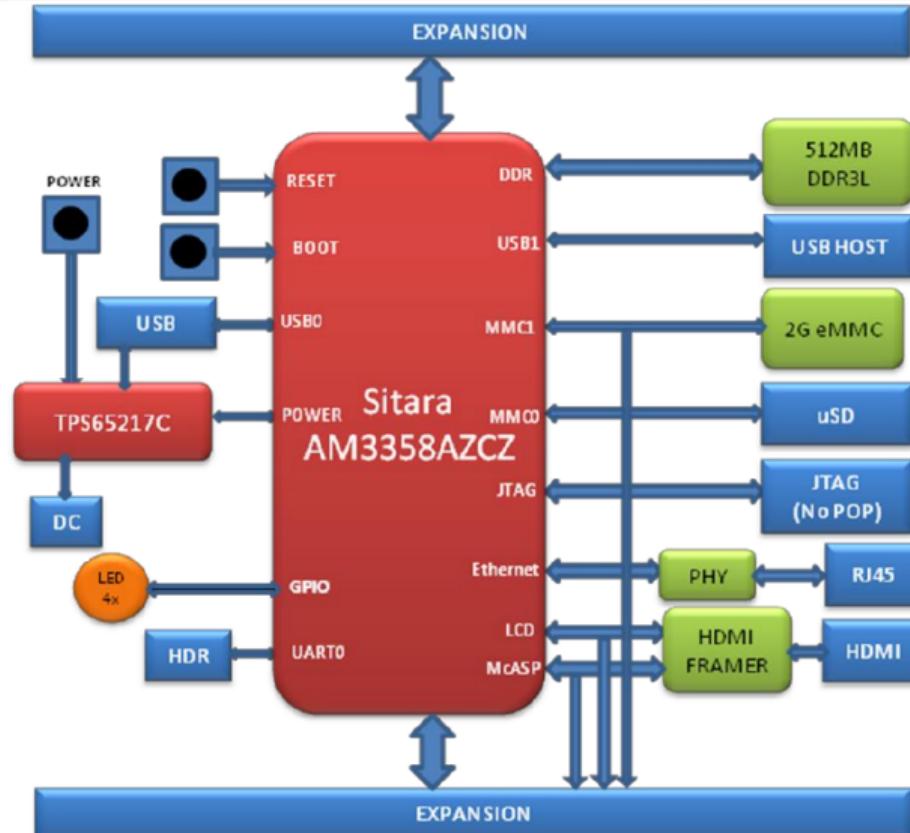
## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Diagrama General de la BBB



# Subsistema de Memoria

# Subsistema de Memoria

- **512MB DDR3L** Utiliza un solo device DRAM de 256Mb $\times$ 16 DDR3L 4Gb (512MB). El chip de memoria es el MT41K256M16HA-125 de Micron. Opera a 400MHz manteniendo una velocidad efectiva de 800MHZ sobre el bus DDR3L, alcanzando así un ancho de banda de memoria de 1.6GB/S.

## Subsistema de Memoria

- **512MB DDR3L** Utiliza un solo device DRAM de 256Mb $\times$ 16 DDR3L 4Gb (512MB). El chip de memoria es el MT41K256M16HA-125 de Micron. Opera a 400MHz manteniendo una velocidad efectiva de 800MHZ sobre el bus DDR3L, alcanzando así un ancho de banda de memoria de 1.6GB/S.
- **4KB EEPROM** Mantiene la información del board: board name, serial number, revisión, etc. .

## Subsistema de Memoria

- **512MB DDR3L** Utiliza un solo device DRAM de 256Mb $\times$ 16 DDR3L 4Gb (512MB). El chip de memoria es el MT41K256M16HA-125 de Micron. Opera a 400MHz manteniendo una velocidad efectiva de 800MHZ sobre el bus DDR3L, alcanzando así un ancho de banda de memoria de 1.6GB/S.
- **4KB EEPROM** Mantiene la información del board: board name, serial number, revisión, etc. .
- **2GB Embedded MMC** Se conecta por default al port MMC1 del procesador, permitiendo accesos de 8 bits de ancho. Se puede cambiar al port MMC0, al slot de la SD.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- **Memorias DDR en la BBB**
- Controlador de DDRn SDRAM en la BBB

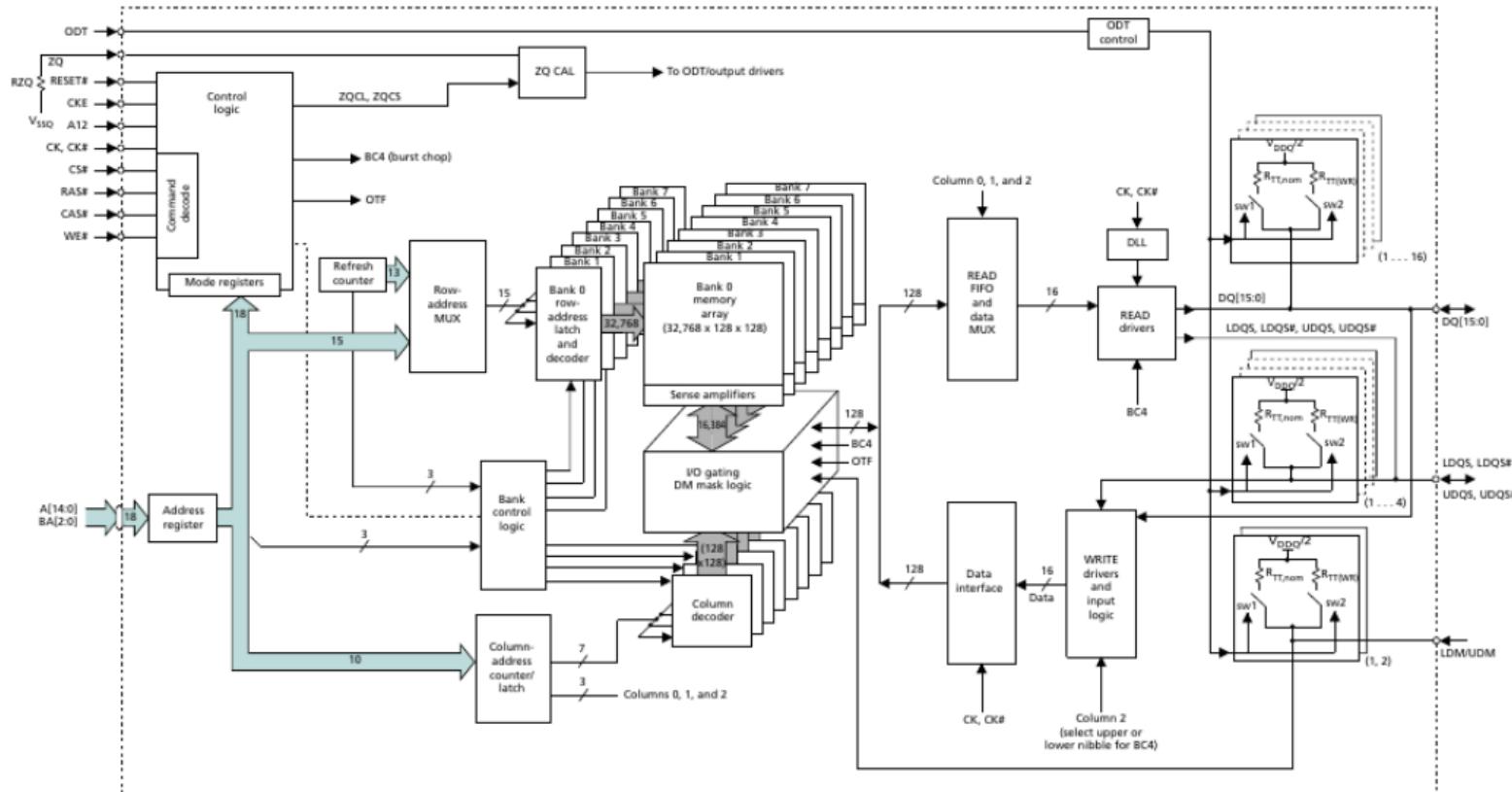
## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

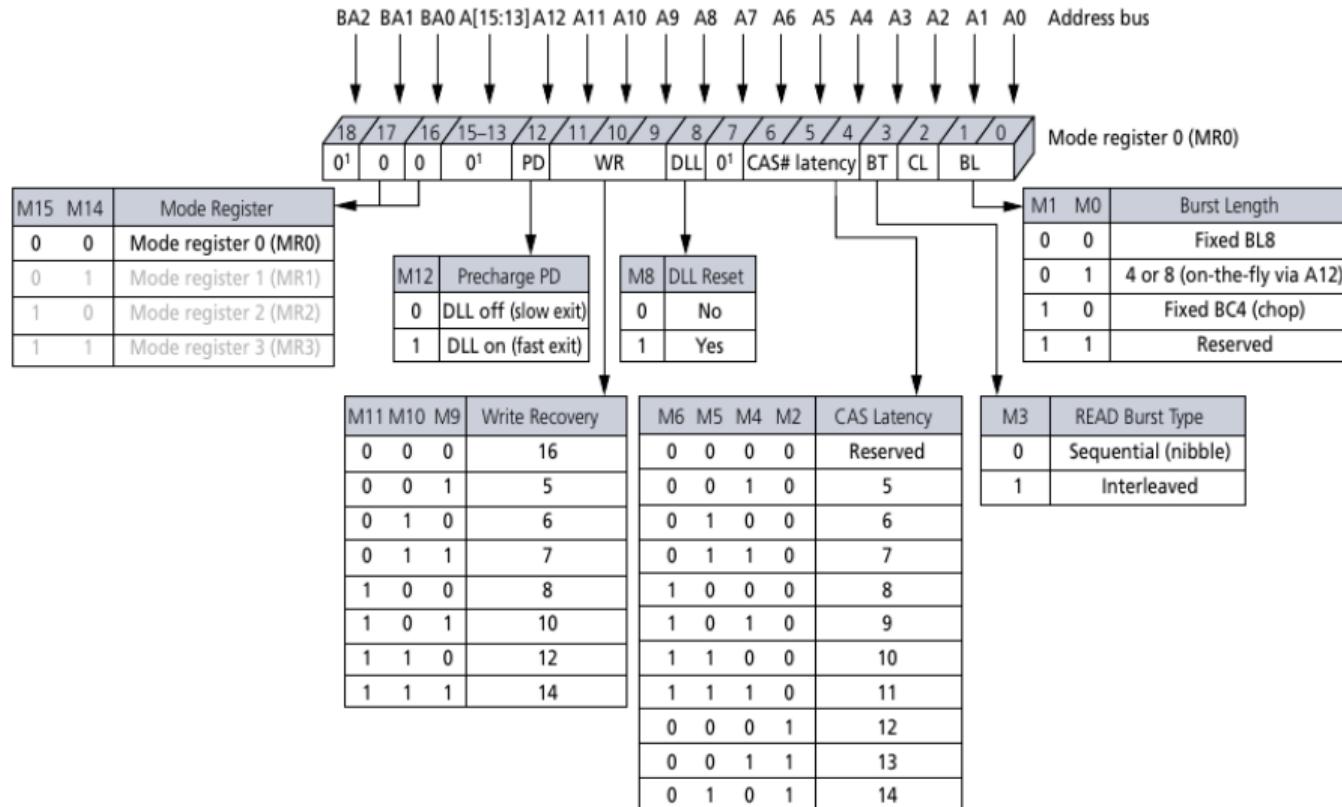
## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

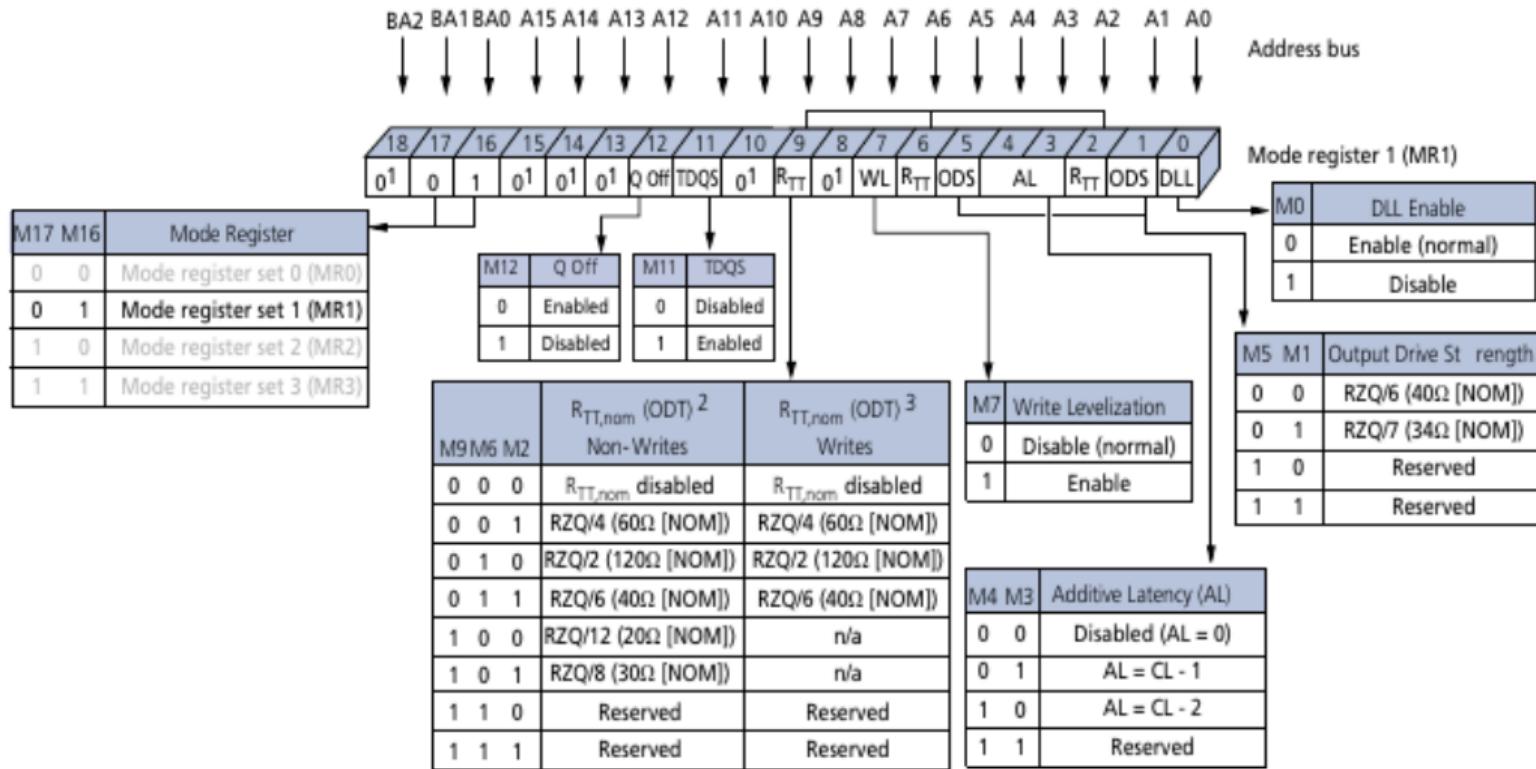
# Organización de la DRAM MT41K256M16 de Micron



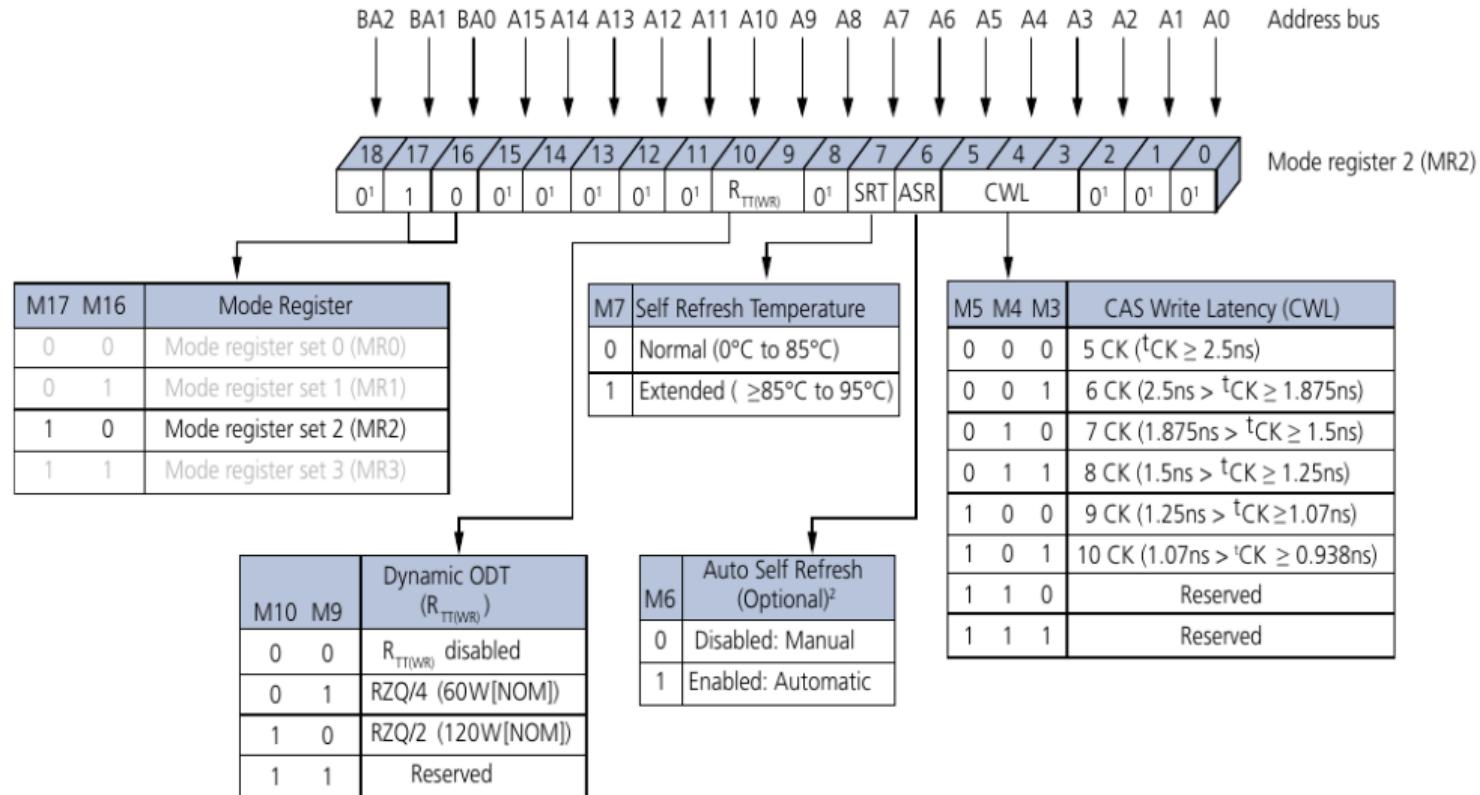
# Registro de Modo 0 del MT41K256M16



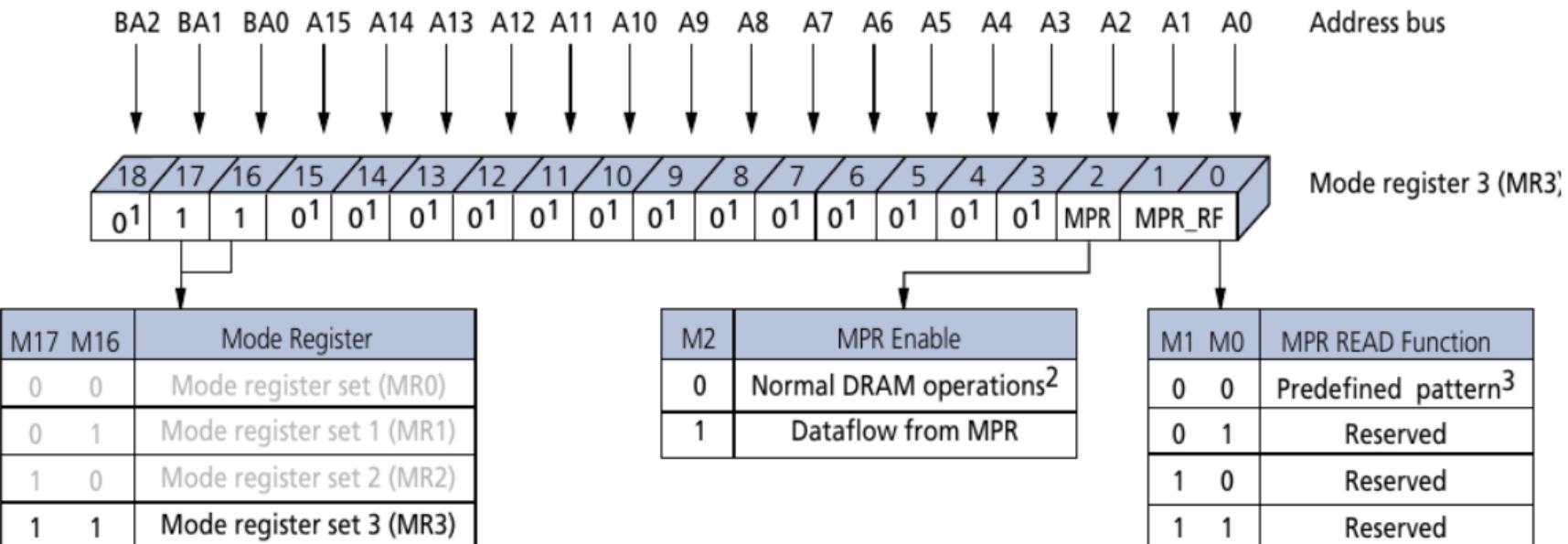
# Registro de Modo 1 del MT41K256M16



# Registro de Modo 2 del MT41K256M16



# Registro de Modo 3 del MT41K256M16



# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Registros Memory Mapped

# Registros Memory Mapped

- Los Cores ARM tienen E/S memory mapped.

# Registros Memory Mapped

- Los Cores ARM tienen E/S memory mapped.
- El controlador de memoria (EMIF) tiene un set de registros de 32 bits mapeados a partir de la dirección 0x4C000000)

# Registros Memory Mapped

- Los Cores ARM tienen E/S memory mapped.
- El controlador de memoria (EMIF) tiene un set de registros de 32 bits mapeados a partir de la dirección 0x4C000000
- En rigor el espacio asignado al controlador es 0x4C000000 a 0x4CFFFFFF.

# Cálculo de los valores a programar

	A	B	C	D	E	F	G	H
1	AM335x DDR3 Timing Configuration Tool							
2								
3		AM335x register bit length	Memory datasheet symbol	Memory Datasheet value	unit	AM335x Setting (Decimal)	Comments	
4		tCK	tCK	3.3 ns				
5	SDRAM_TIM_1	REG_T_RP	4 tRP	13.5 ns		4	typically taken from the speed bin tables	
6		REG_T_RCD	4 tRCD	13.5 ns		4	typically taken from the speed bin tables	
7		REG_T_WR	4 tWR	15 ns		4		
8		REG_T_RAS	5 tRAS	36 ns		10	tRAS should be $\geq$ tRCD	
9		REG_T_RC	6 tRC	49.5 ns		14		
10		REG_T_RRD	3 tRRD	4 tCK		3	use the value given in CK units	
11		REG_T_WTR	3 tWTR	4 tCK		3	use the value given in CK units	
12	SDRAM_TIM_2	REG_T_XP	3 tXP	3 tCK		2	use the value given in CK units	
13		REG_T_ODT	3 tODTlon	3 tCK		3	typically in terms of CWL. First determine CWL	
14		REG_T_XSNR	9 tXS	170 ns		51	usually tRFC+10	
15		REG_T_XSRD	10 tXSDL	512 tCK		511	usually in terms of tDLLK	
16		REG_T RTP	3 tRTP	4 tCK		3	use the value given in CK units	
17		REG_T_CKE	3 tCKE	3 tCK		2	use the value given in CK units	
18		REG_T_PDLL_UL	4			5	set to fixed value of 5	
19	SDRAM_TIM_3	REG_T_ZOCS	6 tZOCs	64 tCK		63		
20		REG_T RFC	9 tRFC	160 ns		48		
21			tREFI					
22		REG_T_RAS_MAX	4 tRASmax			15	for DDR3, must be set to 15	
23								
24								

# Cálculo de los valores a programar

	A	B	C	D	E	F	G	H
1	<b>AM335x DDR3 Timing Configuration Tool</b>							
2								
3		AM335x register name	AM335x register bit length	Memory datasheet symbol	Memory DataSheet value	unit	AM335x Setting (Decimal)	Comments
4		tCK		tCK	3.3 ns			
5	SDRAM_TIM_1	REG_T_RP	4	tRP	13.5 ns		4	typically taken from the speed bin tables
6		REG_T_RCD	4	tRCD	13.5 ns		4	typically taken from the speed bin tables
7		REG_T_WR	4	tWR	15 ns		4	
8		REG_T_RAS	5	tRAS	36 ns		10	tRAS should be $\geq tRCD$
9		REG_T_RC	6	tRC	49.5 ns		14	
10		REG_T_RRD	3	tRRD	4 tCK		3	use the value given in CK units
11		REG_T_WTR	3	tWTR	4 tCK		3	use the value given in CK units
12	SDRAM_TIM_2	REG_T_XP	3	tXP	3 tCK		2	use the value given in CK units
13		REG_T_ODT	3	tODTLon	3 tCK		3	typically in terms of CWL. First determine CWL
14		REG_T_XSNR	9	tXS	170 ns		51	usually tRFC+10
15		REG_T_XSRD	10	tXSDL	512 tCK		511	usually in terms of tDLLK
16		REG_T RTP	3	tRTP	4 tCK		3	use the value given in CK units
17		REG_T_CKE	3	tCKE	3 tCK		2	use the value given in CK units
18		REG_T_PDLL_UL	4				5	set to fixed value of 5
19	SDRAM_TIM_3	REG_T_ZOCS	6	tZOCs	64 tCK		63	
20		REG_T RFC	9	tRFC	160 ns		48	
21				tREFI				
22		REG_T_RAS_MAX	4	tRASmax			15	for DDR3, must be set to 15
23								

## Herramienta de configuración

Generalmente los fabricantes proporcionan una herramienta para calcular los valores a setear en los campos de bits de los registros del controlador de memoria. En este caso, en la columna coloreada en amarillo se ingresan los valores de temporización que figuran en la hoja de datos de la memoria que se decida colocar, y en la Columna “**AM335x Settings**” se obtiene el valor a programar para esa memoria.

# Tiempos

I <sub>DD</sub> Parameter		DDR3L -800		DDR3L -1066		DDR3L -1333		DDR3L -1600		DDR3L -1866	DDR3L -2133	Unit
		-25E	-25	-187E	-187	-15E	-15	-125E	-125	-107	-093	
		5-5-5	6-6-6	7-7-7	8-8-8	9-9-9	10-10-10	10-10-10	11-11-11	13-13-13	14-14-14	
t <sub>CK</sub> (MIN) I <sub>DD</sub>		2.5		1.875		1.5		1.25		1.07	0.938	ns
CL I <sub>DD</sub>		5	6	7	8	9	10	10	11	13	14	CK
t <sub>RCD</sub> (MIN) I <sub>DD</sub>		5	6	7	8	9	10	10	11	13	14	CK
t <sub>RC</sub> (MIN) I <sub>DD</sub>		20	21	27	28	33	34	38	39	45	50	CK
t <sub>RAS</sub> (MIN) I <sub>DD</sub>		15	15	20	20	24	24	28	28	32	36	CK
t <sub>RP</sub> (MIN)		5	6	7	8	9	10	10	11	13	14	CK
t <sub>FAW</sub>	x4, x8	16	16	20	20	20	20	24	24	26	27	CK
	x16	20	20	27	27	30	30	32	32	33	38	CK
t <sub>RRD</sub> I <sub>DD</sub>	x4, x8	4	4	4	4	4	4	5	5	5	6	CK
	x16	4	4	6	6	5	5	6	6	6	7	CK
t <sub>RFC</sub>	1Gb	44	44	59	59	74	74	88	88	103	118	CK
	2Gb	64	64	86	86	107	107	128	128	150	172	CK
	4Gb	104	104	139	139	174	174	208	208	243	279	CK
	8Gb	140	140	187	187	234	234	280	280	328	375	CK

# Tiempos

## Electrical Characteristics and AC Operating Conditions

Parameter	Symbol	DDR3L-800		DDR3L-1066		DDR3L-1333		DDR3L-1600		Unit	Notes
		Min	Max	Min	Max	Min	Max	Min	Max		
DQS, DQS# Low-Z time (RL - 1)	$t_{LZDQS}$	-800	400	-600	300	-500	250	-450	225	ps	22, 23
DQS, DQS# High-Z time (RL + BL/2)	$t_{HZDQS}$	-	400	-	300	-	250	-	225	ps	22, 23
DQS, DQS# differential READ preamble	$t_{RPRE}$	0.9	Note 24	0.9	Note 24	0.9	Note 24	0.9	Note 24	CK	23, 24
DQS, DQS# differential READ postamble	$t_{RPST}$	0.3	Note 27	0.3	Note 27	0.3	Note 27	0.3	Note 27	CK	23, 27
<b>Command and Address Timing</b>											
DLL locking time	$t_{DLLK}$	512	-	512	-	512	-	512	-	CK	28
CTRL, CMD, ADDR setup to CK,CK#	$t_{IS}$ (AC160)	215	-	140	-	80	-	60	-	ps	29, 30, 44
	$V_{REF} @ 1 V/ns$	375	-	300	-	240	-	220	-	ps	20, 30
CTRL, CMD, ADDR setup to CK,CK#	$t_{IS}$ (AC135)	365	-	290	-	205	-	185	-	ps	29, 30, 44
	$V_{REF} @ 1 V/ns$	500	-	425	-	340	-	320	-	ps	20, 30
CTRL, CMD, ADDR setup to CK,CK#	$t_{IH}$ (DC90)	285	-	210	-	150	-	130	-	ps	29, 30, 44
	$V_{REF} @ 1 V/ns$	375	-	300	-	240	-	220	-	ps	20, 30
Minimum CTRL, CMD, ADDR pulse width	$t_{IPW}$	900	-	780	-	620	-	560	-	ps	41
ACTIVATE to internal READ or WRITE delay	$t_{RCD}$	See Speed Bin Tables for $t_{RCD}$								ns	31
PRECHARGE command period	$t_{RP}$	See Speed Bin Tables for $t_{RP}$								ns	31
ACTIVATE-to-PRECHARGE command period	$t_{RAS}$	See Speed Bin Tables for $t_{RAS}$								ns	31, 32
ACTIVATE-to-ACTIVATE command period	$t_{RC}$	See Speed Bin Tables for $t_{RC}$								ns	31, 43
ACTIVATE-to-ACTIVATE minimum command period	$t_{RRD}$	MIN = greater of 4CK or 10ns		MIN = greater of 4CK or 7.5ns		MIN = greater of 4CK or 6ns		MIN = greater of 4CK or 6ns		CK	31
x16 (2KB page size)		MIN = greater of 4CK or 10ns								CK	31
Four ACTIVATE windows	$t_{FAW}$	40	-	37.5	-	30	-	30	-	ns	31
		50	-	50	-	45	-	40	-	ns	31
Write recovery time	$t_{WR}$	MIN = 15ns; MAX = N/A								ns	31, 32, 33,34
Delay from start of internal WRITE transaction to internal READ command	$t_{WTR}$	MIN = greater of 4CK or 7.5ns; MAX = N/A								CK	31, 34
READ-to-PRECHARGE time	$t_{RTP}$	MIN = greater of 4CK or 7.5ns; MAX = N/A								CK	31, 32

# Tiempos

## Electrical Characteristics and AC Operating Conditions

Parameter	Symbol	DDR3L-800		DDR3L-1066		DDR3L-1333		DDR3L-1600		Unit	Notes
		Min	Max	Min	Max	Min	Max	Min	Max		
WWRITE with auto precharge command to power-down entry	BL8 (OTF, MRS) BC4OTF	'WRAP-DEN				MIN = WL + 4 + WR + 1				CK	
	BC4MRS	'WRAP-DEN				MIN = WL + 2 + WR + 1				CK	
<b>Power-Down Exit Timing</b>											
DLL on, any valid command, or DLL off to commands not requiring locked DLL	'XP	MIN = greater of 3CK or 7.5ns; MAX = N/A			MIN = greater of 3CK or 6ns; MAX = N/A			CK			
Precharge power-down with DLL off to commands requiring a locked DLL	'XPDLL	MIN = greater of 10CK or 24ns; MAX = N/A				CK				28	
<b>ODT Timing</b>											
R <sub>TT</sub> synchronous turn-on delay	ODTLon	CWL + AL - 2CK				CK				38	
R <sub>TT</sub> synchronous turn-off delay	ODTLooff	CWL + AL - 2CK				CK				40	
R <sub>TT</sub> turn-on from ODTL on reference	'AON	-400	400	-300	300	-250	250	-225	225	ps	23, 38
R <sub>TT</sub> turn-off from ODTL off reference	'AOF	0.3	0.7	0.3	0.7	0.3	0.7	0.3	0.7	CK	39, 40
Asynchronous R <sub>TT</sub> turn-on delay (power-down with DLL off)	'AONPD	MIN = 2; MAX = 8.5				ns				38	
Asynchronous R <sub>TT</sub> turn-off delay (power-down with DLL off)	'AOFPD	MIN = 2; MAX = 8.5				ns				40	
ODT HIGH time with WWRITE command and BL8	ODTH8	MIN = 6; MAX = N/A				CK					
ODT HIGH time without WRITE command or with WRITE command and BC4	ODTH4	MIN = 4; MAX = N/A				CK					
<b>Dynamic ODT Timing</b>											
R <sub>TT,nom</sub> -to-R <sub>TT,WRI</sub> change skew	ODTLCnw	WL - 2CK				CK					
R <sub>TT,WRI</sub> -to-R <sub>TT,nom</sub> change skew - BC4	ODTLCnw4	4CK + ODTLoff				CK					
R <sub>TT,WRI</sub> -to-R <sub>TT,nom</sub> change skew - BL8	ODTLCnw8	6CK + ODTLoff				CK					
R <sub>TT</sub> dynamic change skew	'ADC	0.3	0.7	0.3	0.7	0.3	0.7	0.3	0.7	CK	39
<b>Write Leveling Timing</b>											
First DQS, DQS# rising edge	'WLMRD	40	-	40	-	40	-	40	-	CK	
DQS, DQS# delay	'WLQSEN	25	-	25	-	25	-	25	-	CK	
Write leveling setup from rising CK, CK# crossing to rising DQS, DQS# crossing	'WLS	325	-	245	-	195	-	165	-	ps	

# Tiempos

## Electrical Characteristics and AC Operating Conditions

Parameter	Symbol	DDR3L-800		DDR3L-1066		DDR3L-1333		DDR3L-1600		Unit	Notes
		Min	Max	Min	Max	Min	Max	Min	Max		
CAS#-to-CAS# command delay	<sup>1</sup> CCD			MIN = 4CK; MAX = N/A						CK	
Auto precharge write recovery + precharge time	<sup>1</sup> DAL			MIN = WR + <sup>1</sup> RP/ <sup>1</sup> CK (AVG); MAX = N/A						CK	
MODE REGISTER SET command cycle time	<sup>1</sup> MRD			MIN = 4CK; MAX = N/A						CK	
MODE REGISTER SET command update delay	<sup>1</sup> MOD			MIN = greater of 12CK or 15ns; MAX = N/A						CK	
MULTIPURPOSE REGISTER READ burst end to mode register set for multipurpose register exit	<sup>1</sup> MPRR			MIN = 1CK; MAX = N/A						CK	
<b>Calibration Timing</b>											
ZQCL command: Long calibration time	POWER-UP and RESET operation	<sup>1</sup> ZQinit	512	–	512	–	512	–	512	–	CK
Normal operation	<sup>1</sup> ZQoper	256	–	256	–	256	–	256	–	CK	
ZQCS command: Short calibration time	<sup>1</sup> ZQCS	64	–	64	–	64	–	64	–	CK	
<b>Initialization and Reset Timing</b>											
Exit reset from CKE HIGH to a valid command	<sup>1</sup> XPR			MIN = greater of 5CK or <sup>1</sup> RFC + 10ns; MAX = N/A						CK	
Begin power supply ramp to power supplies stable	<sup>1</sup> VDDPR			MIN = N/A; MAX = 200						ms	
RESET# LOW to power supplies stable	<sup>1</sup> RPS			MIN = 0; MAX = 200						ms	
RESET# LOW to I/O and R <sub>TT</sub> High-Z	<sup>1</sup> IOZ			MIN = N/A; MAX = 20						ns	35
<b>Refresh Timing</b>											
REFRESH-to-ACTIVATE or REFRESH command period	<sup>1</sup> RFC – 1Gb			MIN = 110; MAX = 70,200						ns	
	<sup>1</sup> RFC – 2Gb			MIN = 160; MAX = 70,200						ns	
	<sup>1</sup> RFC – 4Gb			MIN = 260; MAX = 70,200						ns	
	<sup>1</sup> RFC – 8Gb			MIN = 350; MAX = 70,200						ns	
Maximum refresh period	$T_C \leq 85^\circ\text{C}$	–			64 (1X)						ms
	$T_C > 85^\circ\text{C}$				32 (2X)						ms
Maximum average periodic refresh	$T_C \leq 85^\circ\text{C}$	<sup>1</sup> REFI			7.8 (64ms/8192)						μs
	$T_C > 85^\circ\text{C}$				3.9 (32ms/8192)						μs
<b>Self Refresh Timing</b>											
Exit self refresh to commands not requiring locked DLL	<sup>1</sup> XS			MIN = greater of 5CK or <sup>1</sup> RFC + 10ns; MAX = N/A						CK	

## Tiempos

## Electrical Characteristics and AC Operating Conditions

Parameter	Symbol	DDR3L-800		DDR3L-1066		DDR3L-1333		DDR3L-1600		Unit	Notes						
		Min	Max	Min	Max	Min	Max	Min	Max								
Exit self refresh to commands requiring a locked DLL	<sup>t</sup> XSDLL	MIN = <sup>t</sup> DLL (MIN); MAX = N/A								CK	28						
Minimum CKE low pulse width for self refresh entry to self refresh exit timing	<sup>t</sup> CESR	MIN = <sup>t</sup> CKE (MIN) + CK; MAX = N/A								CK							
Valid clocks after self refresh entry or power-down entry	<sup>t</sup> CSSRE	MIN = greater of 5CK or 10ns; MAX = N/A								CK							
Valid clocks before self refresh exit, power-down exit, or reset exit	<sup>t</sup> CSSRX	MIN = greater of 5CK or 10ns; MAX = N/A								CK							
<b>Power-Down Timing</b>																	
CKE MIN pulse width	<sup>t</sup> CKE (MIN)	Greater of 3CK or 7.5ns		Greater of 3CK or 5.625ns		Greater of 3CK or 5.625ns		Greater of 3CK or 5ns		CK							
Command pass disable delay	<sup>t</sup> CPDED	MIN = 1; MAX = N/A								CK							
Power-down entry to power-down exit timing	<sup>t</sup> PD	MIN = <sup>t</sup> CKE (MIN); MAX = 9 * tREFI								CK							
Begin power-down period prior to CKE registered HIGH	<sup>t</sup> ANPD	WL - 1CK								CK							
Power-down entry period: ODT either synchronous or asynchronous	PDE	Greater of <sup>t</sup> ANPD or <sup>t</sup> RFC - REFRESH command to CKE LOW time								CK							
Power-down exit period: ODT either synchronous or asynchronous	PDX	<sup>t</sup> ANPD + <sup>t</sup> XPDLL								CK							
<b>Power-Down Entry Minimum Timing</b>																	
ACTIVATE command to power-down entry	<sup>t</sup> ACTPDEN	MIN = 1								CK							
PRECHARGE/PRECHARGE ALL command to power-down entry	<sup>t</sup> PRPDEN	MIN = 1								CK							
REFRESH command to power-down entry	<sup>t</sup> REFPDEN	MIN = 1								CK	37						
MRS command to power-down entry	<sup>t</sup> MRSPDEN	MIN = <sup>t</sup> MOD (MIN)								CK							
READ/READ with auto precharge command to power-down entry	<sup>t</sup> RDPDEN	MIN = RL + 4 + 1								CK							
WRITE command to power-down entry	BLB (OTF, MRS) BC4OTF	<sup>t</sup> WRPDEN	MIN = WL + 4 + <sup>t</sup> WR/CK (AVG)								CK						
	BC4MRS	<sup>t</sup> WRPDEN	MIN = WL + 2 + <sup>t</sup> WR/CK (AVG)								CK						

# Tiempos

## Electrical Characteristics and AC Operating Conditions

DDR3-1066 Speed Bin		-187E		-187		Unit	Notes		
CL-'RCD-'RP		7-7-7		8-8-8					
Parameter	Symbol	Min	Max	Min	Max				
Internal READ command to first data	<sup>t</sup> AA	13.125	—	15	—	ns			
ACTIVATE to internal READ or WRITE delay time	<sup>t</sup> RCD	13.125	—	15	—	ns			
PRECHARGE command period	<sup>t</sup> RP	13.125	—	15	—	ns			
ACTIVATE-to-ACTIVATE or REFRESH command period	<sup>t</sup> RC	50.625	—	52.5	—	ns			
ACTIVATE-to-PRECHARGE command period	<sup>t</sup> RAS	37.5	9 x <sup>t</sup> REFI	37.5	9 x <sup>t</sup> REFI	ns	1		
CL = 5	CWL = 5	<sup>t</sup> CK (AVG)	3.0	3.3	3.0	3.3	ns 2		
	CWL = 6	<sup>t</sup> CK (AVG)	Reserved		Reserved		ns 3		
CL = 6	CWL = 5	<sup>t</sup> CK (AVG)	2.5	3.3	2.5	3.3	ns 2		
	CWL = 6	<sup>t</sup> CK (AVG)	Reserved		Reserved		ns 3		
CL = 7	CWL = 5	<sup>t</sup> CK (AVG)	Reserved		Reserved		ns 3		
	CWL = 6	<sup>t</sup> CK (AVG)	1.875	<2.5	Reserved		ns 2, 3		
CL = 8	CWL = 5	<sup>t</sup> CK (AVG)	Reserved		Reserved		ns 3		
	CWL = 6	<sup>t</sup> CK (AVG)	1.875	<2.5	1.875	<2.5	ns 2		
Supported CL settings		5, 6, 7, 8		5, 6, 8		CK			
Supported CWL settings		5, 6		5, 6		CK			

# Registro SDRAM\_TIM\_1

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	0h	
28-25	reg_t_rp	R/W	0h	Minimum number of DDR clock cycles from Precharge to Activate or Refresh, minus one.
24-21	reg_t_rcd	R/W	0h	Minimum number of DDR clock cycles from Activate to Read or Write, minus one.
20-17	reg_t_wr	R/W	0h	Minimum number of DDR clock cycles from last Write transfer to Pre-charge, minus one. The SDRAM initialization sequence will be started when the value of this field is changed from the previous value and the EMIF is in DDR2 mode.
16-12	reg_t_ras	R/W	0h	Minimum number of DDR clock cycles from Activate to Pre-charge, minus one. $\text{reg\_t\_ras} \geq \text{reg\_t\_rcd}$ .
11-6	reg_t_rc	R/W	0h	Minimum number of DDR clock cycles from Activate to Activate, minus one.
5-3	reg_t_rrd	R/W	0h	Minimum number of DDR clock cycles from Activate to Activate for a different bank, minus one. For an 8 bank DDR2 and DDR3, this field must be equal to $((tFAW/(4*tCK))-1)$ .
2-0	reg_t_wtr	R/W	0h	Minimum number of DDR clock cycles from last Write to Read, minus one.

# Registro SDRAM\_TIM\_2

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	
30-28	reg_t_xp	R/W	0h	Minimum number of DDR clock cycles from Powerdown exit to any command other than a Read command, minus one. For DDR2 and LPDDR1, this field must satisfy greater of tXP or tCKE.
27-25	reg_t_odt	R/W	0h	Minimum number of DDR clock cycles from ODT enable to write data driven for DDR2 and DDR3. reg_t_odt must be equal to tAOND.
24-16	reg_t_xsnr	R/W	0h	Minimum number of DDR clock cycles from Self-Refresh exit to any command other than a Read command, minus one.
15-6	reg_t_xsrd	R/W	0h	Minimum number of DDR clock cycles from Self-Refresh exit to a Read command, minus one.
5-3	reg_t_rtp	R/W	0h	Minimum number of DDR clock cycles from the last Read command to a Pre-charge command for DDR2 and DDR3, minus one.
2-0	reg_t_cke	R/W	0h	Minimum number of DDR clock cycles between pad_cke_o changes, minus one.

# Registro SDRAM\_TIM\_3

Bit	Field	Type	Reset	Description
31-28	reg_t_pdll_ul	R/W	0h	Minimum number of DDR clock cycles for PHY DLL to unlock. A value of N will be equal to N x 128 clocks.
27-24	RESERVED	R	0h	
23-21	RESERVED	R/W	0h	Reserved.
20-15	reg_zq_zqcs	R/W	0h	Number of DDR clock clock cycles for a ZQCS command, minus one.
14-13	RESERVED	R/W	0h	Reserved.
12-4	reg_t_rfc	R/W	0h	Minimum number of DDR clock cycles from Refresh or Load Mode to Refresh or Activate, minus one.
3-0	reg_t_ras_max	R/W	0h	Maximum number of reg_refresh_rate intervals from Activate to Precharge command. This field must be equal to ((tRASmax / tREFI)-1) rounded down to the next lower integer. This field is only applicable for mDDR. This field must be programmed to 0xF for other SDRAM types.

# Visión Técnica de Bajo nivel

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

### ● Vistazo introductorio

- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Problemas derivados del scaling

- La disminución del ancho del gate de un CMOS fuerza la disminución de la tensión de umbral que contra-intuitivamente termina aumentando la corriente de leakage.

# Problemas derivados del scaling

- La disminución del ancho del gate de un CMOS fuerza la disminución de la tensión de umbral que contra-intuitivamente termina aumentando la corriente de leakage.
- El estado lógico de un bit de memoria SRAM es finalmente una acumulación de cargas en un circuito capaz de mantener su estado. Si los transistores son mas pequeños, la cantidad de carga que compone un estado lógico es menor lo que la hace mas factible de ser alteradas por fenómenos externos (EMI Por ejemplo).

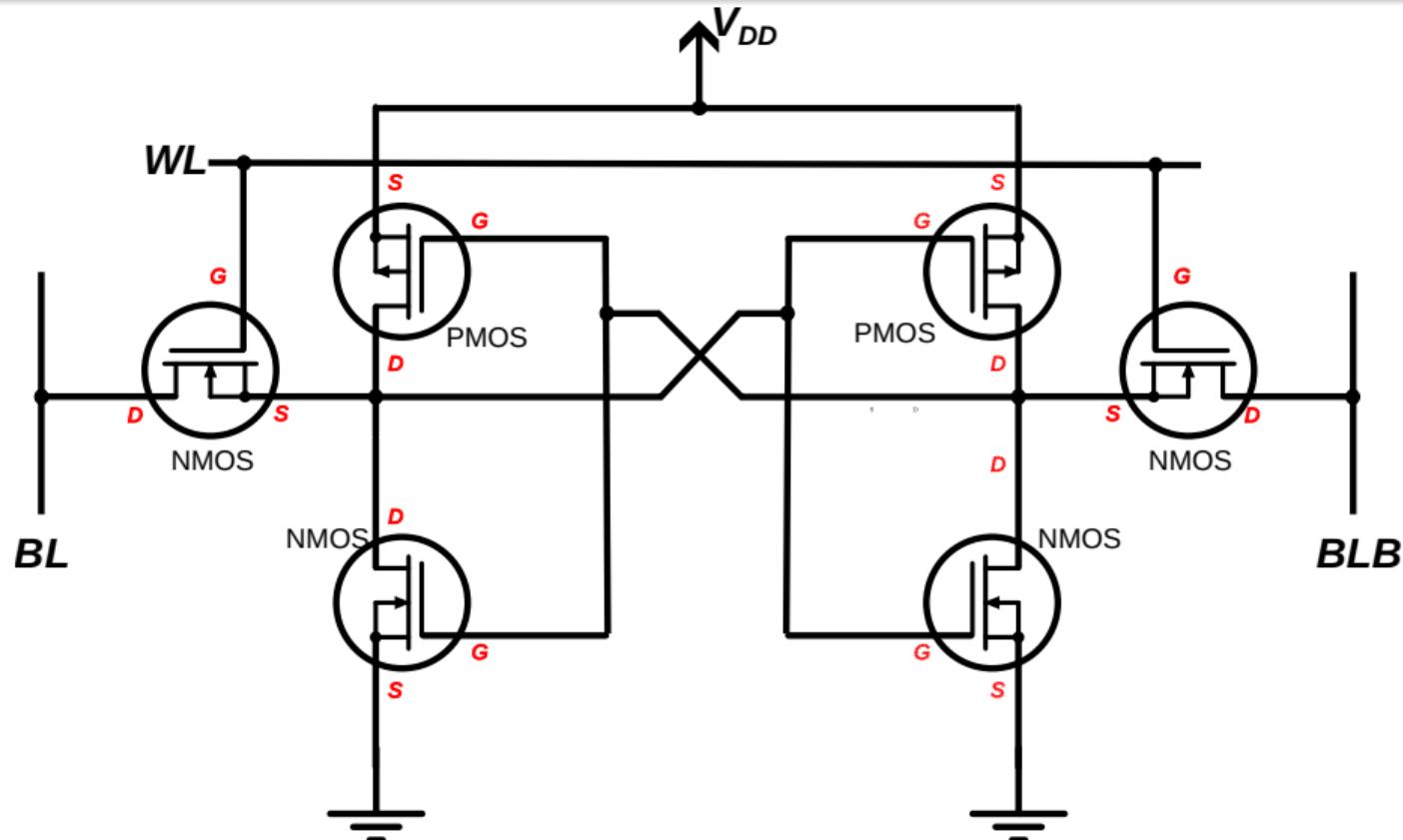
# Problemas derivados del scaling

- La disminución del ancho del gate de un CMOS fuerza la disminución de la tensión de umbral que contra-intuitivamente termina aumentando la corriente de leakage.
- El estado lógico de un bit de memoria SRAM es finalmente una acumulación de cargas en un circuito capaz de mantener su estado. Si los transistores son mas pequeños, la cantidad de carga que compone un estado lógico es menor lo que la hace mas factible de ser alteradas por fenómenos externos (EMI Por ejemplo).
- El proceso de fabricación debe necesariamente variar cuando se reduce el transistor. Esto genera diferencias entre el modelo diseñado y el obtenido

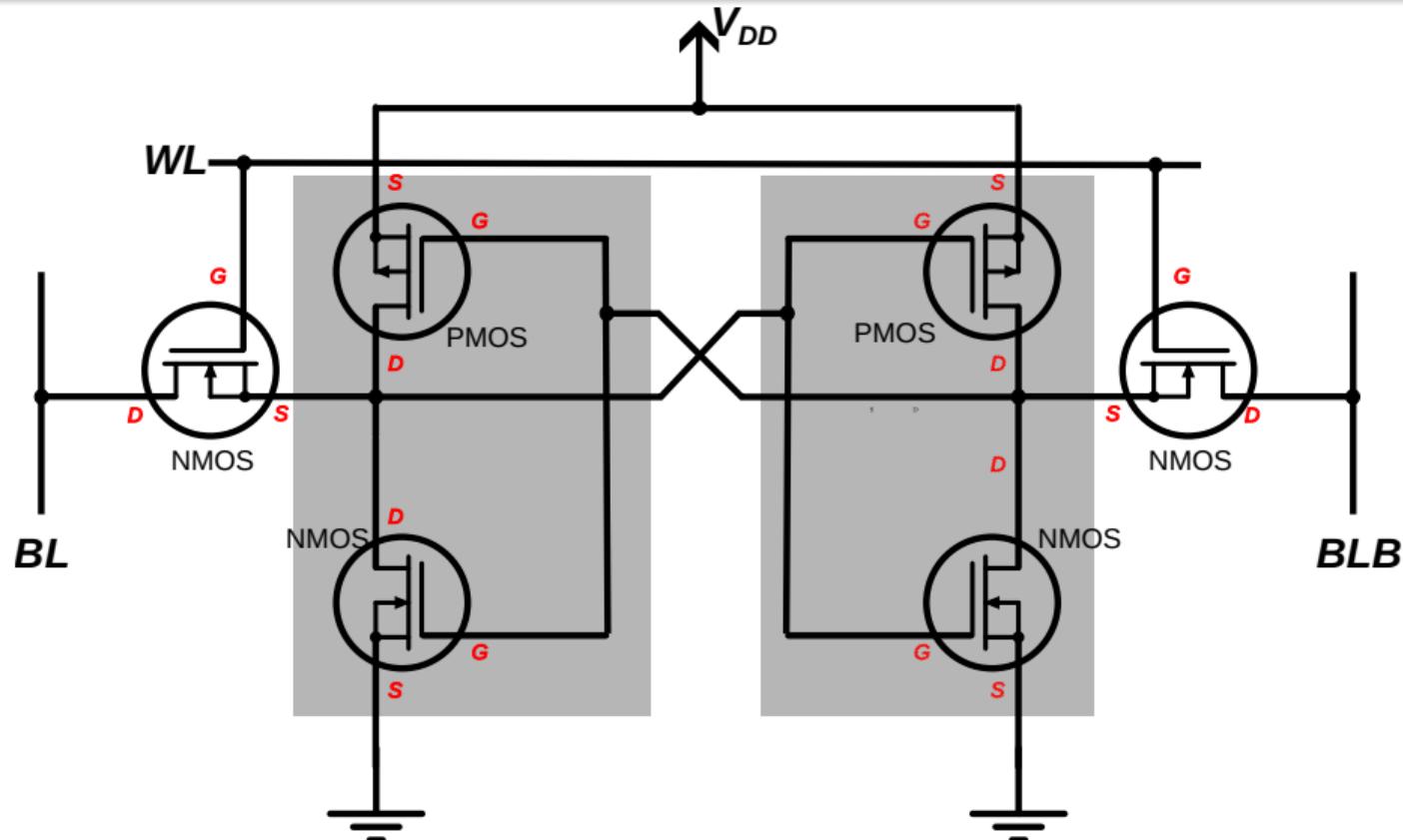
# Problemas derivados del scaling

- La disminución del ancho del gate de un CMOS fuerza la disminución de la tensión de umbral que contra-intuitivamente termina aumentando la corriente de leakage.
- El estado lógico de un bit de memoria SRAM es finalmente una acumulación de cargas en un circuito capaz de mantener su estado. Si los transistores son mas pequeños, la cantidad de carga que compone un estado lógico es menor lo que la hace mas factible de ser alteradas por fenómenos externos (EMI Por ejemplo).
- El proceso de fabricación debe necesariamente variar cuando se reduce el transistor. Esto genera diferencias entre el modelo diseñado y el obtenido
- Los caminos de conexión de transistores (wires), comienzan a ser significativos en términos de delay y de disipación.

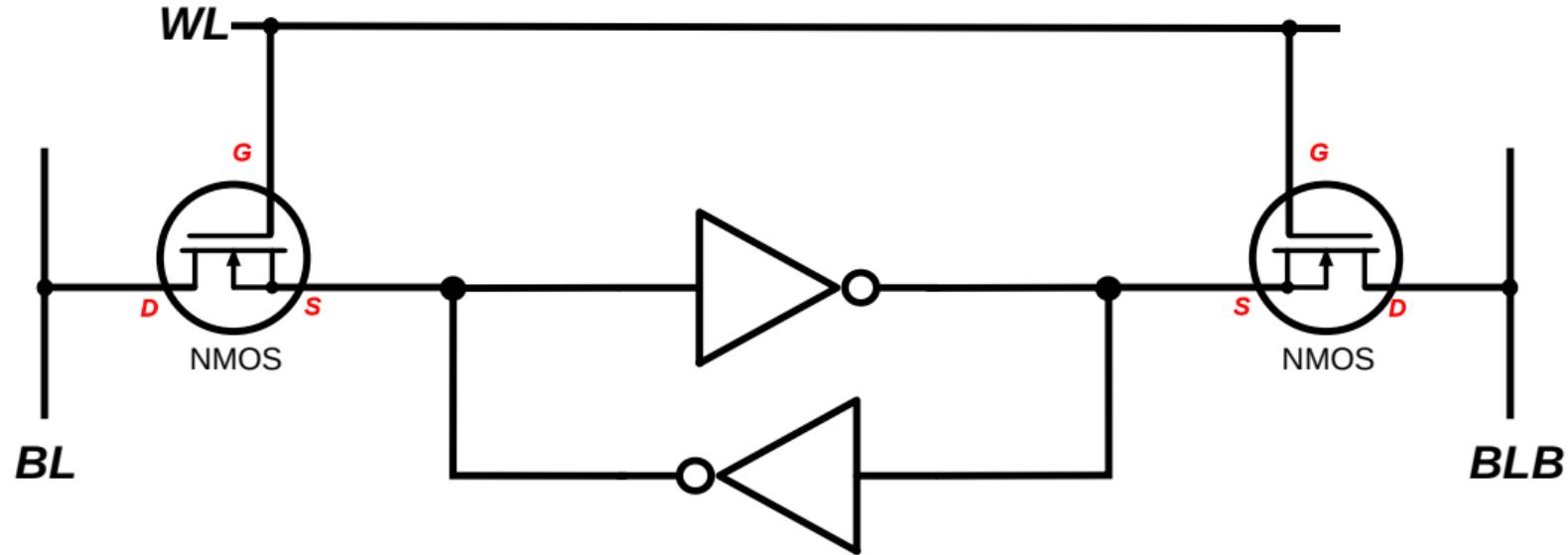
# Celda de un bit de SRAM



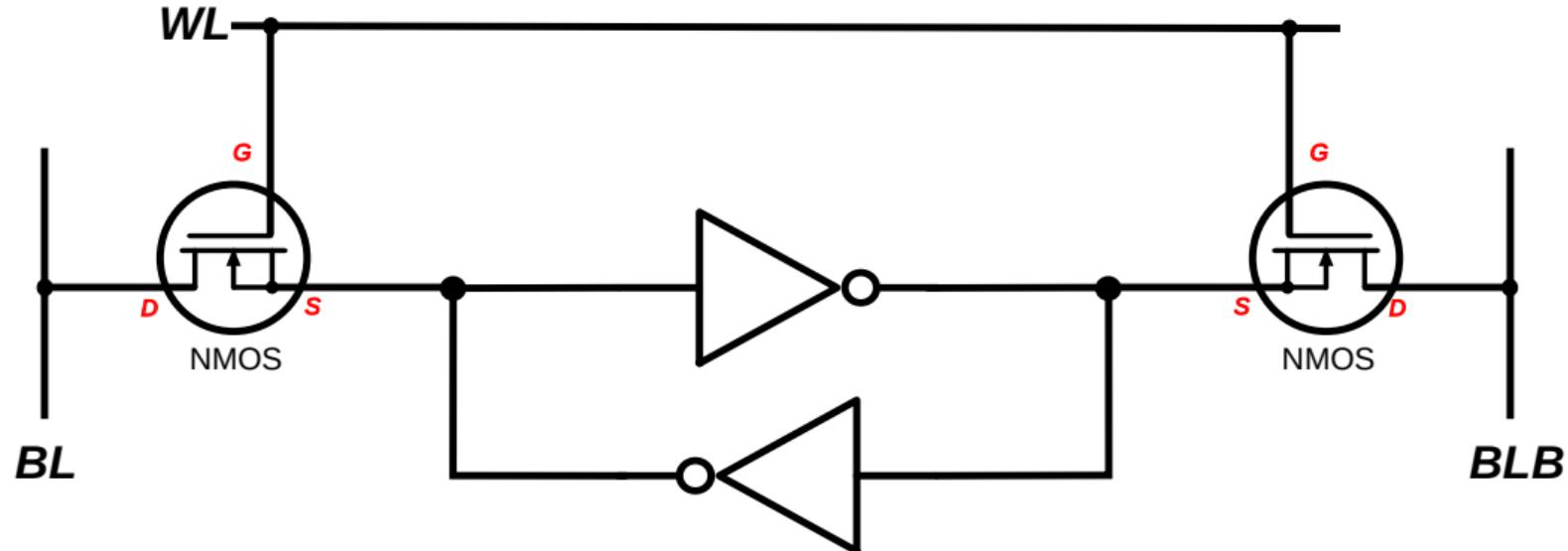
# Celda de un bit de SRAM



# Celda de un bit de SRAM

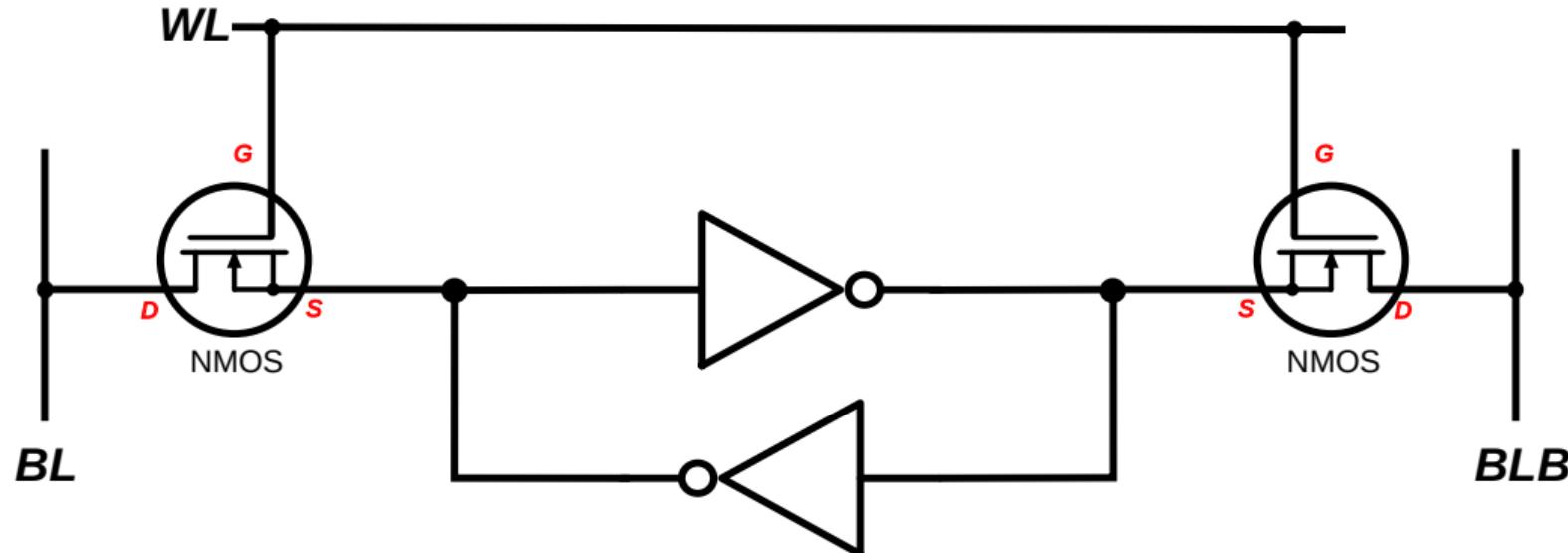


# Celda de un bit de SRAM



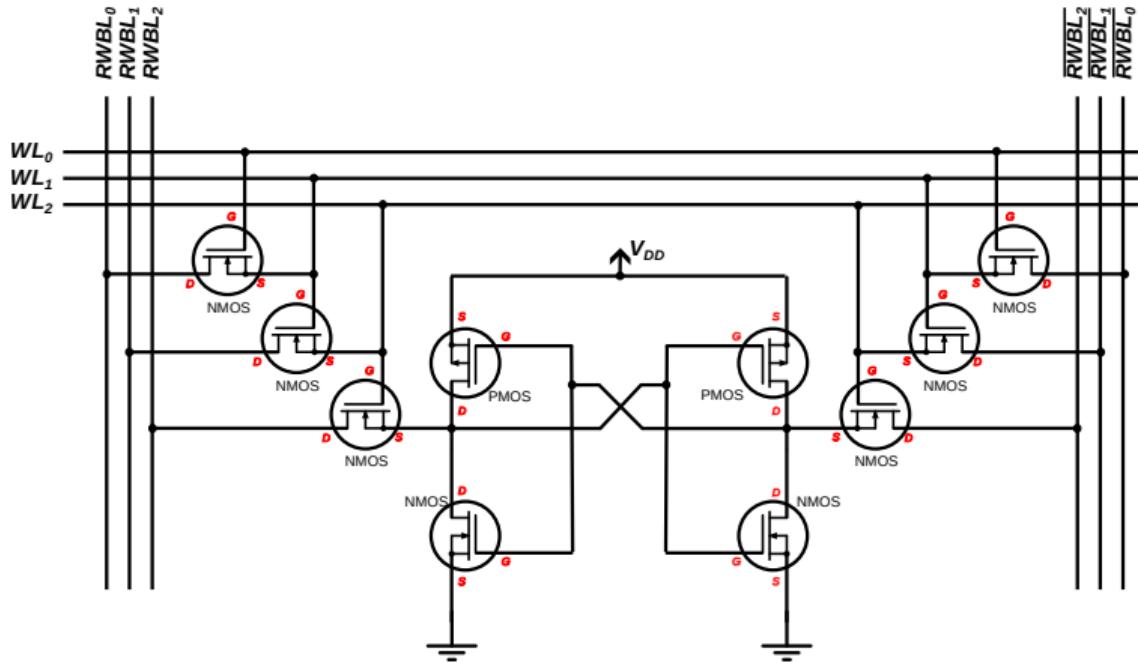
La lectura del bit se realiza activando **WL**, y detectando la tensión diferencial entre el par de Líneas de Bit (**BL** y **BLB**), inicialmente precargadas para entregar un '**1**' .

# Celda de un bit de SRAM



La escritura del bit se realiza activando **WL**, y colocando una tensión diferencial entre el par de Líneas de Bit (**BL** y **BLB**) proveniente de una fuente externa que fuerce el nuevo estado en el biestable.

# Celda de un bit de SRAM Multiport



Acepta lecturas y escrituras concurrentes en el mismo ciclo de clock. **Costo:** Dos ports 33 % mas de hardware. Implica superficie de Si y consumo de energía.

**Uso:** Cache Level1 cuando se busca performance.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
  - Coherencia de un cache
  - Protocolos de Coherencia para Multicore
  - Casos del Mundo real
- ## 5 Memorias Dinámicas
- Introducción
  - Organización interna
- ## 6 Standards
- Estado del arte
  - JEDEC SDRAM
- ## 7 Controladores de Memoria
- Introducción General
  - Arquitectura
- ## 8 Configuración
- Configuración del DRAM Device

- Entrada Salida de Datos
- ## 9 Integración con el sistema Cache
- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
  - **Decodificación**
  - Implementación
- ## 12 DRAM Detalles de implementación
- Acceso a las celdas
  - JEDEC DDR SDRAM
  - Protocolo de acceso

# Decodificación de Direcciones

# Decodificación de Direcciones

- La decodificación de direcciones es un proceso sencillo que consiste en tomar el valor lógico de una dirección de memoria y a partir de éste valor y las señales de control del bus activar las señales lógicas necesarias para realizar la operación solicitada en la posición de memoria indicada.

# Decodificación de Direcciones

- La decodificación de direcciones es un proceso sencillo que consiste en tomar el valor lógico de una dirección de memoria y a partir de éste valor y las señales de control del bus activar las señales lógicas necesarias para realizar la operación solicitada en la posición de memoria indicada.
- La forma mas simple de implementarla es inyectando la dirección de  $n$  bits en un decodificador de  $n$  a  $2^n$ .

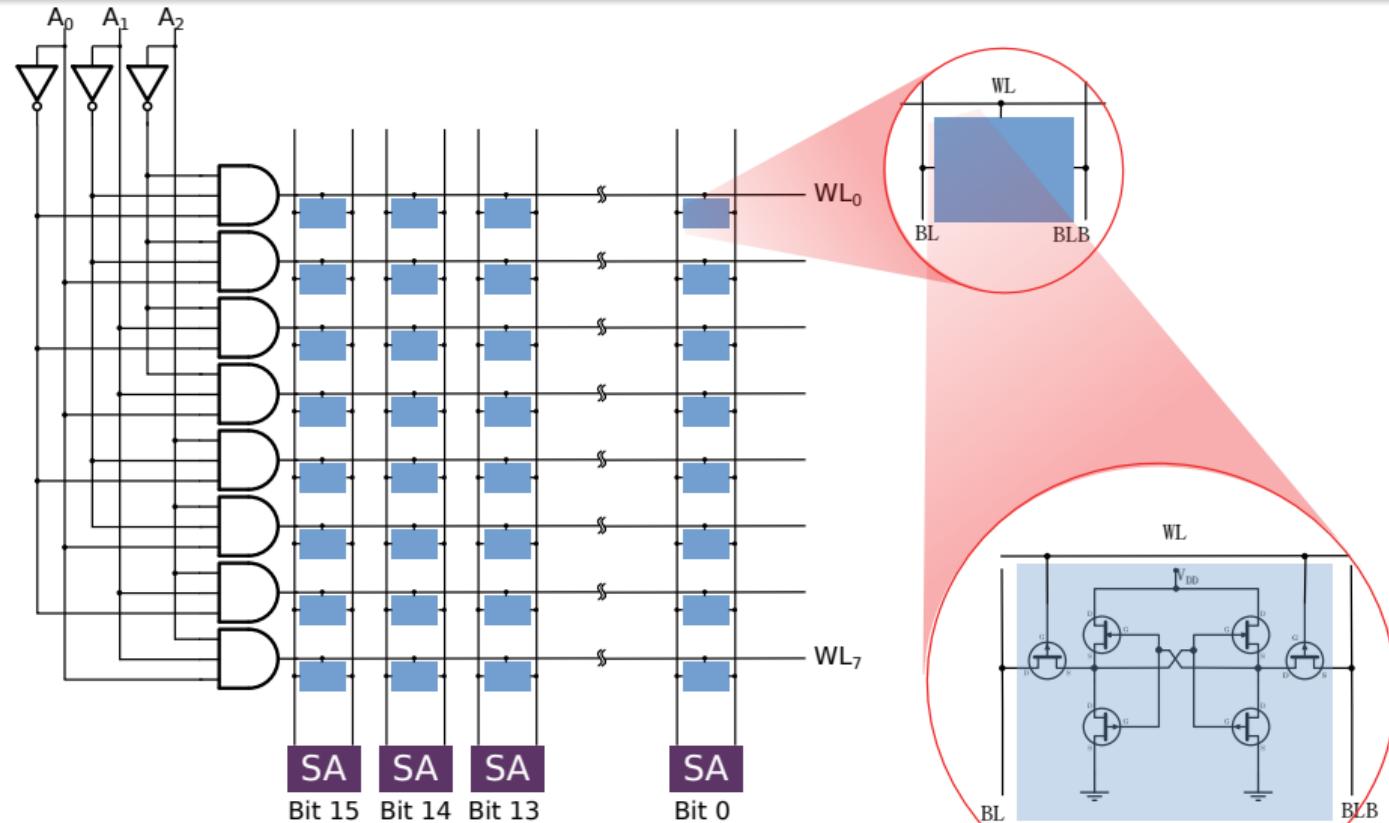
# Decodificación de Direcciones

- La decodificación de direcciones es un proceso sencillo que consiste en tomar el valor lógico de una dirección de memoria y a partir de éste valor y las señales de control del bus activar las señales lógicas necesarias para realizar la operación solicitada en la posición de memoria indicada.
- La forma mas simple de implementarla es inyectando la dirección de  $n$  bits en un decodificador de  $n$  a  $2^n$ .
- Esto involucra una operación **AND** sobre los valores posibles de entrada cuya salida active solo la **WL** correspondiente a la celda direccionada.

# Decodificación de Direcciones

- La decodificación de direcciones es un proceso sencillo que consiste en tomar el valor lógico de una dirección de memoria y a partir de éste valor y las señales de control del bus activar las señales lógicas necesarias para realizar la operación solicitada en la posición de memoria indicada.
- La forma mas simple de implementarla es inyectando la dirección de  $n$  bits en un decodificador de  $n$  a  $2^n$ .
- Esto involucra una operación **AND** sobre los valores posibles de entrada cuya salida active solo la **WL** correspondiente a la celda direccionada.
- A continuación un caso sencillo para un bus de address de 3 líneas,  $A_2$ ,  $A_1$ ,  $A_0$

# Decodificación de Direcciones



# Decodificación de Direcciones

# Decodificación de Direcciones

- La principal preocupación de diseño son los requerimientos de Fan-in y Fan-out en situaciones reales en donde la matriz representada en el gráfico anterior escala geométricamente confirme aumentan las líneas de Address.

# Decodificación de Direcciones

- La principal preocupación de diseño son los requerimientos de Fan-in y Fan-out en situaciones reales en donde la matriz representada en el gráfico anterior escala geométricamente confirme aumentan las líneas de Address.
- Esto impide por ineficiente un diseño con un solo nivel de compuertas AND.

# Decodificación de Direcciones

- La principal preocupación de diseño son los requerimientos de Fan-in y Fan-out en situaciones reales en donde la matriz representada en el gráfico anterior escala geométricamente confirme aumentan las líneas de Address.
- Esto impide por ineficiente un diseño con un solo nivel de compuertas AND.
- Los decodificadores reales se implementan con estructuras multinivel de compuertas AND.

# Decodificación de Direcciones

- La principal preocupación de diseño son los requerimientos de Fan-in y Fan-out en situaciones reales en donde la matriz representada en el gráfico anterior escala geométricamente confirme aumentan las líneas de Address.
- Esto impide por ineficiente un diseño con un solo nivel de compuertas AND.
- Los decodificadores reales se implementan con estructuras multinivel de compuertas AND.
- Es tan no trivial este tema que el diseño del decodificador interno es crítico tanto en el delay de escritura y lectura (tiempo de acceso) como en disipación de energía.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

- Organización de Sistemas Multiprocesador
- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

- Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- **Implementación**

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Diseño de Memoria

# Diseño de Memoria

- Podemos añadir memoria a un circuito de dos formas.

# Diseño de Memoria

- Podemos añadir memoria a un circuito de dos formas.
- Añadir lazos cerrados de realimentación en un circuito combinacional, en el que la memoria se manifiesta implícitamente como estados del sistema. Debido a los riesgos potenciales de temporización y las carreras, este enfoque es muy complicado y no es adecuado para la síntesis.

# Diseño de Memoria

- Podemos añadir memoria a un circuito de dos formas.
- Añadir lazos cerrados de realimentación en un circuito combinacional, en el que la memoria se manifiesta implícitamente como estados del sistema. Debido a los riesgos potenciales de temporización y las carreras, este enfoque es muy complicado y no es adecuado para la síntesis.
- Utilizar componentes de memoria prediseñados. Todas las bibliotecas de dispositivos tienen determinadas celdas de memoria, que se diseñan cuidadosamente y se analizan de manera exhaustiva.

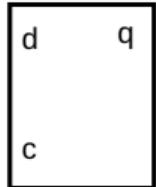
# Diseño de Memoria

- Podemos añadir memoria a un circuito de dos formas.
- Añadir lazos cerrados de realimentación en un circuito combinacional, en el que la memoria se manifiesta implícitamente como estados del sistema. Debido a los riesgos potenciales de temporización y las carreras, este enfoque es muy complicado y no es adecuado para la síntesis.
- Utilizar componentes de memoria prediseñados. Todas las bibliotecas de dispositivos tienen determinadas celdas de memoria, que se diseñan cuidadosamente y se analizan de manera exhaustiva.
- Estos elementos se dividen en dos categorías: latch y flip-flop (FF).

# Diseño de Memoria

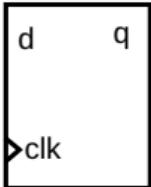
- Podemos añadir memoria a un circuito de dos formas.
- Añadir lazos cerrados de realimentación en un circuito combinacional, en el que la memoria se manifiesta implícitamente como estados del sistema. Debido a los riesgos potenciales de temporización y las carreras, este enfoque es muy complicado y no es adecuado para la síntesis.
- Utilizar componentes de memoria prediseñados. Todas las bibliotecas de dispositivos tienen determinadas celdas de memoria, que se diseñan cuidadosamente y se analizan de manera exhaustiva.
- Estos elementos se dividen en dos categorías: latch y flip-flop (FF).
- Repasemos las características básicas de un latch D y un FF D.

# Latch D

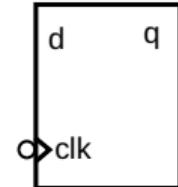


Latch D

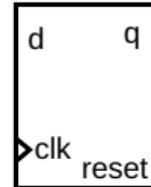
<u>c</u>	<u>q*</u>
0	q
1	d

Flip Flop D disparo  
por flanco Positivo

clk	q*
0	q
1	q

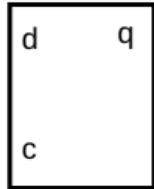
Flip Flop D disparo  
por flanco Negativo

clk	q*
0	q
1	q

Flip Flop D con  
reset asincrónico

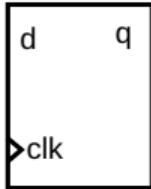
reset	clk	q*
1	-	0
0	0	q
0	1	q
0	1	d

# Latch D

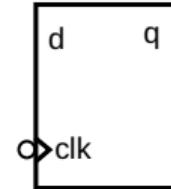


Latch D

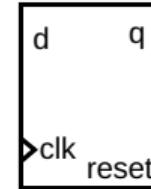
<u>c</u>	<u>q*</u>
0	q
1	d

Flip Flop D disparo  
por flanco Positivo

clk	q*
0	q
1	q

Flip Flop D disparo  
por flanco Negativo

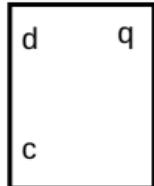
clk	q*
0	q
1	q

Flip Flop D con  
reset asincrónico

reset	clk	q*
1	-	0
0	0	q
0	1	q
0	1	d

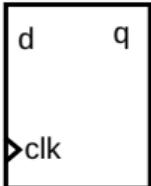
- El símbolo y la tabla de verdad de un latch D se muestran a la izquierda de la figura

# Latch D

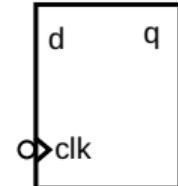


Latch D

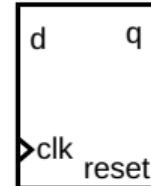
<u>c</u>	<u>q*</u>
0	q
1	d

Flip Flop D disparo  
por flanco Positivo

clk	q*
0	q
1	q

Flip Flop D disparo  
por flanco Negativo

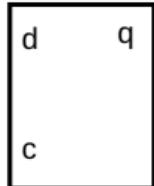
clk	q*
0	q
1	q

Flip Flop D con  
reset asincrónico

reset	clk	q*
1	-	0
0	0	q
0	1	q
0	1	d

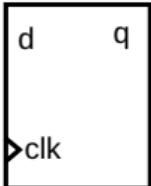
- El símbolo y la tabla de verdad de un latch D se muestran a la izquierda de la figura
- Utilizamos \* para representar el siguiente valor. Por tanto  $q^*$  significa el siguiente valor de  $q$ .

# Latch D

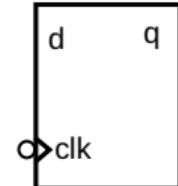


Latch D

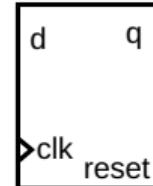
<u>c</u>	<u>q*</u>
0	q
1	d

Flip Flop D disparo  
por flanco Positivo

clk	q*
0	q
1	d

Flip Flop D disparo  
por flanco Negativo

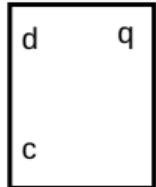
clk	q*
0	q
1	d

Flip Flop D con  
reset asincrónico

reset	clk	q*
1	-	0
0	0	q
0	1	q
0	1	d

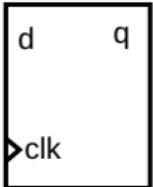
- El símbolo y la tabla de verdad de un latch D se muestran a la izquierda de la figura
- Utilizamos \* para representar el siguiente valor. Por tanto  $q^*$  significa el siguiente valor de  $q$ .
- Las entradas  $c$  y  $d$  pueden considerarse como señal de control y entrada de datos respectivamente.

# Latch D

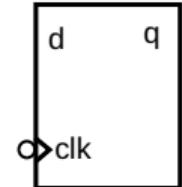


Latch D

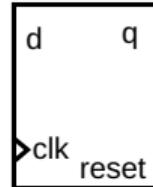
<u>c</u>	<u>q*</u>
0	q
1	d

Flip Flop D disparo  
por flanco Positivo

clk	q*
0	q
1	d

Flip Flop D disparo  
por flanco Negativo

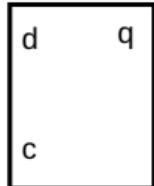
clk	q*
0	q
1	d

Flip Flop D con  
reset asincrónico

reset	clk	q*
1	-	0
0	0	q
0	1	q
0	1	d

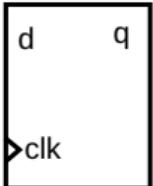
- El símbolo y la tabla de verdad de un latch D se muestran a la izquierda de la figura
- Utilizamos \* para representar el siguiente valor. Por tanto  $q^*$  significa el siguiente valor de  $q$ .
- Las entradas  $c$  y  $d$  pueden considerarse como señal de control y entrada de datos respectivamente.
- Cuando se activa  $c$ , el dato presente en la entrada  $d$  pasa a la salida  $q$ .

# Latch D

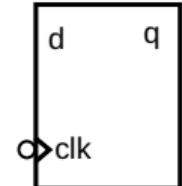


Latch D

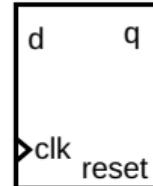
<u>c</u>	<u>q*</u>
0	q
1	d

Flip Flop D disparo  
por flanco Positivo

clk	q*
0	q
1	d

Flip Flop D disparo  
por flanco Negativo

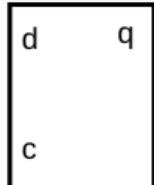
clk	q*
0	q
1	d

Flip Flop D con  
reset asincrónico

reset	clk	q*
1	-	0
0	0	q
0	1	q
0	1	d

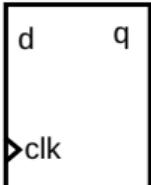
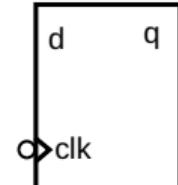
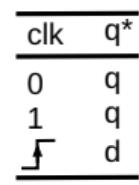
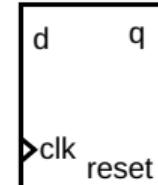
- El símbolo y la tabla de verdad de un latch D se muestran a la izquierda de la figura
- Utilizamos \* para representar el siguiente valor. Por tanto  $q^*$  significa el siguiente valor de  $q$ .
- Las entradas  $c$  y  $d$  pueden considerarse como señal de control y entrada de datos respectivamente.
- Cuando se activa  $c$ , el dato presente en la entrada  $d$  pasa a la salida  $q$ .
- Cuando  $c$  se inactiva la salida  $q$  se mantiene.

# Latch D



Latch D

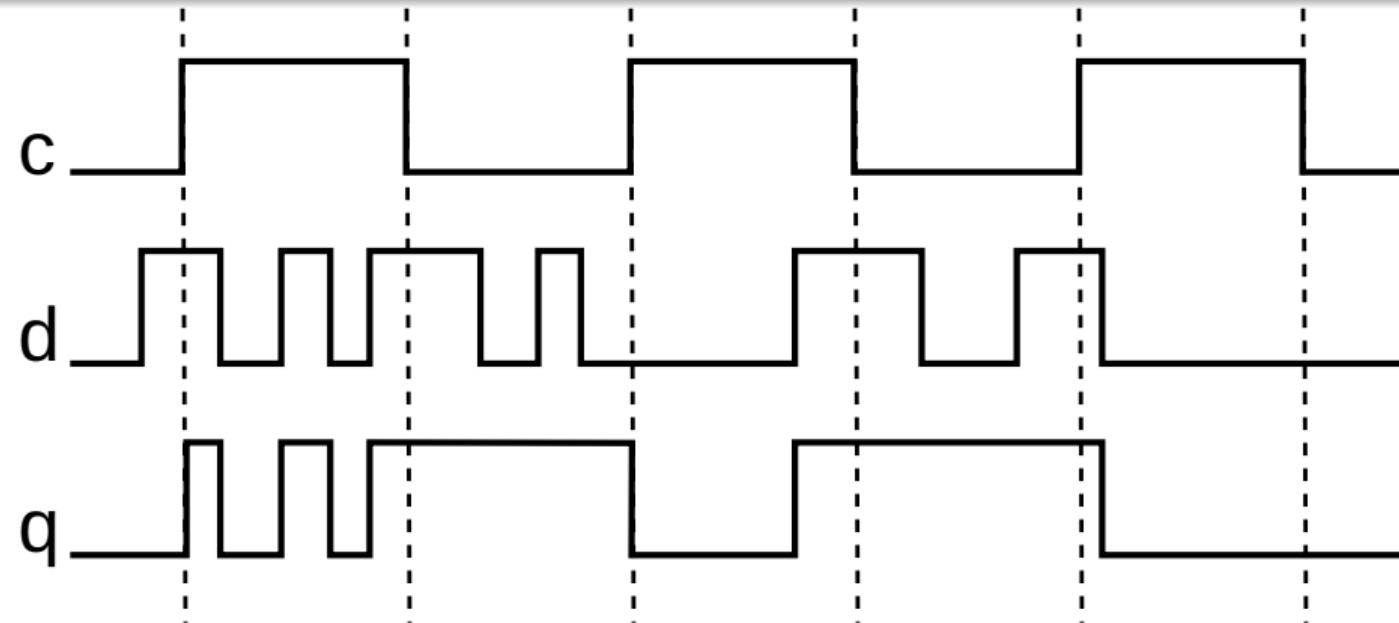
<u>c</u>	<u>q*</u>
0	q
1	d

Flip Flop D disparo  
por flanco PositivoFlip Flop D disparo  
por flanco NegativoFlip Flop D con  
reset asincrónico

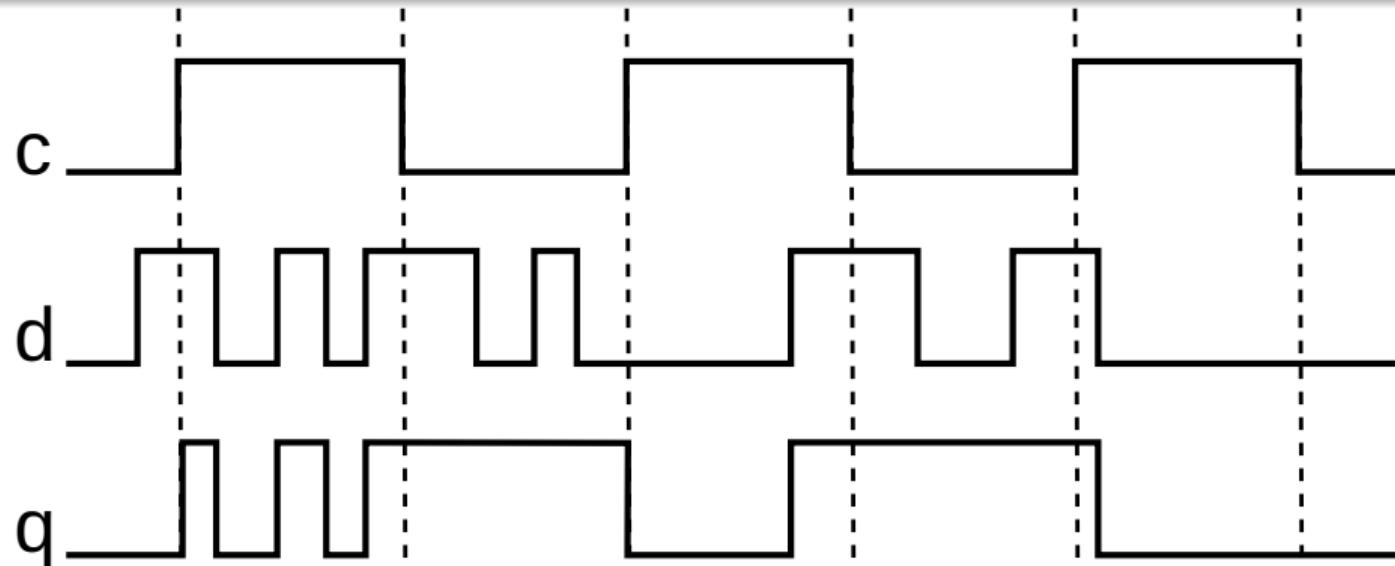
reset	clk	q*
1	-	0
0	0	q
0	1	q
0	1	d

- El símbolo y la tabla de verdad de un latch D se muestran a la izquierda de la figura
- Utilizamos \* para representar el siguiente valor. Por tanto  $q^*$  significa el siguiente valor de  $q$ .
- Las entradas  $c$  y  $d$  pueden considerarse como señal de control y entrada de datos respectivamente.
- Cuando se activa  $c$ , el dato presente en la entrada  $d$  pasa a la salida  $q$ .
- Cuando  $c$  se inactiva la salida  $q$  se mantiene.
- Dado que el funcionamiento del latch D depende del nivel de la señal de control, decimos que es activo por nivel.

## Latch D

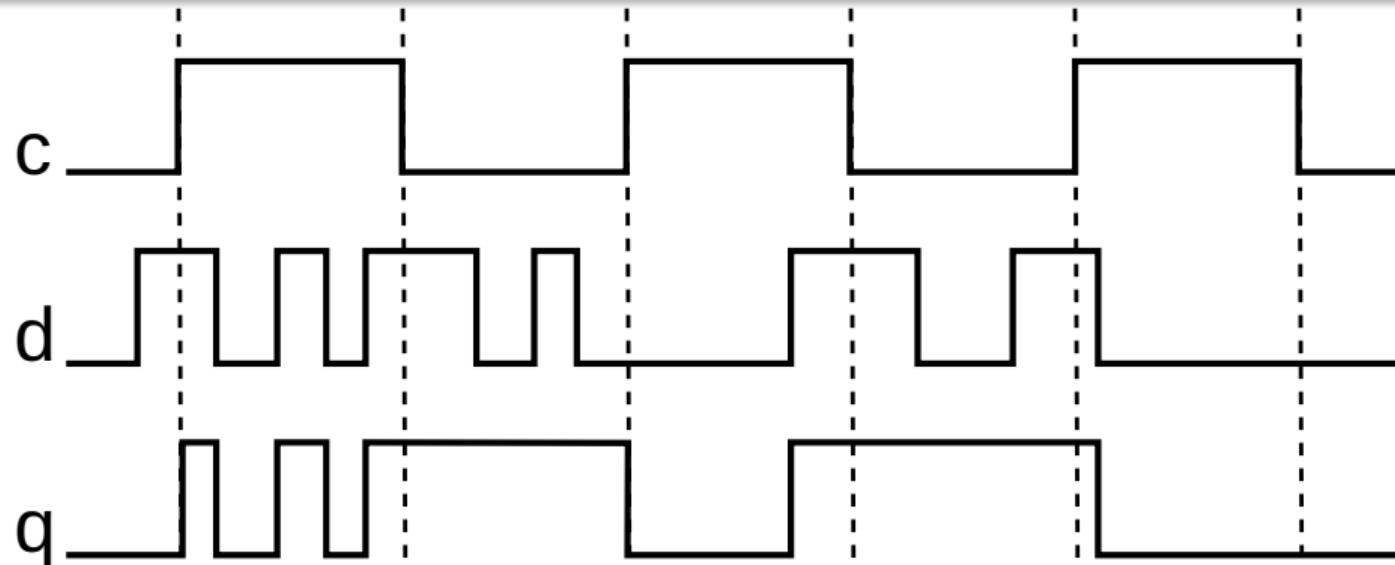


# Latch D



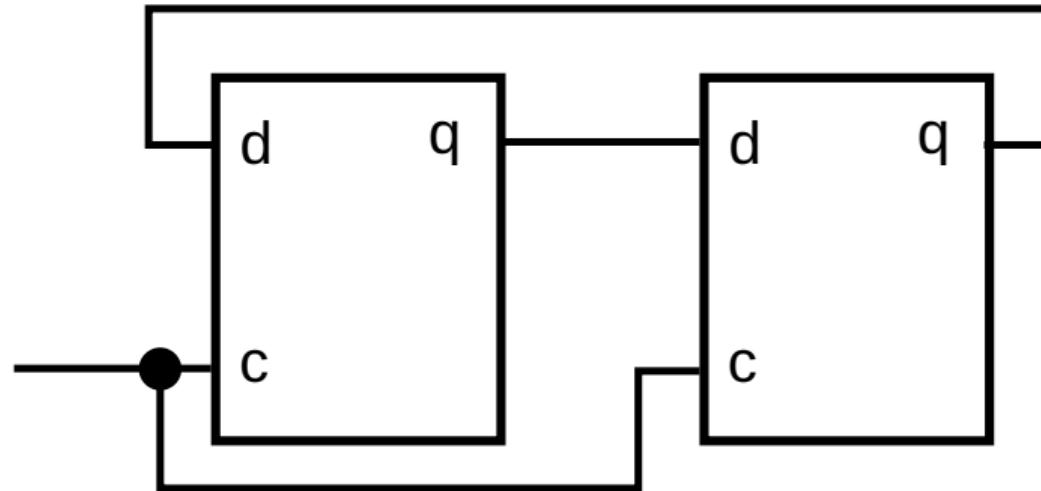
- En el diagrama de temporización, los datos de entrada se almacenan realmente en el latch en el flanco descendente de la señal de control.

# Latch D



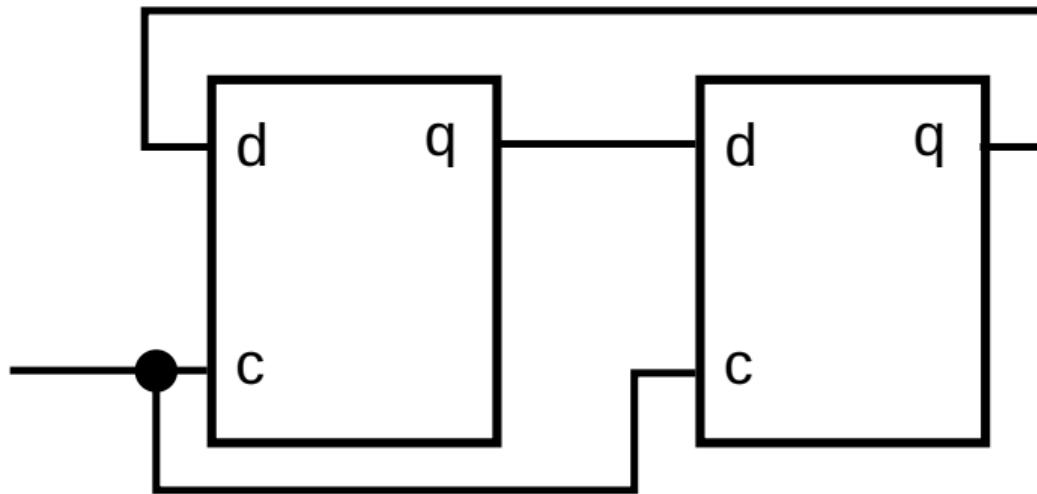
- En el diagrama de temporización, los datos de entrada se almacenan realmente en el latch en el flanco descendente de la señal de control.
- El latch es “transparente” cuando **c** está activa, los riesgos de carreras son grandes si el circuito introduce lazos.

## Latch D



## Data Swapping using Latch D

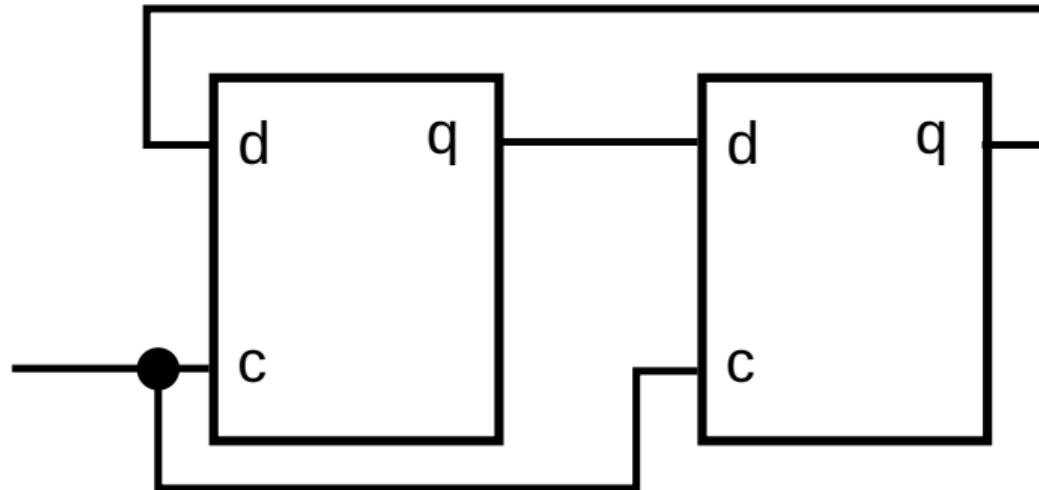
# Latch D



## Data Swapping usando Latch D

- El circuito de la Figura intenta intercambiar el contenido de dos latches.

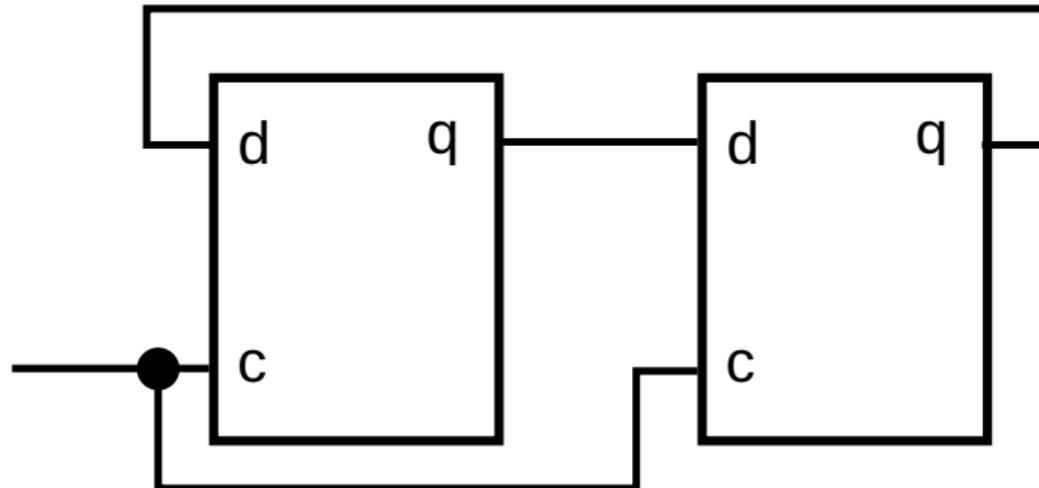
# Latch D



## Data Swapping usando Latch D

- El circuito de la Figura intenta intercambiar el contenido de dos latches.
- La carrera ocurre cuando se activa *c*.

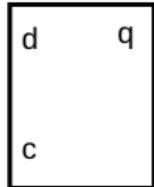
# Latch D



## Data Swapping using Latch D

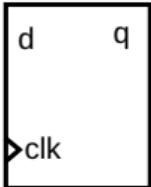
- El circuito de la Figura intenta intercambiar el contenido de dos latches.
- La carrera ocurre cuando se activa **c**.
- Debido a esta complicación potencial de la temporización, normalmente no utilizamos latches en síntesis.

# Flip Flop D

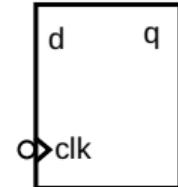


Latch D

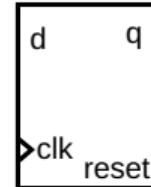
<u>c</u>	<u>q*</u>
0	q
1	d

Flip Flop D disparo  
por flanco Positivo

clk	q*
0	q
1	q

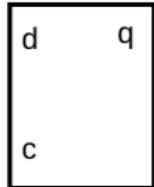
Flip Flop D disparo  
por flanco Negativo

clk	q*
0	q
1	q

Flip Flop D con  
reset asincrónico

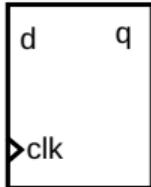
reset	clk	q*
1	-	0
0	0	q
0	1	q
0	1	d

# Flip Flop D

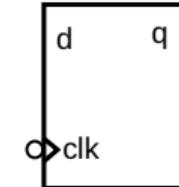


Latch D

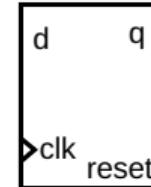
c	$q^*$
0	q
1	d

Flip Flop D disparo  
por flanco Positivo

clk	$q^*$
0	q
1	q

Flip Flop D disparo  
por flanco Negativo

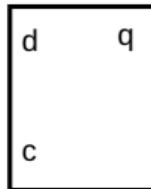
clk	$q^*$
0	q
1	d

Flip Flop D con  
reset asincrónico

reset	clk	$q^*$
1	-	0
0	0	q
0	1	q
0	1	d

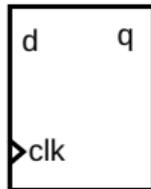
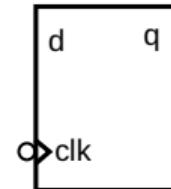
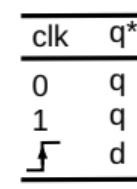
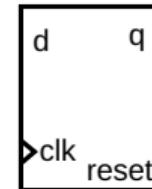
- El símbolo y la tabla de verdad de algunas variantes de un Flip Flop D se muestran en la Figura, a la derecha del latch D

# Flip Flop D



Latch D

c	$q^*$
0	q
1	d

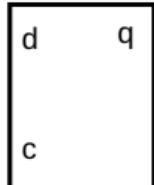
Flip Flop D disparo  
por flanco PositivoFlip Flop D disparo  
por flanco Negativo

reset	clk	$q^*$
1	-	0
0	0	q
0	1	q
0	F	d

Flip Flop D con  
reset asincrónico

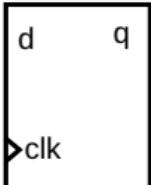
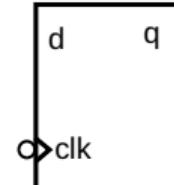
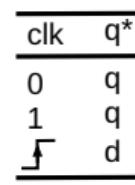
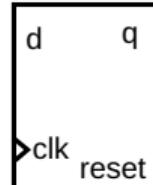
- El símbolo y la tabla de verdad de algunas variantes de un Flip Flop D se muestran en la Figura, a la derecha del latch D
- Tienen una señal de control especial conocida como señal de reloj, etiquetada como **clk** en el diagrama.

# Flip Flop D



Latch D

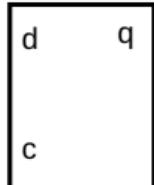
c	$q^*$
0	q
1	d

Flip Flop D disparo  
por flanco PositivoFlip Flop D disparo  
por flanco NegativoFlip Flop D con  
reset asincrónico

reset	clk	$q^*$
1	-	0
0	0	q
0	1	q
0	1	d

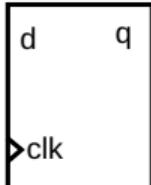
- El símbolo y la tabla de verdad de algunas variantes de un Flip Flop D se muestran en la Figura, a la derecha del latch D
- Tienen una señal de control especial conocida como señal de reloj, etiquetada como **clk** en el diagrama.
- El FF D se activa solo durante las transiciones de esta señal. Durante el resto del período la salida permanece constante.

# Flip Flop D

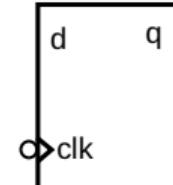


Latch D

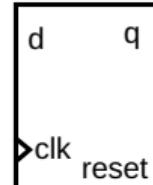
c	$q^*$
0	q
1	d

Flip Flop D disparo  
por flanco Positivo

clk	$q^*$
0	q
1	q

Flip Flop D disparo  
por flanco Negativo

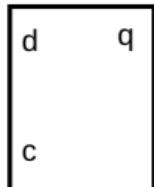
clk	$q^*$
0	q
1	q

Flip Flop D con  
reset asincrónico

reset	clk	$q^*$
1	-	0
0	0	q
0	1	q
0	1	d

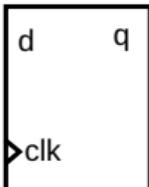
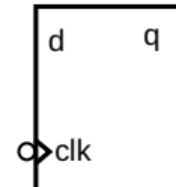
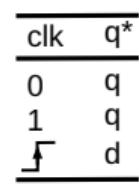
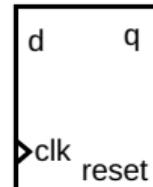
- El símbolo y la tabla de verdad de algunas variantes de un Flip Flop D se muestran en la Figura, a la derecha del latch D
- Tienen una señal de control especial conocida como señal de reloj, etiquetada como **clk** en el diagrama.
- El FF D se activa solo durante las transiciones de esta señal. Durante el resto del período la salida permanece constante.
- Si se activa en la transición de '0' a '1', se lo conoce como activo por flanco creciente o positivo.

# Flip Flop D



Latch D

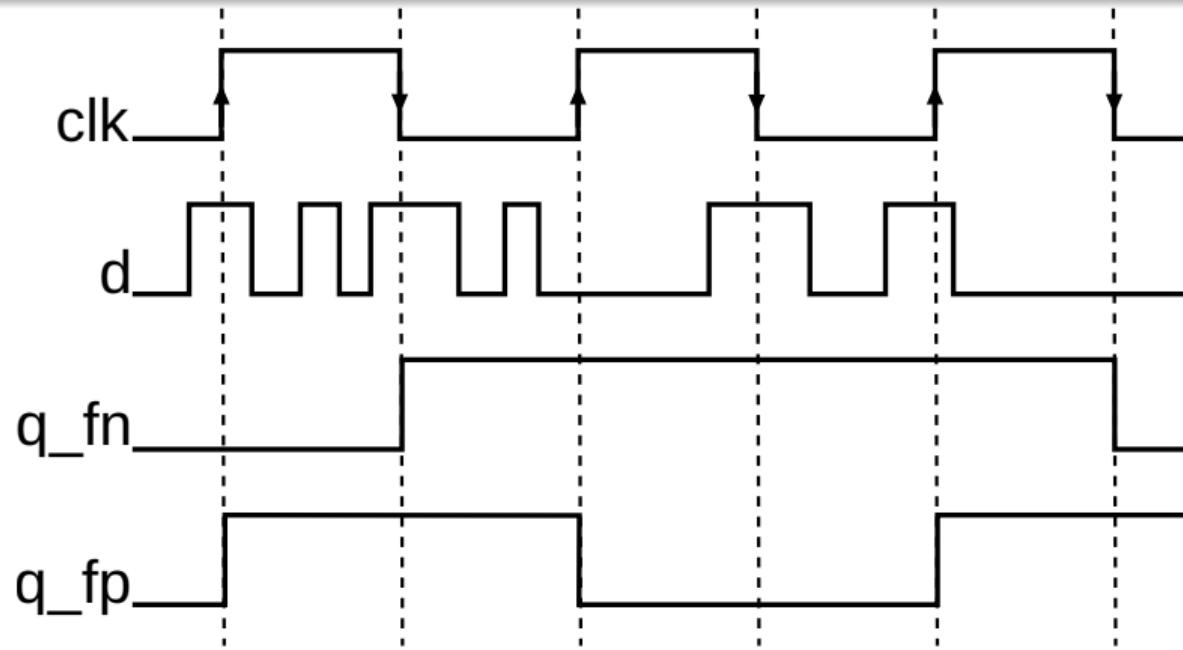
c	q*
0	q
1	d

Flip Flop D disparo  
por flanco PositivoFlip Flop D disparo  
por flanco NegativoFlip Flop D con  
reset asincrónico

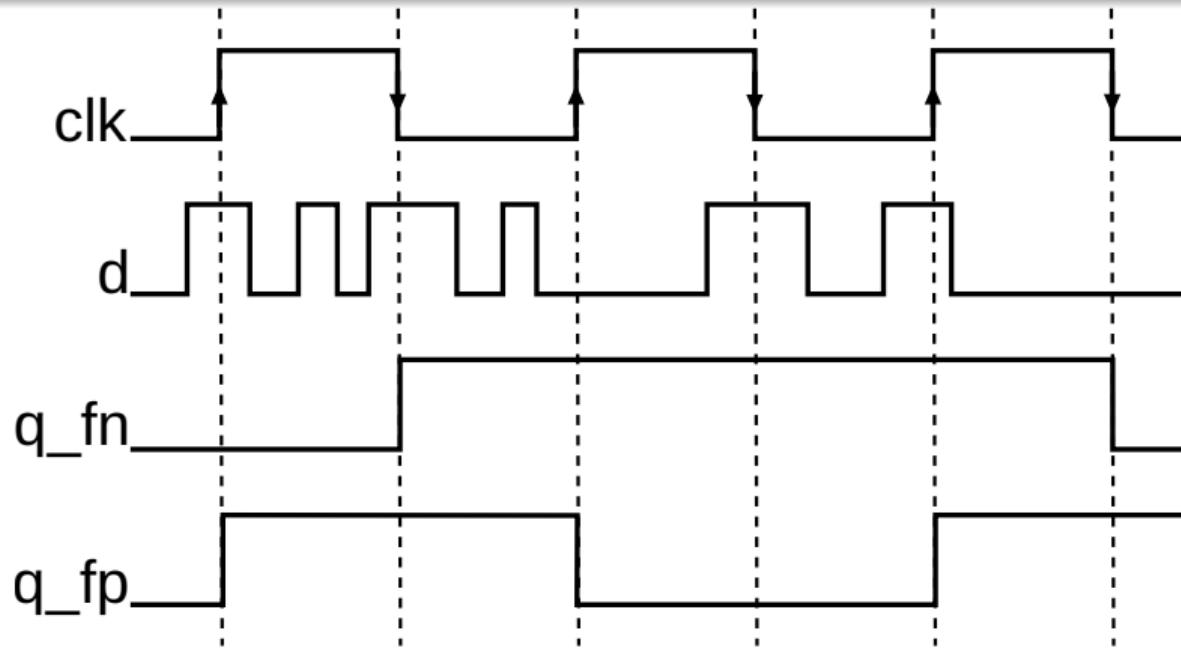
reset	clk	q*
1	-	0
0	0	q
0	1	q
0	1	d

- El símbolo y la tabla de verdad de algunas variantes de un Flip Flop D se muestran en la Figura, a la derecha del latch D
- Tienen una señal de control especial conocida como señal de reloj, etiquetada como **clk** en el diagrama.
- El FF D se activa solo durante las transiciones de esta señal. Durante el resto del período la salida permanece constante.
- Si se activa en la transición de '0' a '1', se lo conoce como activo por flanco creciente o positivo.
- Si se activa en la transición de '1' a '0', se lo conoce como activo por flanco decreciente o negativo.

# Flip Flop D



# Flip Flop D



- En otras palabras, en el flanco activo del reloj, un FF D toma una muestra de los datos de entrada, almacena el valor en la memoria y pasa el valor a la salida. La salida, refleja el valor almacenado, y no cambia hasta el siguiente flanco activo.

# Flip Flop D

# Flip Flop D

- La señal de reloj, ***clk***, funciona como una señal de muestreo, que toma una muestra de los datos de entrada, ***d***, en el flanco activo.

# Flip Flop D

- La señal de reloj, ***clk***, funciona como una señal de muestreo, que toma una muestra de los datos de entrada, ***d***, en el flanco activo.
- La señal de reloj juega un papel clave en un circuito secuencial y añadimos un pequeño triángulo, en el diagrama para enfatizar el uso de un FF disparado por flanco.

# Flip Flop D

- La señal de reloj, ***clk***, funciona como una señal de muestreo, que toma una muestra de los datos de entrada, ***d***, en el flanco activo.
- La señal de reloj juega un papel clave en un circuito secuencial y añadimos un pequeño triángulo, en el diagrama para enfatizar el uso de un FF disparado por flanco.
- La propiedad de muestreo de los FF tiene varias ventajas.

# Flip Flop D

- La señal de reloj, ***clk***, funciona como una señal de muestreo, que toma una muestra de los datos de entrada, ***d***, en el flanco activo.
- La señal de reloj juega un papel clave en un circuito secuencial y añadimos un pequeño triángulo, en el diagrama para enfatizar el uso de un FF disparado por flanco.
- La propiedad de muestreo de los FF tiene varias ventajas.
  - Las variaciones y “glitches” entre dos flancos ascendentes no afectan al contenido de la memoria.

# Flip Flop D

- La señal de reloj, ***clk***, funciona como una señal de muestreo, que toma una muestra de los datos de entrada, ***d***, en el flanco activo.
- La señal de reloj juega un papel clave en un circuito secuencial y añadimos un pequeño triángulo, en el diagrama para enfatizar el uso de un FF disparado por flanco.
- La propiedad de muestreo de los FF tiene varias ventajas.
  - Las variaciones y “glitches” entre dos flancos ascendentes no afectan al contenido de la memoria.
  - No hay condiciones de carrera en un lazo cerrado de realimentación. Si reconstruimos el circuito de bit Swap anterior sustituyendo los latches por los FFs, como éstos intercambian sus contenidos en cada flanco activo del reloj, el circuito funciona como se espera.

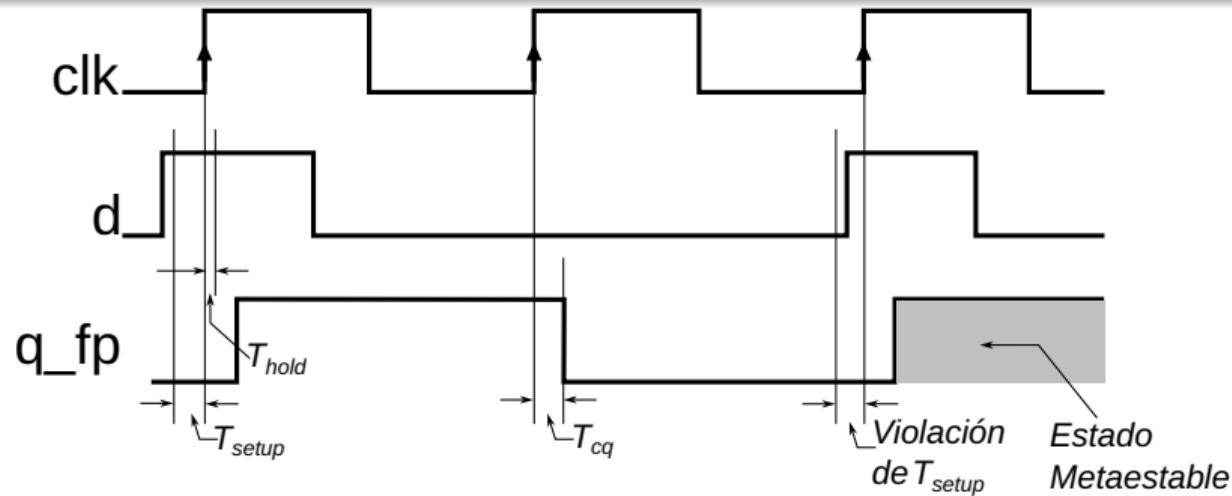
# Flip Flop D

- La señal de reloj, ***clk***, funciona como una señal de muestreo, que toma una muestra de los datos de entrada, ***d***, en el flanco activo.
- La señal de reloj juega un papel clave en un circuito secuencial y añadimos un pequeño triángulo, en el diagrama para enfatizar el uso de un FF disparado por flanco.
- La propiedad de muestreo de los FF tiene varias ventajas.
  - Las variaciones y “glitches” entre dos flancos ascendentes no afectan al contenido de la memoria.
  - No hay condiciones de carrera en un lazo cerrado de realimentación. Si reconstruimos el circuito de bit Swap anterior sustituyendo los latches por los FFs, como éstos intercambian sus contenidos en cada flanco activo del reloj, el circuito funciona como se espera.
  - La desventaja del FF es el tamaño del circuito, que es aproximadamente dos veces mayor que el de un latch D.

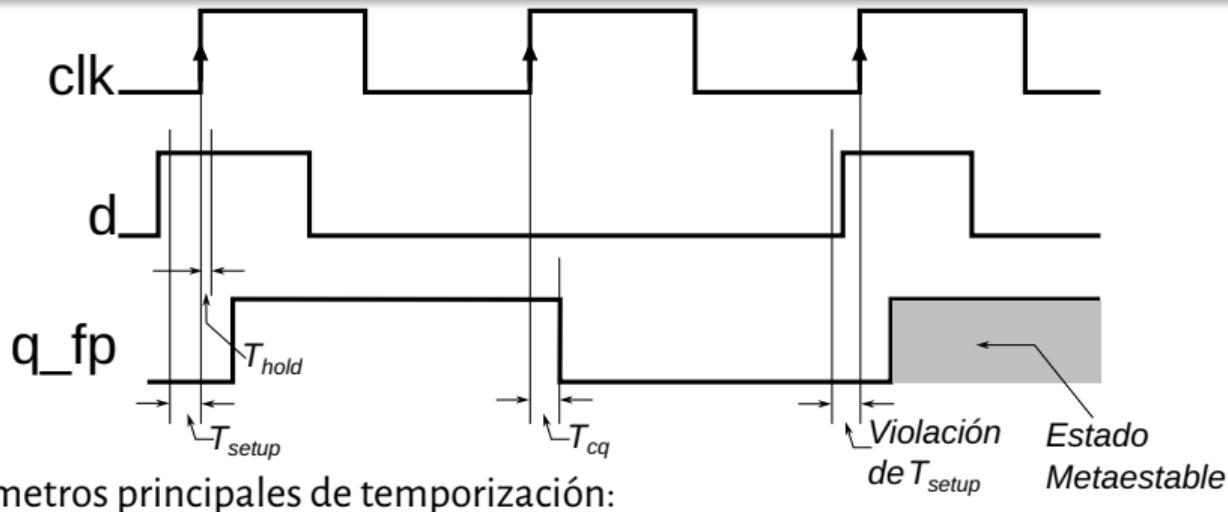
# Flip Flop D

- La señal de reloj, ***clk***, funciona como una señal de muestreo, que toma una muestra de los datos de entrada, ***d***, en el flanco activo.
- La señal de reloj juega un papel clave en un circuito secuencial y añadimos un pequeño triángulo, en el diagrama para enfatizar el uso de un FF disparado por flanco.
- La propiedad de muestreo de los FF tiene varias ventajas.
  - Las variaciones y “glitches” entre dos flancos ascendentes no afectan al contenido de la memoria.
  - No hay condiciones de carrera en un lazo cerrado de realimentación. Si reconstruimos el circuito de bit Swap anterior sustituyendo los latches por los FFs, como éstos intercambian sus contenidos en cada flanco activo del reloj, el circuito funciona como se espera.
  - La desventaja del FF es el tamaño del circuito, que es aproximadamente dos veces mayor que el de un latch D.
  - Dado que sus ventajas superan con creces la desventaja del tamaño, los circuitos secuenciales actuales utilizan normalmente D FF como elementos de almacenamiento.

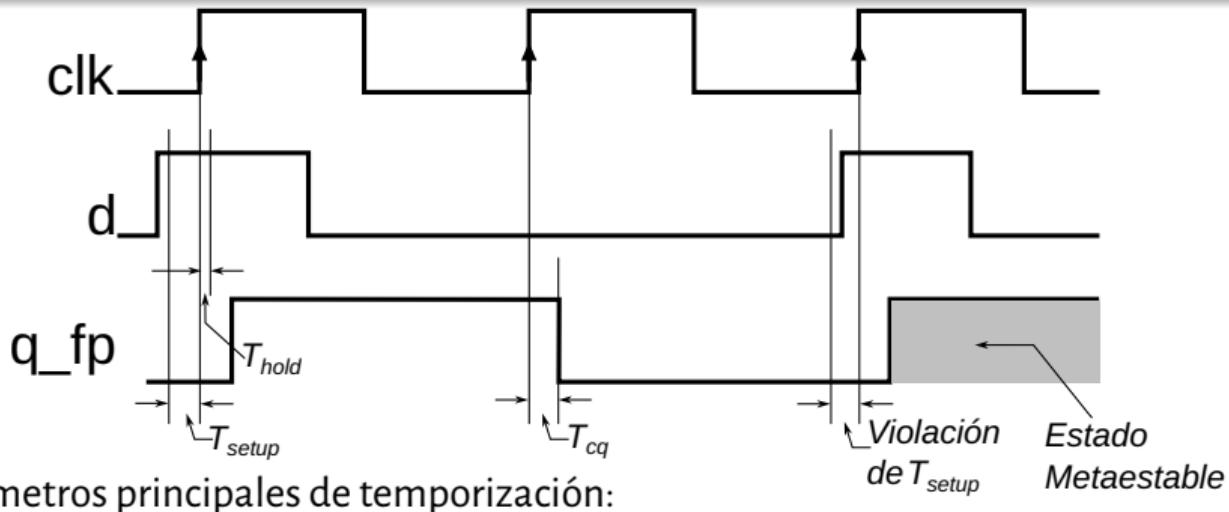
# Flip Flop D



# Flip Flop D



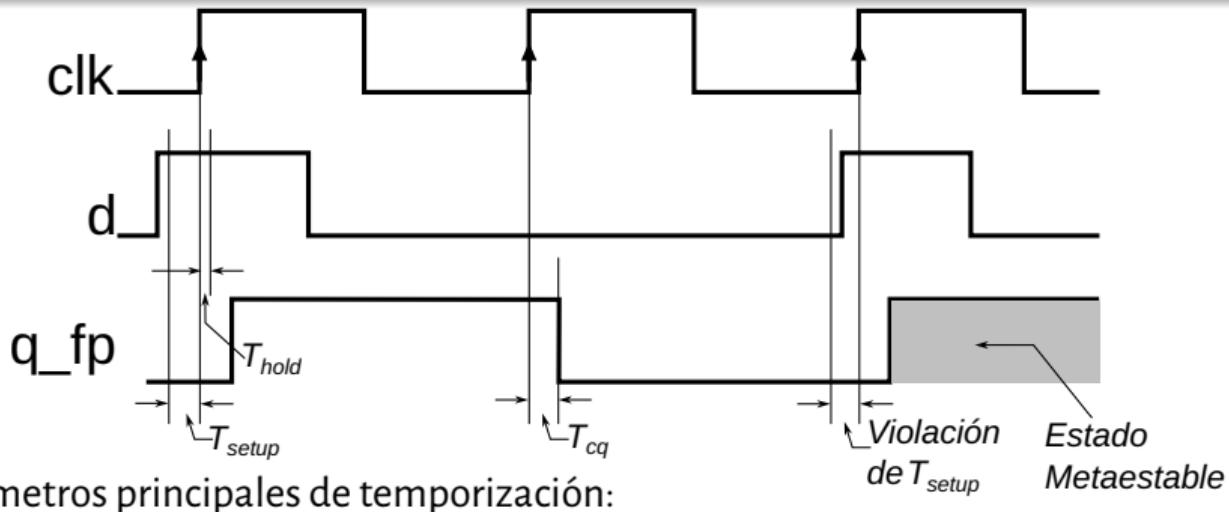
# Flip Flop D



- tres parámetros principales de temporización:

**T<sub>cq</sub>**: Retardo de **clk** a **q**. Tiempo de propagación necesario para que la entrada **d** aparezca en la salida **q** después del flanco activo.

# Flip Flop D

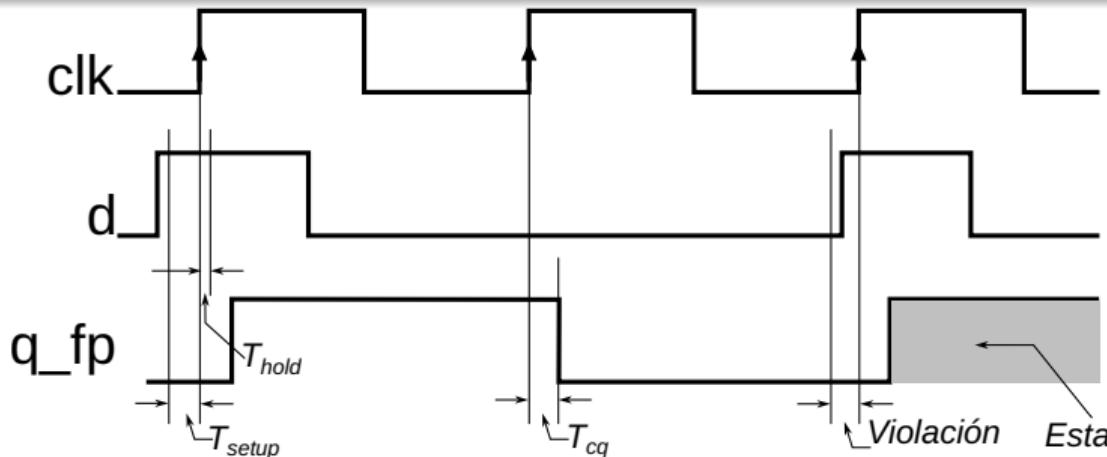


- tres parámetros principales de temporización:

**T<sub>cq</sub>:** Retardo de **clk** a **q**. Tiempo de propagación necesario para que la entrada **d** aparezca en la salida **q** después del flanco activo.

**T<sub>setup</sub>:** Tiempo de establecimiento. Intervalo de tiempo en el que la señal **d** debe ser estable antes del flanco activo de **clk**.

# Flip Flop D



- tres parámetros principales de temporización:

**Tcq:** Retardo de **clk** a **q**. Tiempo de propagación necesario para que la entrada **d** aparezca en la salida **q** después del flanco activo.

**Tsetup:** Tiempo de establecimiento. Intervalo de tiempo en el que la señal **d** debe ser estable antes del flanco activo de **clk**.

**Thold:** Tiempo de mantenimiento. Intervalo de tiempo en el que la señal **d** debe ser estable luego del flanco activo de **clk**.

# Flip Flop D

# Flip Flop D

- **T<sub>cq</sub>** corresponde aproximadamente al retardo de propagación de un componente combinacional.

# Flip Flop D

- ***Tcq*** corresponde aproximadamente al retardo de propagación de un componente combinacional.
- ***Tsetup* y *Thold***, son restricciones de tiempo. Especifican que la señal ***d*** debe ser estable en una pequeña ventana alrededor del flanco de muestreo del reloj.

# Flip Flop D

- **Tcq** corresponde aproximadamente al retardo de propagación de un componente combinacional.
- **Tsetup** y **Thold**, son restricciones de tiempo. Especifican que la señal **d** debe ser estable en una pequeña ventana alrededor del flanco de muestreo del reloj.
- Se genera una Violación del tiempo de establecimiento o Violación del tiempo de retención, cuando la señal en **d** cambia dentro de la ventana de tiempo dada por **Tsetup + Thold**.

# Flip Flop D

- **$T_{cq}$**  corresponde aproximadamente al retardo de propagación de un componente combinacional.
- **$T_{setup}$  y  $T_{hold}$** , son restricciones de tiempo. Especifican que la señal  **$d$**  debe ser estable en una pequeña ventana alrededor del flanco de muestreo del reloj.
- Se genera una Violación del tiempo de establecimiento o Violación del tiempo de retención, cuando la señal en  **$d$**  cambia dentro de la ventana de tiempo dada por  **$T_{setup} + T_{hold}$** .
- El FF puede tomar tres comportamientos:

# Flip Flop D

- **$Tcq$**  corresponde aproximadamente al retardo de propagación de un componente combinacional.
- **$Tsetup$  y  $Thold$** , son restricciones de tiempo. Especifican que la señal  **$d$**  debe ser estable en una pequeña ventana alrededor del flanco de muestreo del reloj.
- Se genera una Violación del tiempo de establecimiento o Violación del tiempo de retención, cuando la señal en  **$d$**  cambia dentro de la ventana de tiempo dada por  **$Tsetup + Thold$** .
- El FF puede tomar tres comportamientos:
  - La salida toma el resultado deseado y no causa ningún problema.

# Flip Flop D

- ***Tcq*** corresponde aproximadamente al retardo de propagación de un componente combinacional.
- ***Tsetup* y *Thold***, son restricciones de tiempo. Especifican que la señal ***d*** debe ser estable en una pequeña ventana alrededor del flanco de muestreo del reloj.
- Se genera una Violación del tiempo de establecimiento o Violación del tiempo de retención, cuando la señal en ***d*** cambia dentro de la ventana de tiempo dada por ***Tsetup + Thold***.
- El FF puede tomar tres comportamientos:
  - La salida toma el resultado deseado y no causa ningún problema.
  - La salida toma el estado inverso, lo cual implica que el FF acaba de muestrear el valor anterior. Si la entrada permanece sin cambios, el valor correcto se muestreará en el siguiente flanco ascendente. Dado que no hacemos suposiciones sobre el tiempo de llegada de la señal de entrada, no habrá ningún efecto negativo.

# Flip Flop D

- **$T_{cq}$**  corresponde aproximadamente al retardo de propagación de un componente combinacional.
- **$T_{setup}$  y  $T_{hold}$** , son restricciones de tiempo. Especifican que la señal  **$d$**  debe ser estable en una pequeña ventana alrededor del flanco de muestreo del reloj.
- Se genera una Violación del tiempo de establecimiento o Violación del tiempo de retención, cuando la señal en  **$d$**  cambia dentro de la ventana de tiempo dada por  **$T_{setup} + T_{hold}$** .
- El FF puede tomar tres comportamientos:
  - La salida toma el resultado deseado y no causa ningún problema.
  - La salida toma el estado inverso, lo cual implica que el FF acaba de muestrear el valor anterior. Si la entrada permanece sin cambios, el valor correcto se muestreará en el siguiente flanco ascendente. Dado que no hacemos suposiciones sobre el tiempo de llegada de la señal de entrada, no habrá ningún efecto negativo.
  - La salida toma un estado metaestable, en el que  **$q$**  no está ni en "0" ni en "1", sino en una tensión intermedia.

# Flip Flop D: Metaestabilidad

# Flip Flop D: Metaestabilidad

- Como su nombre lo indica, un estado metaestable no es realmente un estado estable.

# Flip Flop D: Metaestabilidad

- Como su nombre lo indica, un estado metaestable no es realmente un estado estable.
- Sin embargo si una salida metaestable se usa para controlar otras celdas lógicas, el valor intermedio puede propagarse aguas abajo y lleva a todo el sistema digital a un estado desconocido.

## Flip Flop D: Metaestabilidad

- Como su nombre lo indica, un estado metaestable no es realmente un estado estable.
- Sin embargo si una salida metaestable se usa para controlar otras celdas lógicas, el valor intermedio puede propagarse aguas abajo y lleva a todo el sistema digital a un estado desconocido.
- Su propia condición, sin embargo, hace que un pequeño ruido o perturbación compense su “equilibrio” y fuerce al FF a entrar en uno de los estados estables.

## Flip Flop D: Metaestabilidad

- Como su nombre lo indica, un estado metaestable no es realmente un estado estable.
- Sin embargo si una salida metaestable se usa para controlar otras celdas lógicas, el valor intermedio puede propagarse aguas abajo y lleva a todo el sistema digital a un estado desconocido.
- Su propia condición, sin embargo, hace que un pequeño ruido o perturbación compense su “equilibrio” y fuerce al FF a entrar en uno de los estados estables.
- Finalmente, luego de un tiempo  $T_r$ , el FF convergerá a un estado estable.

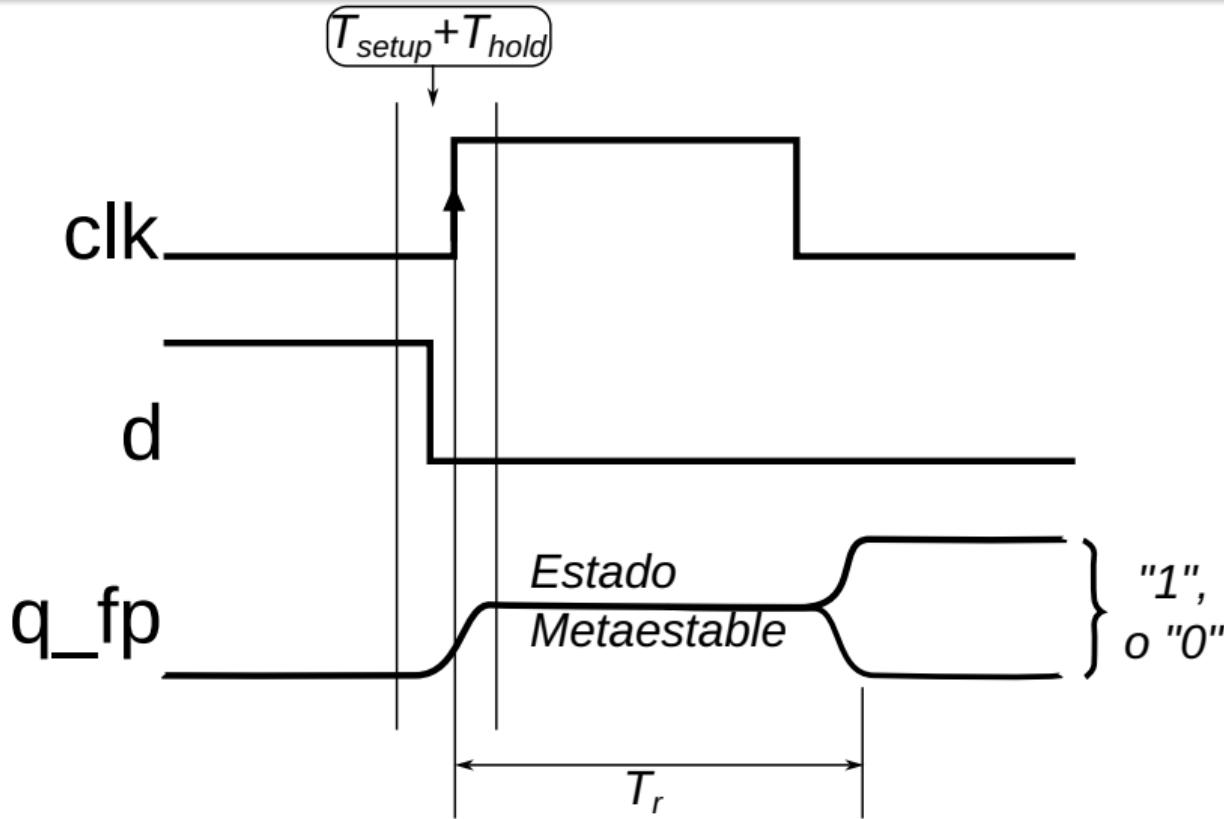
## Flip Flop D: Metaestabilidad

- Como su nombre lo indica, un estado metaestable no es realmente un estado estable.
- Sin embargo si una salida metaestable se usa para controlar otras celdas lógicas, el valor intermedio puede propagarse aguas abajo y lleva a todo el sistema digital a un estado desconocido.
- Su propia condición, sin embargo, hace que un pequeño ruido o perturbación compense su “equilibrio” y fuerce al FF a entrar en uno de los estados estables.
- Finalmente, luego de un tiempo  $T_r$ , el FF convergerá a un estado estable.
- Estudios teóricos muestran que un dispositivo biestable siempre tiene un estado metaestable, y este fenómeno es inevitable.

## Flip Flop D: Metaestabilidad

- Como su nombre lo indica, un estado metaestable no es realmente un estado estable.
- Sin embargo si una salida metaestable se usa para controlar otras celdas lógicas, el valor intermedio puede propagarse aguas abajo y lleva a todo el sistema digital a un estado desconocido.
- Su propia condición, sin embargo, hace que un pequeño ruido o perturbación compense su “equilibrio” y fuerce al FF a entrar en uno de los estados estables.
- Finalmente, luego de un tiempo  $T_r$ , el FF convergerá a un estado estable.
- Estudios teóricos muestran que un dispositivo biestable siempre tiene un estado metaestable, y este fenómeno es inevitable.
- La única solución es proporcionar suficiente tiempo para que el dispositivo resuelva la situación y alcance un estado estable.

# Flip Flop D: Metaestabilidad



# Flip Flop D: Metaestabilidad

# Flip Flop D: Metaestabilidad

- El tiempo de resolución, lamentablemente, no es determinista. Se caracteriza por una función de distribución de probabilidad:

$$P(T_r) = e^{-\frac{T_r}{\tau}}$$

## Flip Flop D: Metaestabilidad

- El tiempo de resolución, lamentablemente, no es determinista. Se caracteriza por una función de distribución de probabilidad:

$$P(T_r) = e^{-\frac{T_r}{\tau}}$$

- En esta ecuación,  $\tau$  es la constante de tiempo de decaimiento y está determinada por las características eléctricas del FF.

## Flip Flop D: Metaestabilidad

- El tiempo de resolución, lamentablemente, no es determinista. Se caracteriza por una función de distribución de probabilidad:

$$P(T_r) = e^{-\frac{T_r}{\tau}}$$

- En esta ecuación,  $\tau$  es la constante de tiempo de decaimiento y está determinada por las características eléctricas del FF.
- Un valor típico de la tecnología de dispositivos actual es de alrededor de una fracción de nanosegundo.

## Flip Flop D: Metaestabilidad

- El tiempo de resolución, lamentablemente, no es determinista. Se caracteriza por una función de distribución de probabilidad:

$$P(T_r) = e^{-\frac{T_r}{\tau}}$$

- En esta ecuación,  $\tau$  es la constante de tiempo de decaimiento y está determinada por las características eléctricas del FF.
- Un valor típico de la tecnología de dispositivos actual es de alrededor de una fracción de nanosegundo.
- La ecuación indica la probabilidad de que la condición de metaestabilidad persista más allá de  $T_r$ , después del flanco de **clk**.

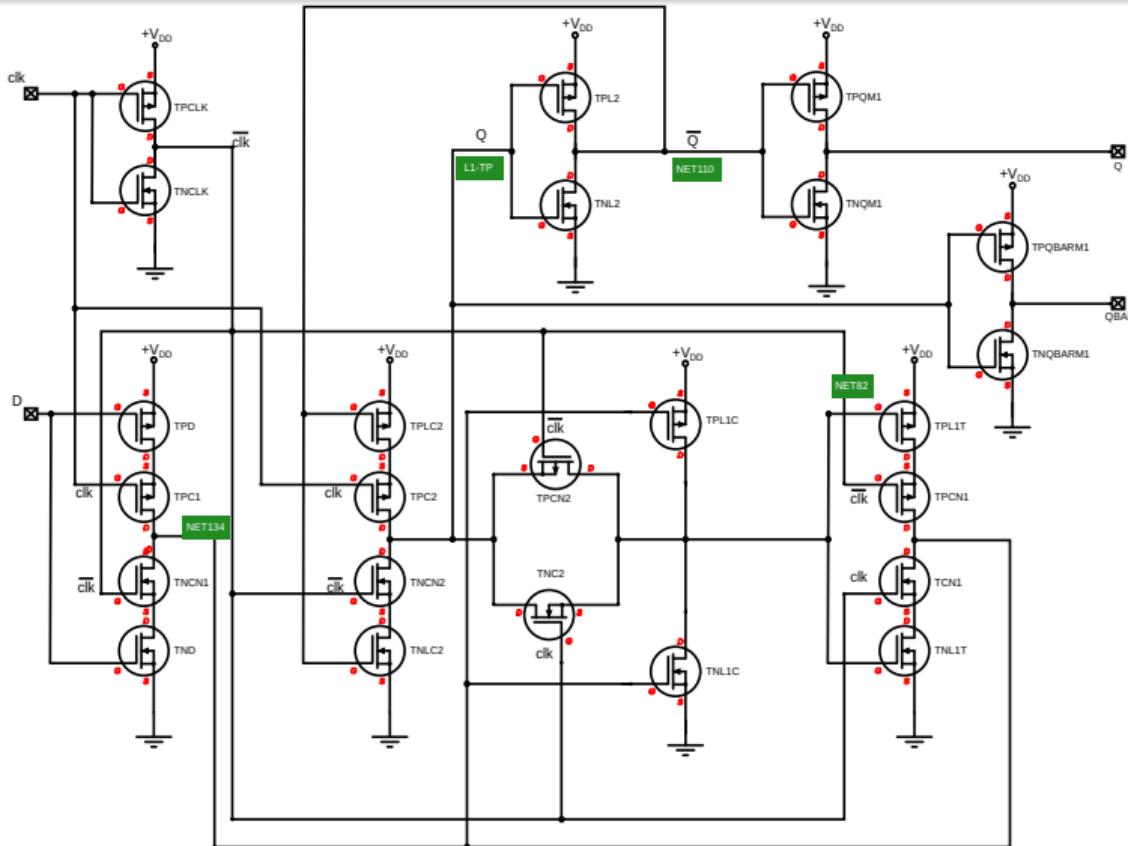
## Flip Flop D: Metaestabilidad

- El tiempo de resolución, lamentablemente, no es determinista. Se caracteriza por una función de distribución de probabilidad:

$$P(T_r) = e^{-\frac{T_r}{\tau}}$$

- En esta ecuación,  $\tau$  es la constante de tiempo de decaimiento y está determinada por las características eléctricas del FF.
- Un valor típico de la tecnología de dispositivos actual es de alrededor de una fracción de nanosegundo.
- La ecuación indica la probabilidad de que la condición de metaestabilidad persista más allá de  $T_r$ , después del flanco de **clk**.
- Puede interpretarse como la probabilidad de que la metaestabilidad no se resuelva en  $T_r$  segundos.

# Flip Flop D: Circuito típico



# Cache Directory

- Una SRAM de tamaño relativamente pequeño, como puede ser un cache (al menos el L1) se implementa con un Register File.

# Cache Directory

- Una SRAM de tamaño relativamente pequeño, como puede ser un cache (al menos el L1) se implementa con un Register File.
- Un Register File es un array de SRAM multiport.

# Cache Directory

- Una SRAM de tamaño relativamente pequeño, como puede ser un cache (al menos el L1) se implementa con un Register File.
- Un Register File es un array de SRAM multiport.
- Se compone de un conjunto de registros, y lógica combinacional destinada a minimizar el cableado y señales de E/S necesarias para acceder a una palabra de almacenamiento.

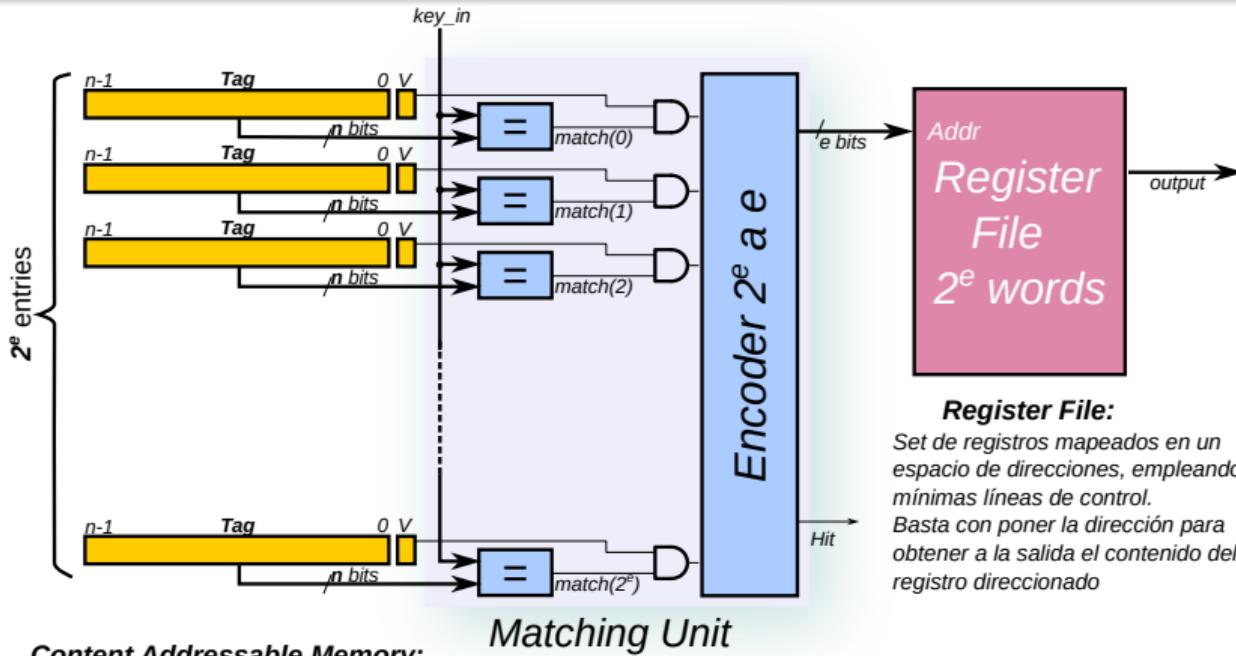
# Cache Directory

- Una SRAM de tamaño relativamente pequeño, como puede ser un cache (al menos el L1) se implementa con un Register File.
- Un Register File es un array de SRAM multiport.
- Se compone de un conjunto de registros, y lógica combinacional destinada a minimizar el cableado y señales de E/S necesarias para acceder a una palabra de almacenamiento.
- Si bien son mas transistores por bit, puede accederse en uno o dos ciclos de reloj. Y eso es lo que se busca en dispositivos que van a conformar la jerarquía de memoria mas próxima a la CPU.

# Cache Directory

- Una SRAM de tamaño relativamente pequeño, como puede ser un cache (al menos el L1) se implementa con un Register File.
- Un Register File es un array de SRAM multiport.
- Se compone de un conjunto de registros, y lógica combinacional destinada a minimizar el cableado y señales de E/S necesarias para acceder a una palabra de almacenamiento.
- Si bien son mas transistores por bit, puede accederse en uno o dos ciclos de reloj. Y eso es lo que se busca en dispositivos que van a conformar la jerarquía de memoria mas próxima a la CPU.
- El Cache Directory permite convertir el Register File en una Content Addressable Memory.

# Content Addressable Memory



## Register File:

Set de registros mapeados en un espacio de direcciones, empleando mínimas líneas de control.

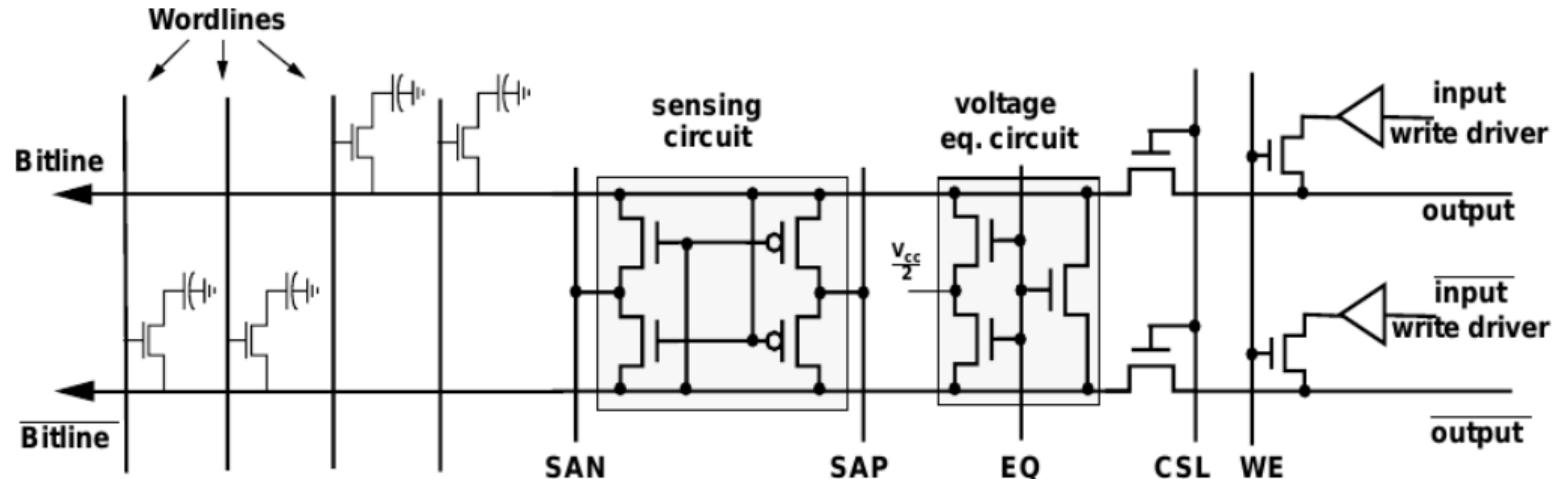
Basta con poner la dirección para obtener a la salida el contenido del registro direccionado

# Temario

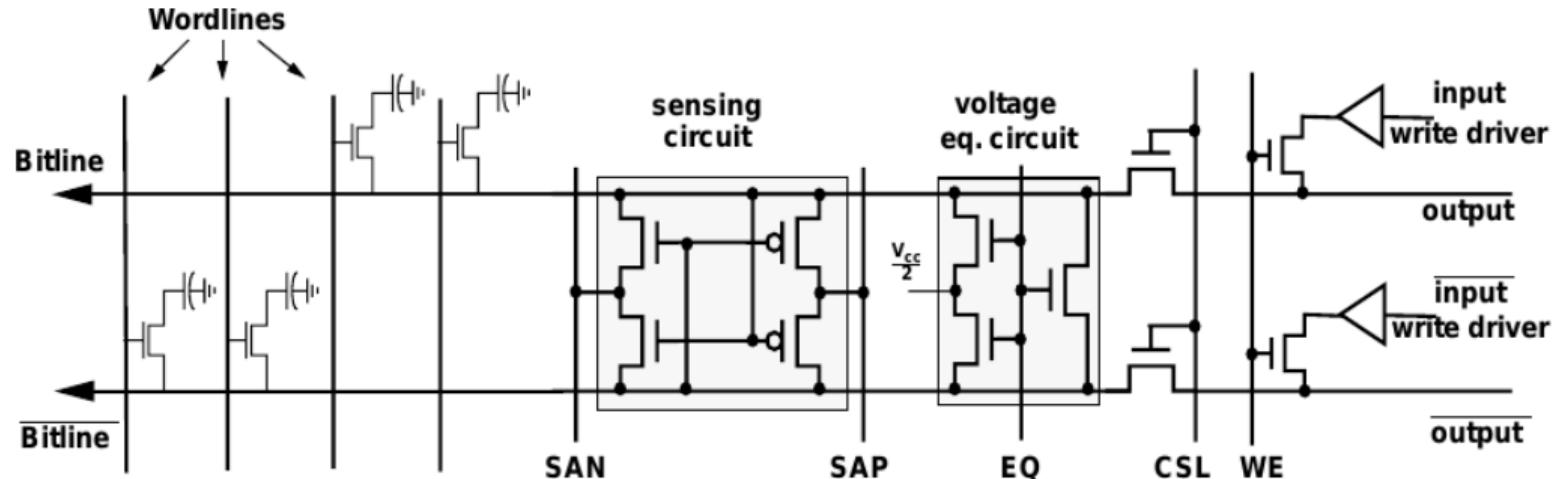
- 1 El sistema de Memoria
  - Antecedentes
  - Rol de la memoria en un computador
  - Jerarquía de memoria
- 2 Tecnologías de Memoria
  - Clasificación
  - Memorias No volátiles
  - Memorias Volátiles
  - Memorias y velocidad del Procesador
- 3 Memoria Cache
  - Principio de Funcionamiento
  - Métricas y Performance
  - Hardware dedicado = + complejidad
  - organización de un cache
- 4 Cache en Sistemas Multiprocesador

- 5 Organización de Sistemas Multiprocesador
  - Coherencia de un cache
  - Protocolos de Coherencia para Multicore
  - Casos del Mundo real
- 6 Memorias Dinámicas
  - Introducción
  - Organización interna
- 7 Standards
  - Estado del arte
  - JEDEC SDRAM
- 8 Controladores de Memoria
  - Introducción General
  - Arquitectura
- 9 Configuración
  - Configuración del DRAM Device
- 10 Entrada Salida de Datos
  - Integración con el sistema Cache
  - Evitar cuellos de botella es la clave
- 11 Casos Prácticos
  - Beagle Bone Black
  - Memorias DDR en la BBB
  - Controlador de DDRn SDRAM en la BBB
- 12 SRAM Cuestiones de Implementación
  - Vistazo introductorio
  - Decodificación
  - Implementación
- 13 DRAM Detalles de implementación
  - Acceso a las celdas
  - JEDEC DDR SDRAM
  - Protocolo de acceso

# Amplificador de Detección

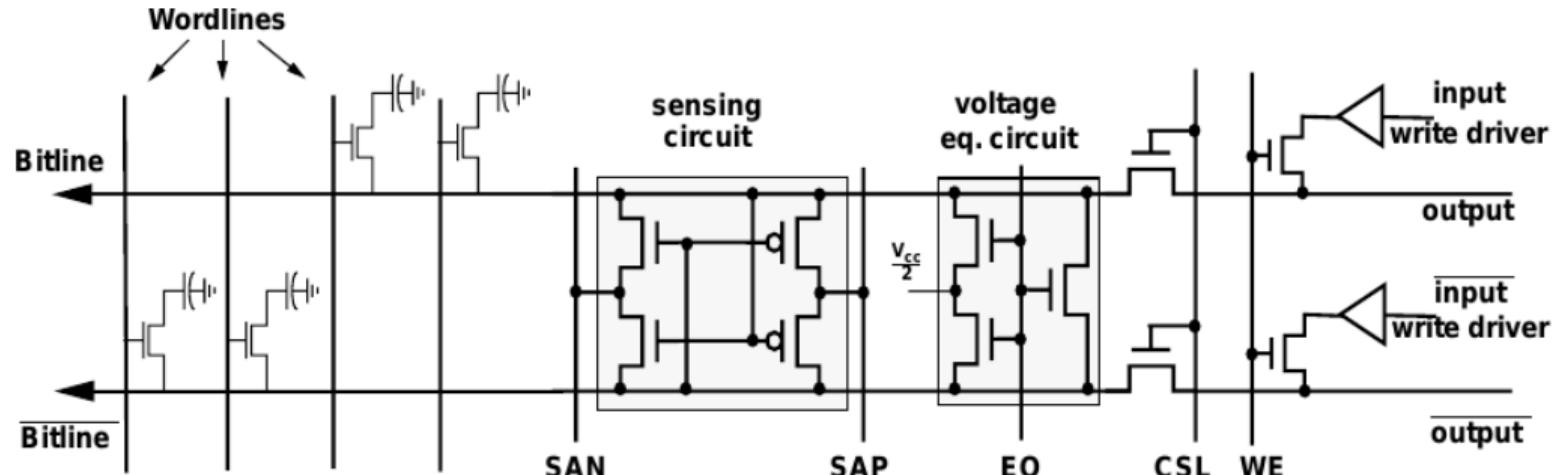


# Amplificador de Detección



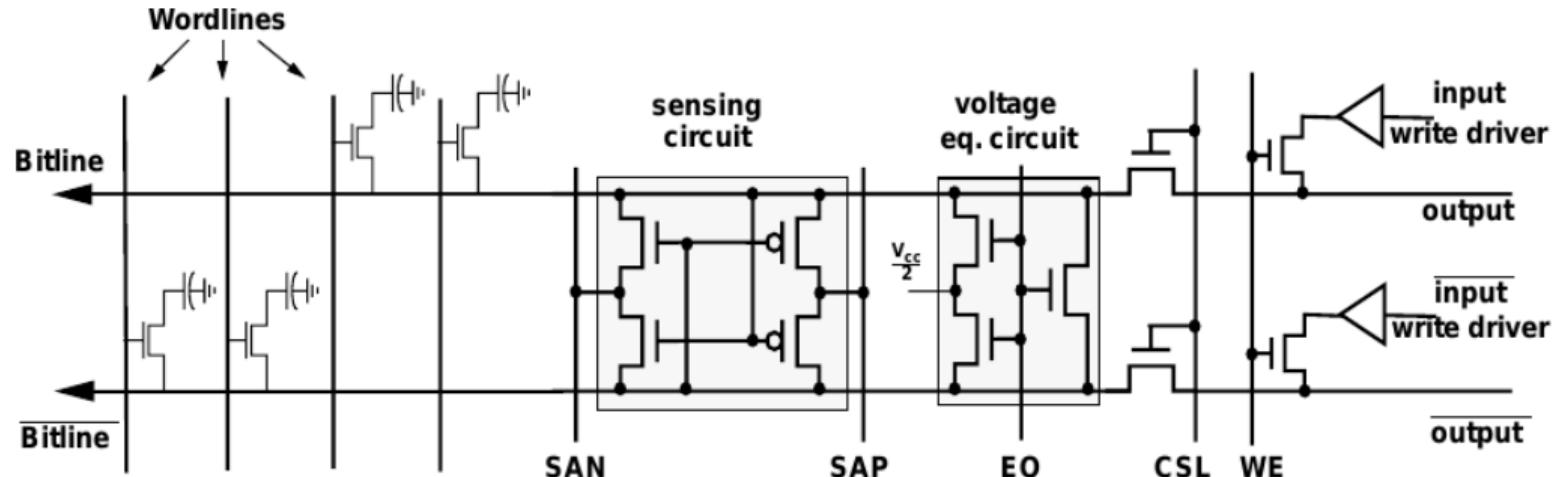
- La figura muestra un amplificador de detección básico.

# Amplificador de Detección



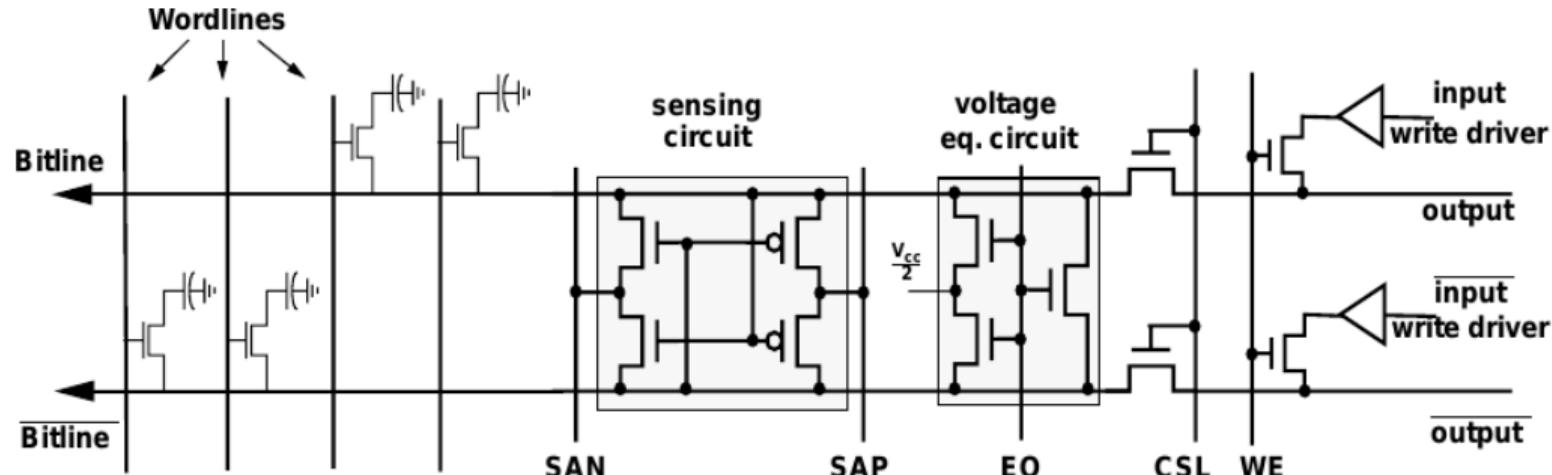
- La figura muestra un amplificador de detección básico.
- Los dispositivos DRAM modernos contienen estos elementos básicos, mas otros adicionales que contribuyen con la aislación entre arrays, equilibrio estructural del amplificador, y aceleración de la capacidad de detección.

# Amplificador de Detección



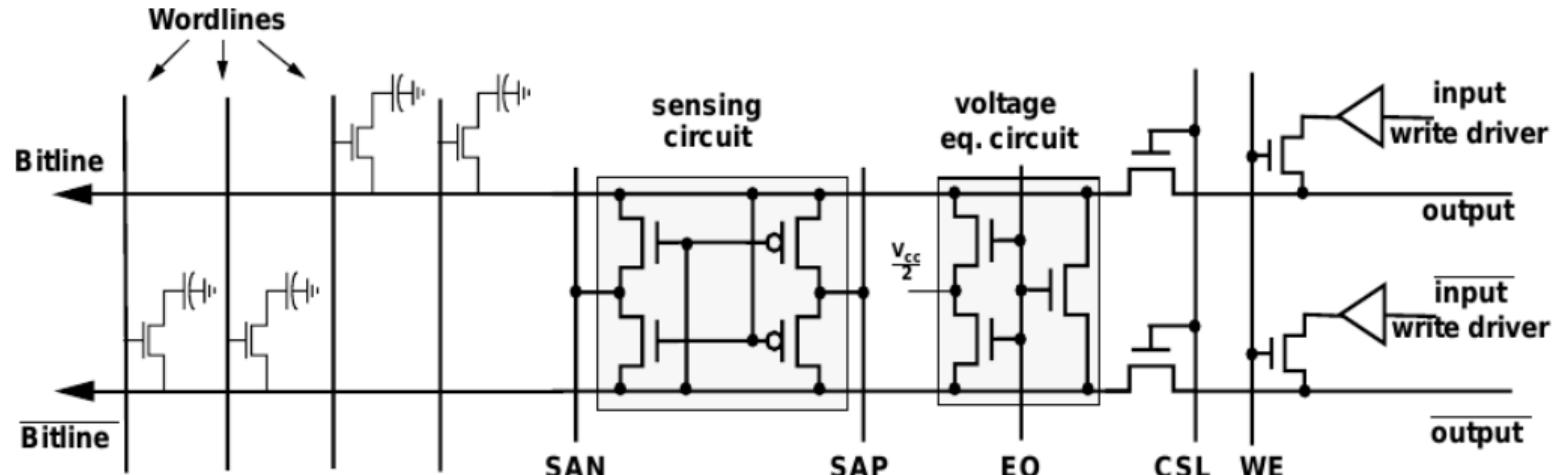
- La línea de señal de ecualización (EQ) controla el circuito de ecualización de tensión.

# Amplificador de Detección



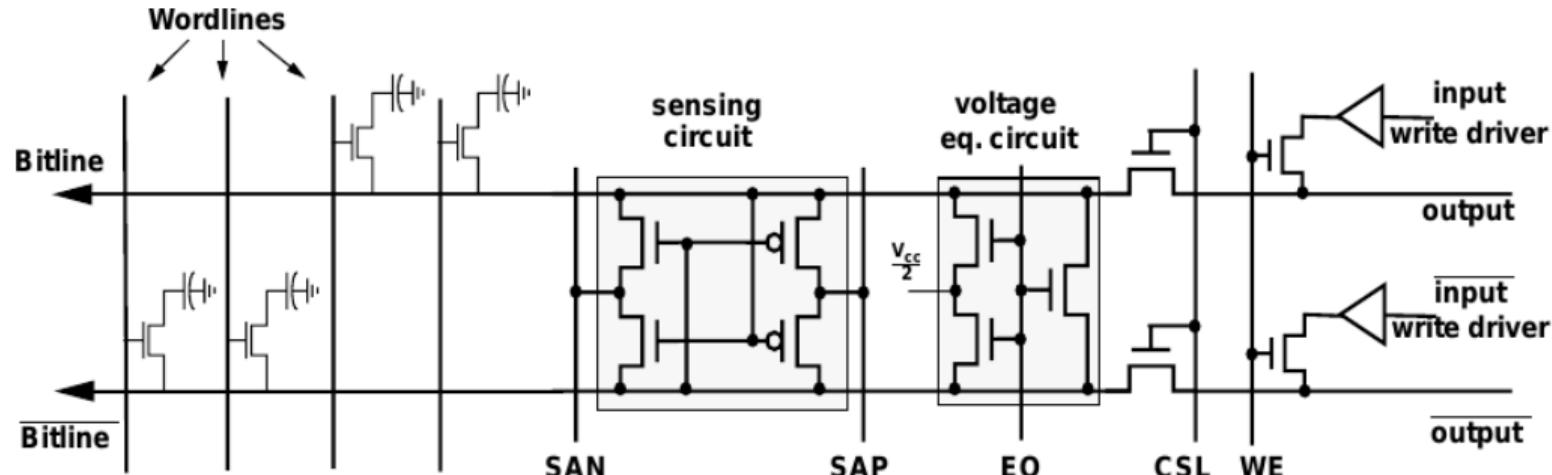
- La línea de señal de ecualización (EQ) controla el circuito de ecualización de tensión.
- Garantiza que la tensión en los pares de Bit-lines coincidan lo más posible.

# Amplificador de Detección



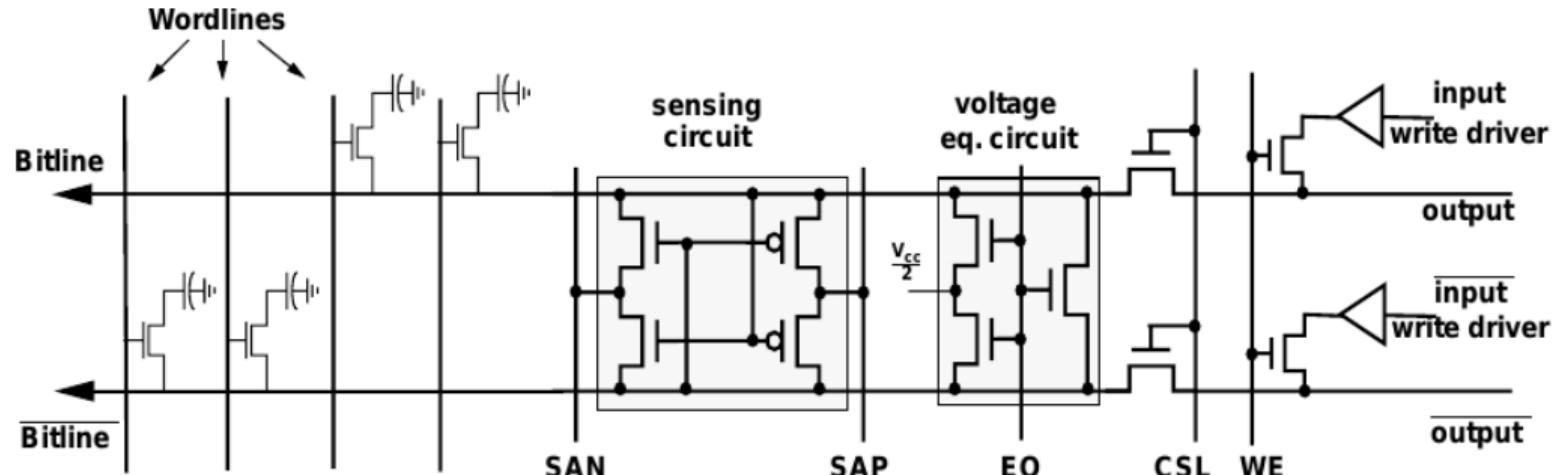
- La línea de señal de ecualización (EQ) controla el circuito de ecualización de tensión.
- Garantiza que la tensión en los pares de Bit-lines coincidan lo más posible.
- Dado que el amplificador de detección diferencial está diseñado para amplificar el diferencial de tensión entre un par de Bit-lines, cualquier desequilibrio de tensión entre estas Bit-lines antes de la activación de los transistores de acceso, degradaría su eficacia.

# Amplificador de Detección



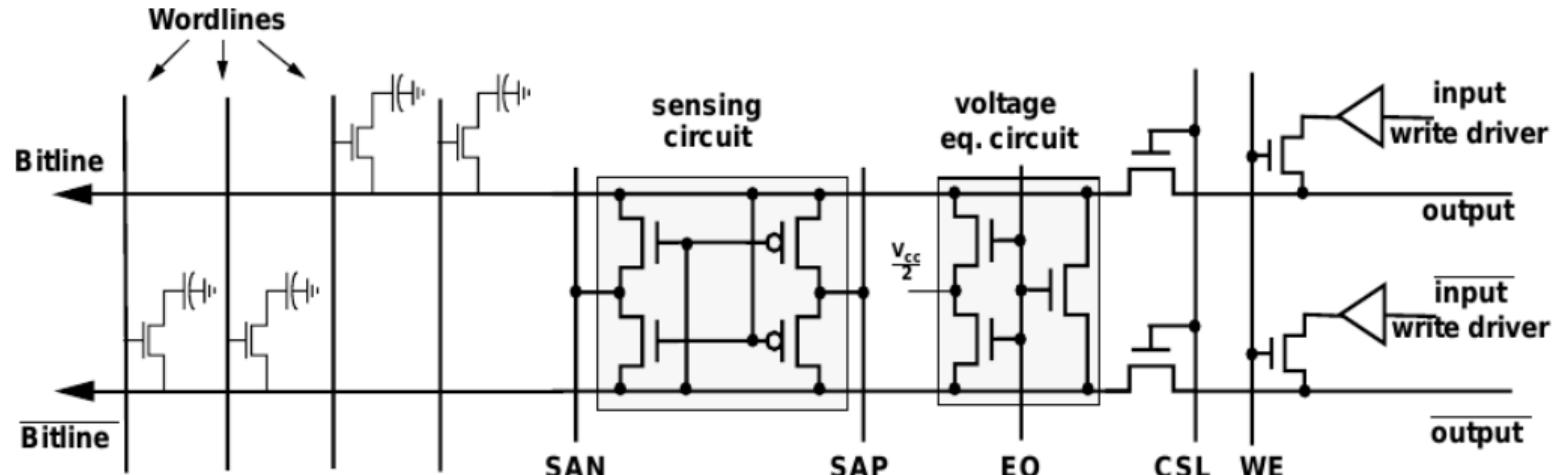
- El corazón del amplificador de detección es el conjunto de cuatro transistores interconectados, etiquetados como el circuito de detección (sensing circuit).

# Amplificador de Detección



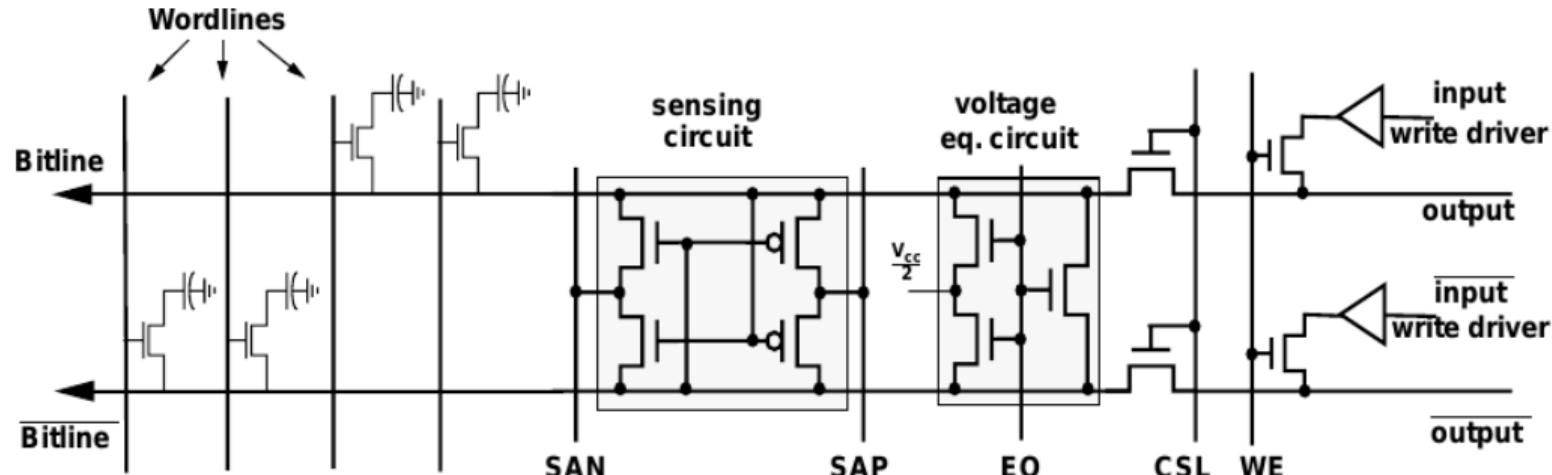
- El corazón del amplificador de detección es el conjunto de cuatro transistores interconectados, etiquetados como el circuito de detección (sensing circuit).
- Es esencialmente un biestable diseñado para conducir los pares de Bit-lines a los extremos de tensión complementarios, dependiendo de los valores de tensión en las Bit-lines al momento en que se activan las señales de detección **SAN** (Sense-Amplifier N-Fet Control) y **SAP** (Sense-Amplifier P-Fet Control).

# Amplificador de Detección



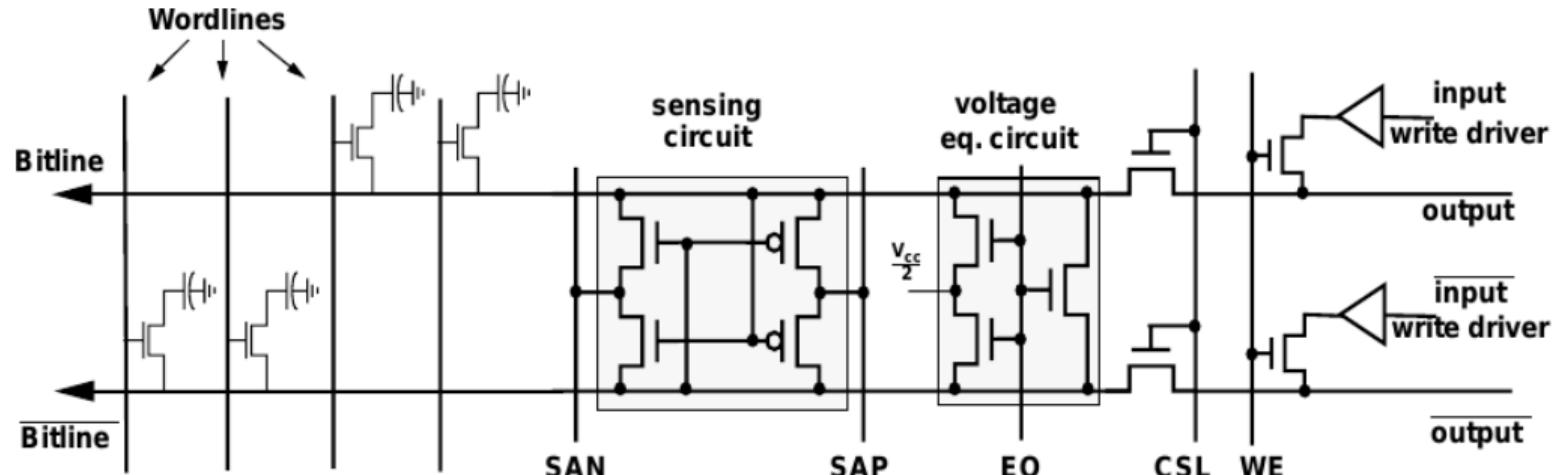
- **SAN** controla la activación de los NFets y **SAP** controla la activación de los PFets en el circuito de detección.

# Amplificador de Detección



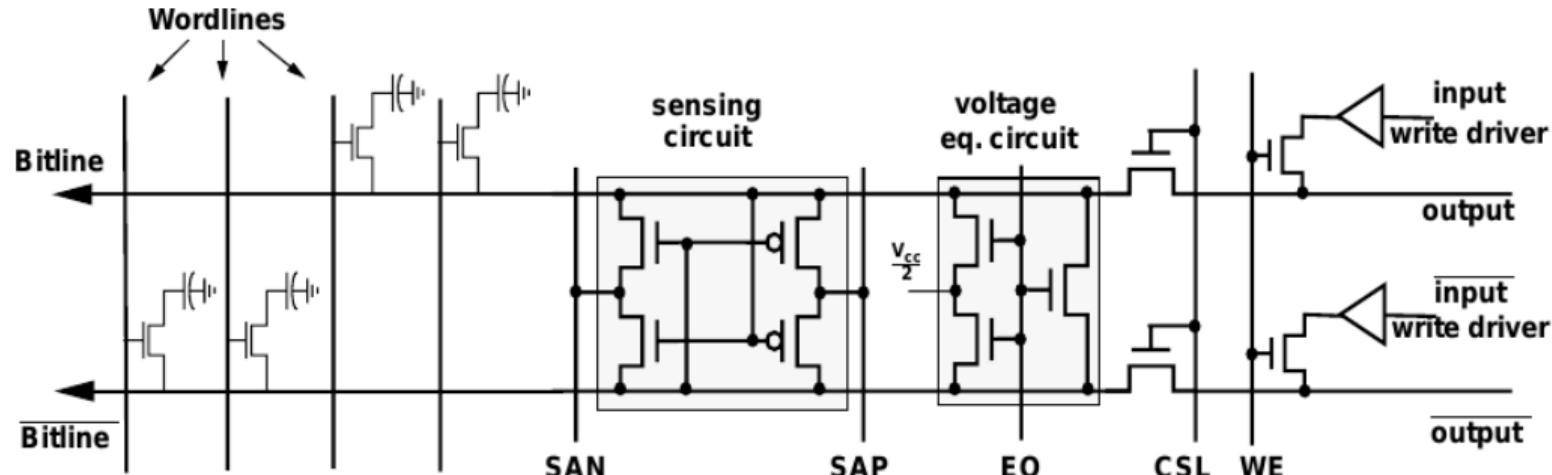
- **SAN** controla la activación de los NFets y **SAP** controla la activación de los PFets en el circuito de detección.
- Una vez aseguradas **SAN** y **SAP**, las Bit-lines se conducen a los niveles de tensión complementarios.

# Amplificador de Detección



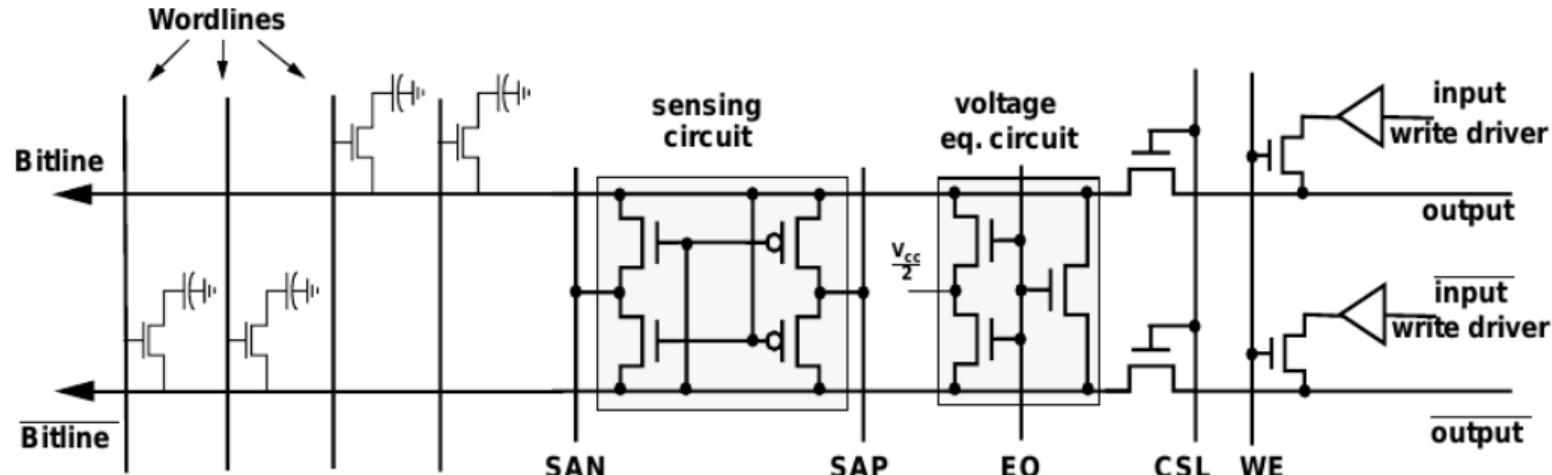
- **SAN** controla la activación de los NFets y **SAP** controla la activación de los PFets en el circuito de detección.
- Una vez aseguradas **SAN** y **SAP**, las Bit-lines se conducen a los niveles de tensión complementarios.
- La línea de selección de columna (**CSL**) activa los transistores de salida que ponen la tensión en la salida para ser leída desde el exterior de DRAM device.

# Amplificador de Detección



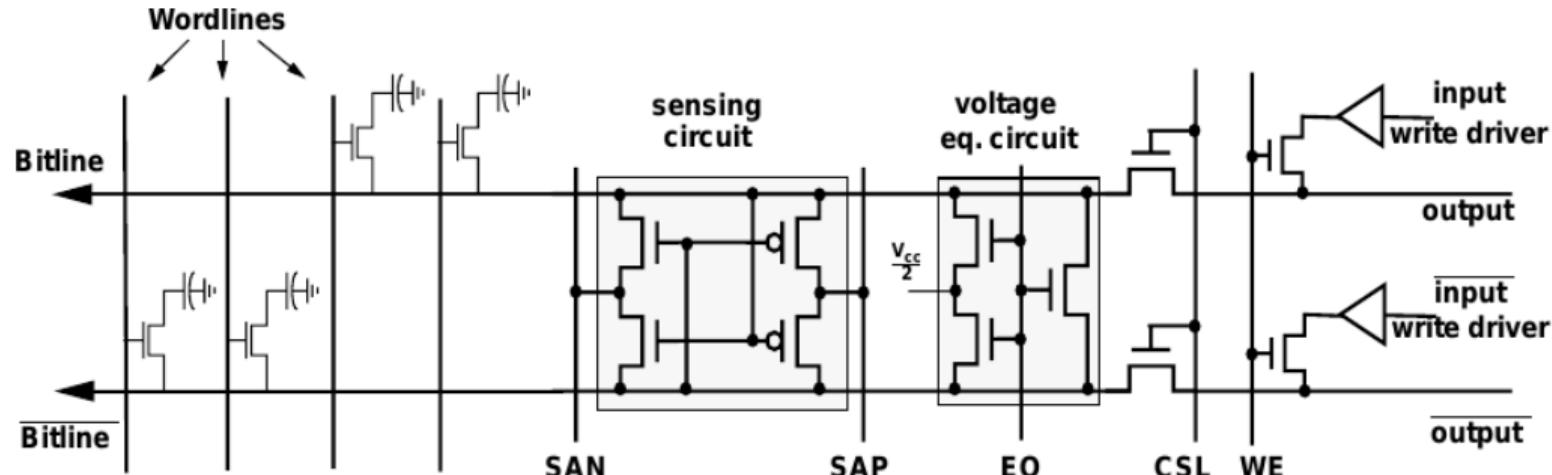
- Al mismo tiempo, el transistor de acceso de celda a la que se accede permanece abierto y la tensión en la Bit-line recarga el capacitor de almacenamiento.

# Amplificador de Detección



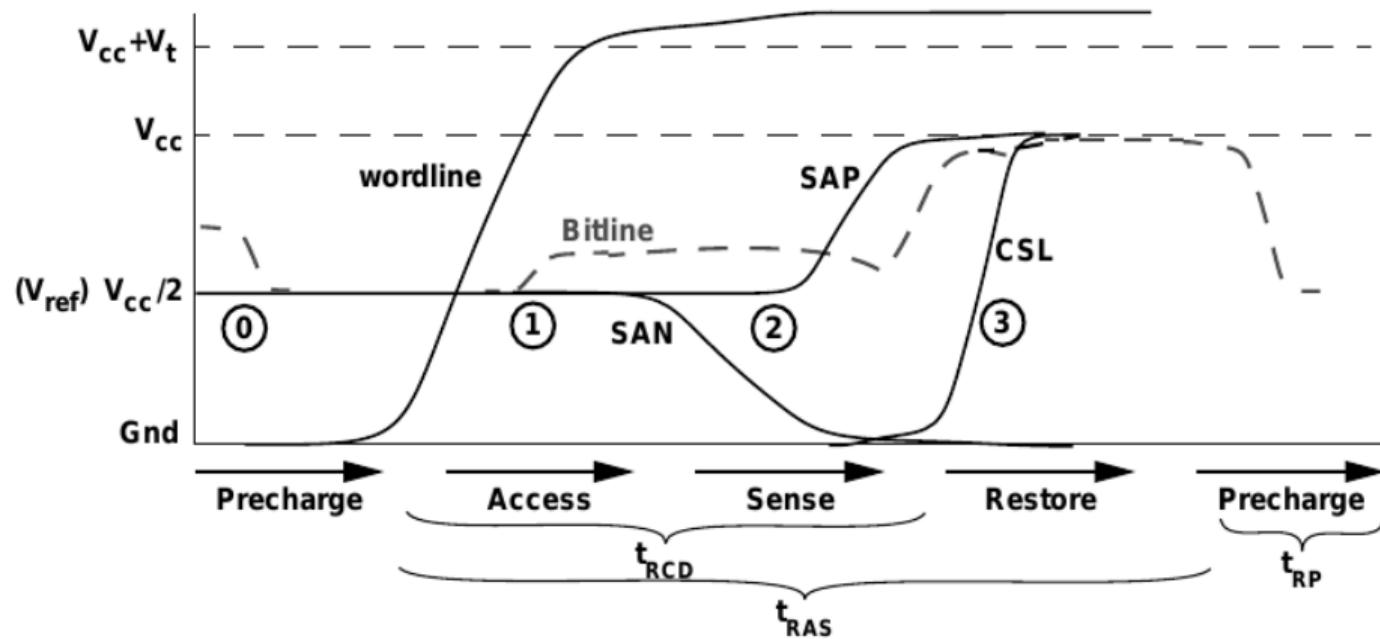
- Al mismo tiempo, el transistor de acceso de celda a la que se accede permanece abierto y la tensión en la Bit-line recarga el capacitor de almacenamiento.
- En el caso de una operación de escritura, la líneas CSL y de habilitación de escritura (WE) permiten que los controladores de escritura de entrada proporcionen una corriente suficiente para sobrecargar al amplificador de detección y la tensión de la Bit-line.

# Amplificador de Detección

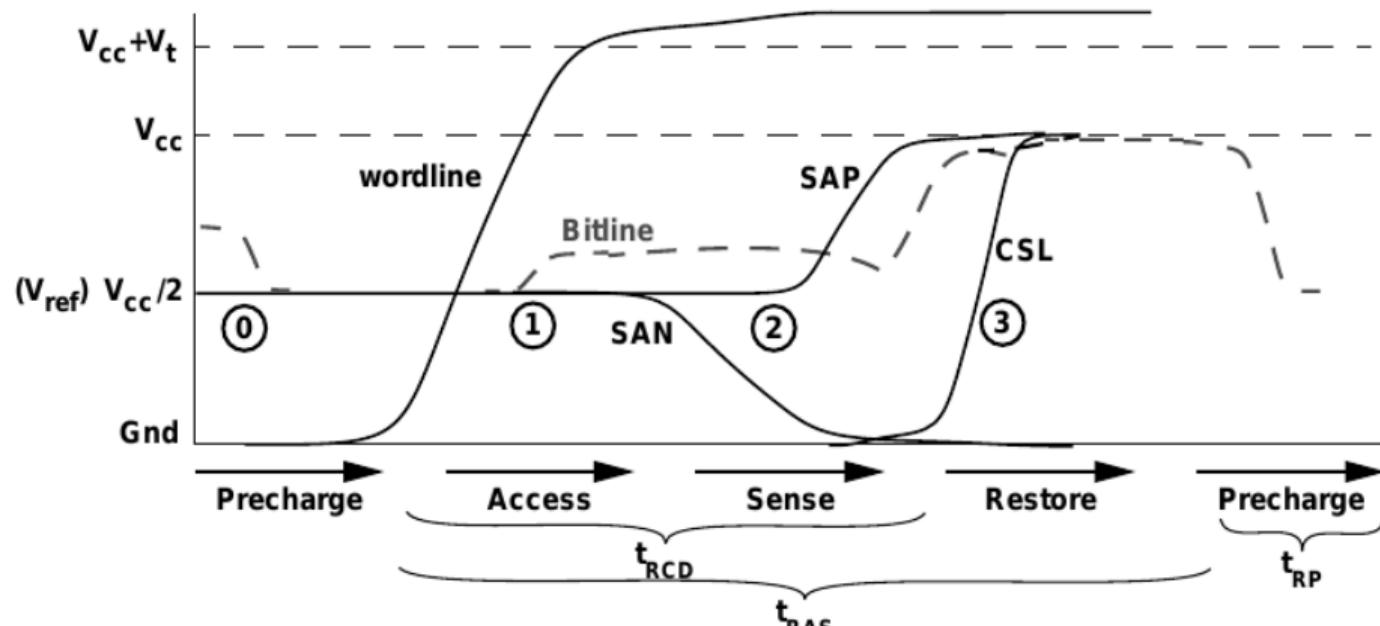


- Una vez que el amplificador de detección se sobrecargue al nuevo valor, lo mantendrá y lo conducirá a la celda DRAM a través del transistor de acceso aún abierto.

# Operación del amplificador de detección

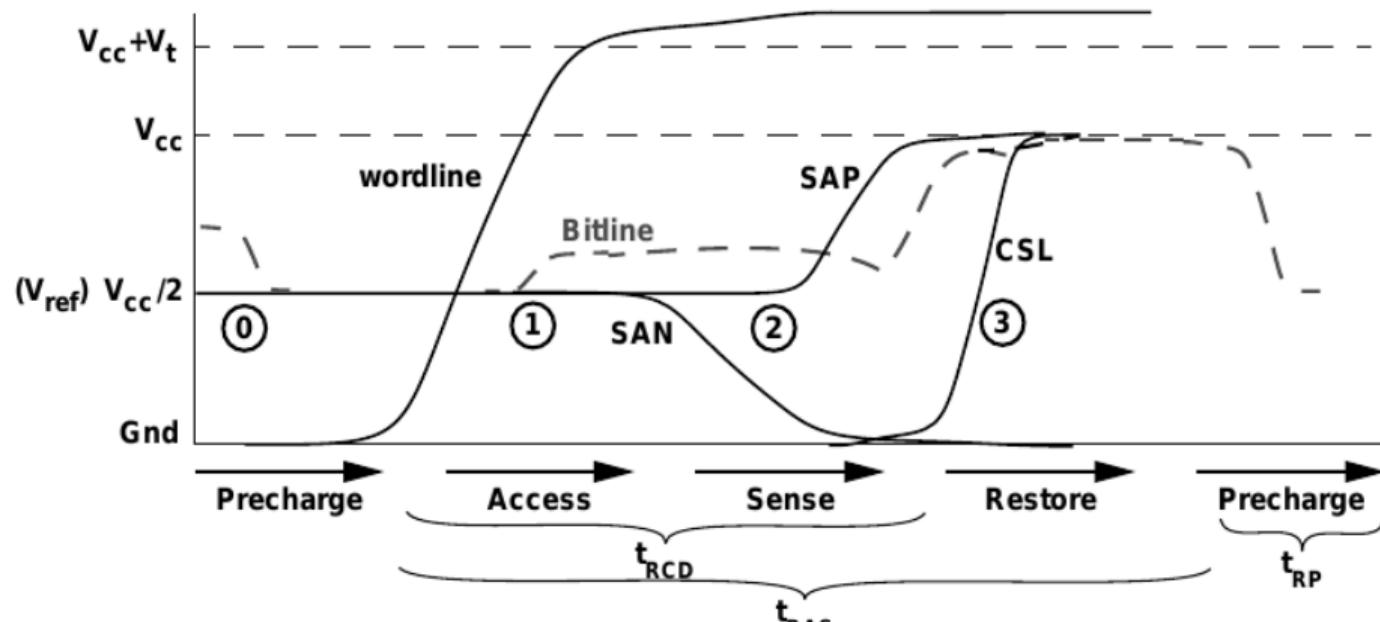


# Operación del amplificador de detección



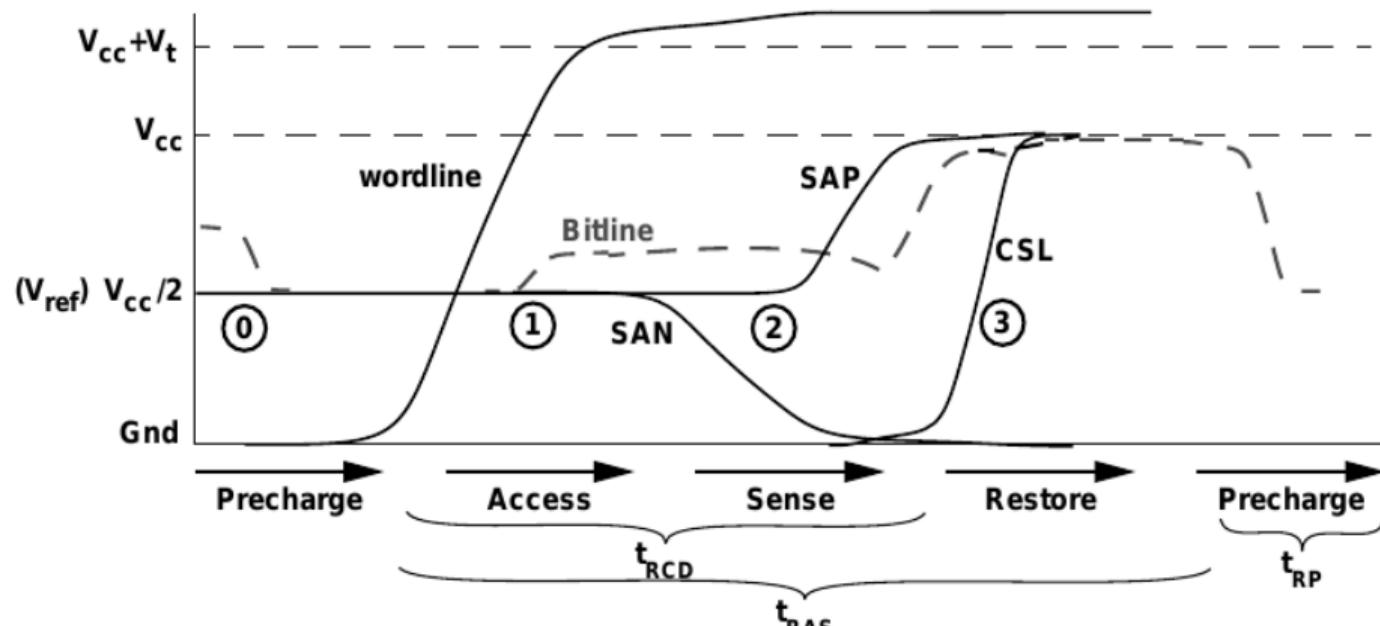
- La tensión máxima que se puede colocar a través del transistor de acceso es  $V_{gs} - V_t$ .

# Operación del amplificador de detección



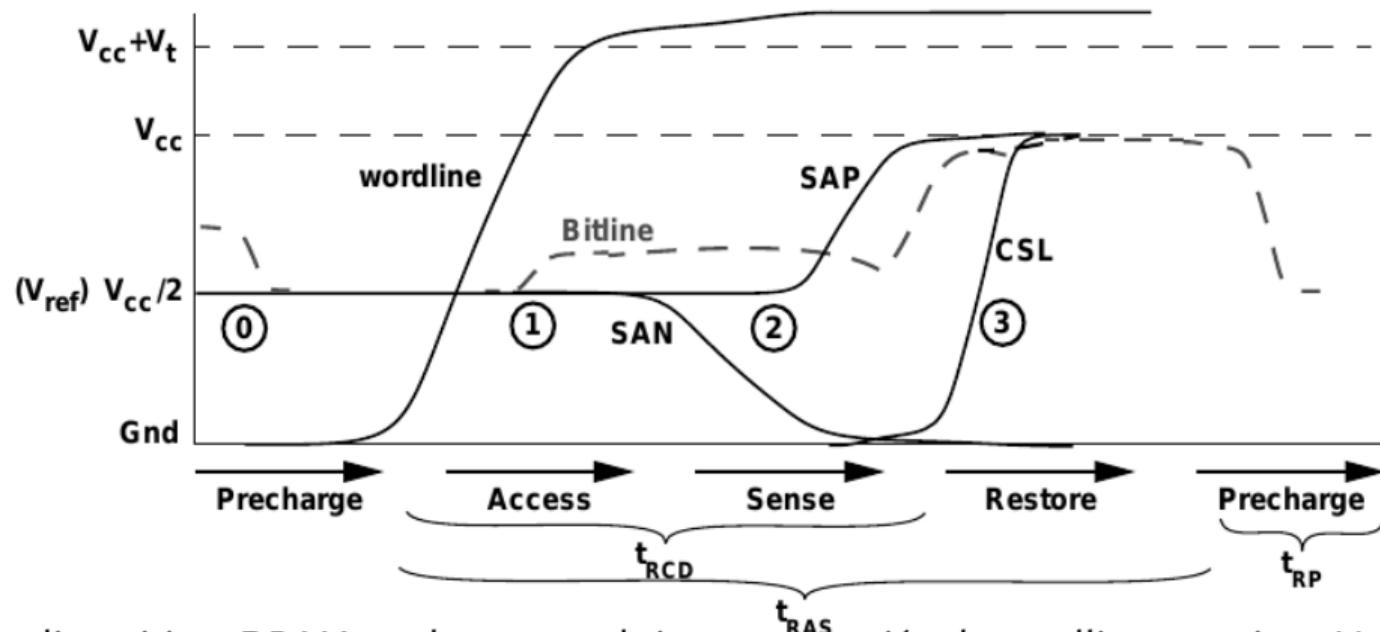
- $V_t$  es la tensión de umbral del transistor de acceso y  $V_{gs}$  es la tensión Gate Source del transistor de acceso.

# Operación del amplificador de detección



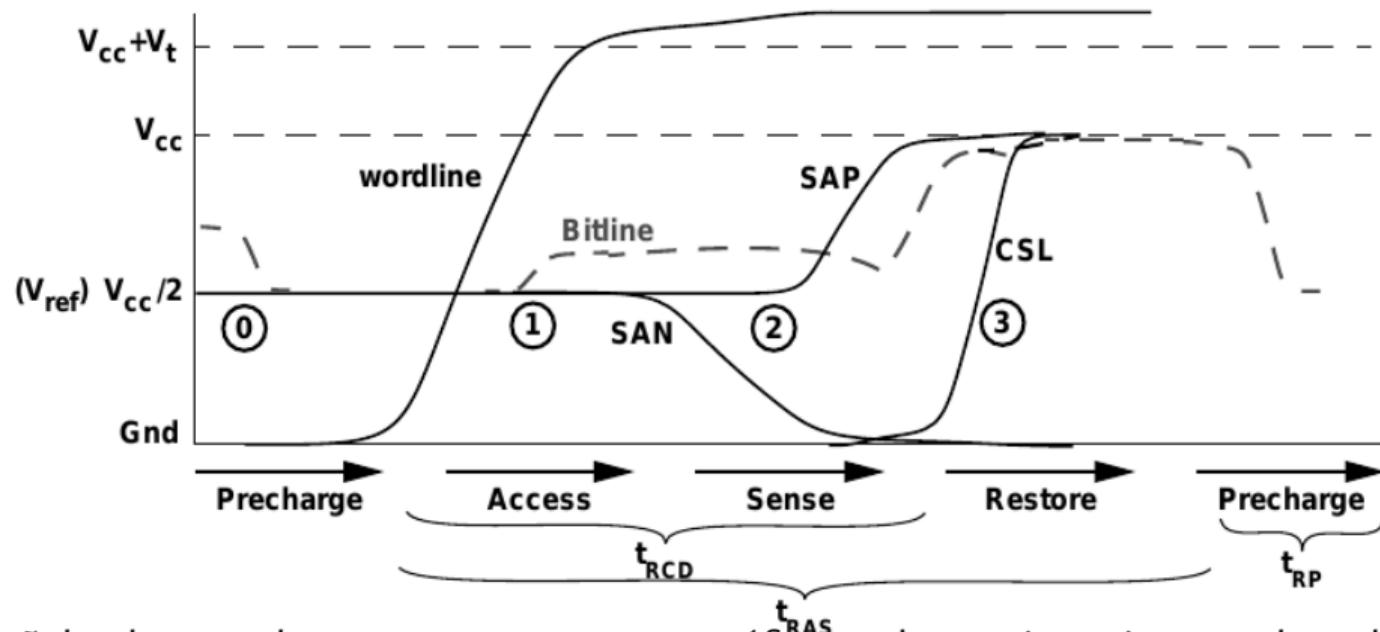
- Al sobrecargar la tensión de la wordline a  $V_{gs} - V_t$ , el capacitor de almacenamiento puede cargarse a  $V_{cc}$  máxima mediante el amplificador de detección en la fase Restore de la operación de detección.

# Operación del amplificador de detección



- En los dispositivos DRAM modernos, se obtiene una tensión de wordline superior a  $V_{cc}$  mediante un circuito de cambio de nivel por *voltage pumping*. Vale la pena analizarlo en un curso específico de diseño de DRAM Devices.

# Operación del amplificador de detección



- Las señales de control que se muestran en este gráfico son las que intervienen en los subsiguientes diagramas de operación para c/u de las cuatro fases del eje horizontal.

# Operación del amplificador de detección

# Operación del amplificador de detección

- En los siguientes slides, se muestra las cuatro fases diferentes en la operación de detección de un amplificador de detección diferencial.

# Operación del amplificador de detección

- En los siguientes slides, se muestra las cuatro fases diferentes en la operación de detección de un amplificador de detección diferencial.
- Las operaciones de ***precarga, acceso, detección y restauración*** de un amplificador de detección están etiquetadas como fases cero, uno, dos y tres, respectivamente.

# Operación del amplificador de detección

- En los siguientes slides, se muestra las cuatro fases diferentes en la operación de detección de un amplificador de detección diferencial.
- Las operaciones de ***precarga, acceso, detección y restauración*** de un amplificador de detección están etiquetadas como fases cero, uno, dos y tres, respectivamente.
- La fase de ***precarga*** normalmente se considera como una operación separada de las fases de una operación de acceso a Filas, por eso se etiqueta como fase cero.

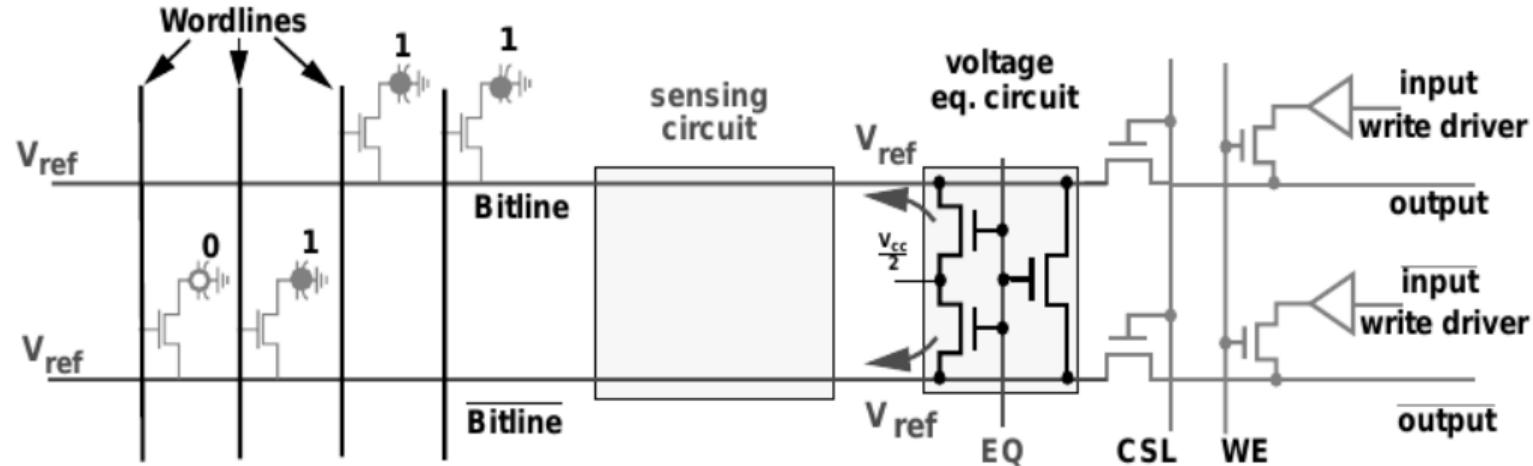
# Operación del amplificador de detección

- En los siguientes slides, se muestra las cuatro fases diferentes en la operación de detección de un amplificador de detección diferencial.
- Las operaciones de ***precarga***, ***acceso***, ***detección*** y ***restauración*** de un amplificador de detección están etiquetadas como fases cero, uno, dos y tres, respectivamente.
- La fase de ***precarga*** normalmente se considera como una operación separada de las fases de una operación de acceso a Filas, por eso se etiqueta como fase cero.
- La fase de ***precarga*** es un requisito previo para una operación de acceso a Filas, pero se realiza por separado de la operación de acceso a Filas en sí.

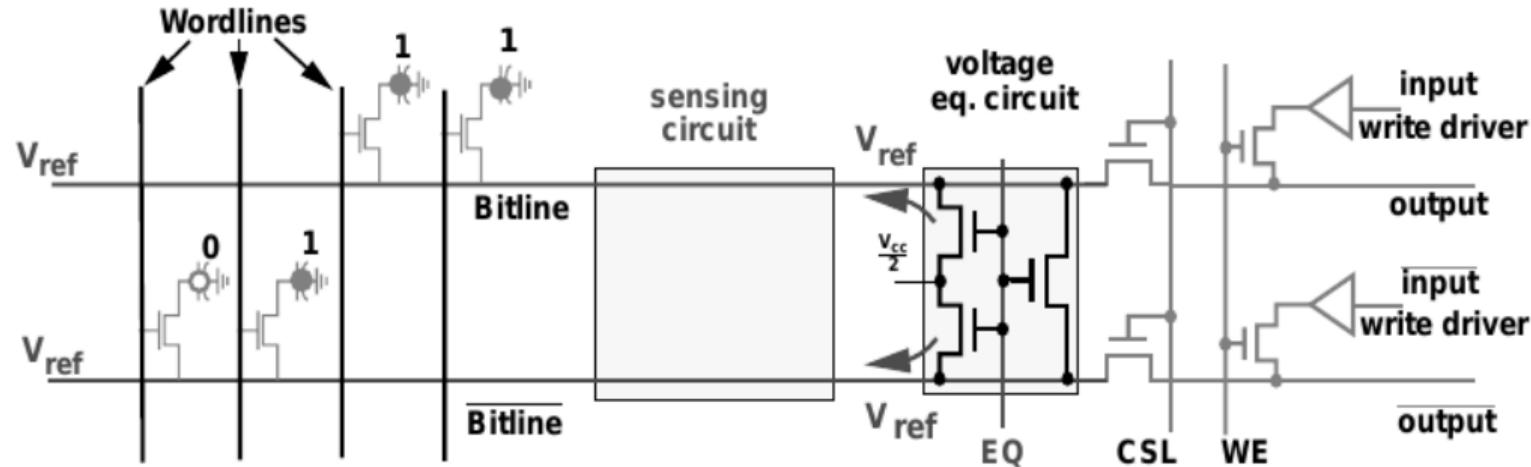
# Operación del amplificador de detección

- En los siguientes slides, se muestra las cuatro fases diferentes en la operación de detección de un amplificador de detección diferencial.
- Las operaciones de ***precarga, acceso, detección y restauración*** de un amplificador de detección están etiquetadas como fases cero, uno, dos y tres, respectivamente.
- La fase de ***precarga*** normalmente se considera como una operación separada de las fases de una operación de acceso a Filas, por eso se etiqueta como fase cero.
- La fase de ***precarga*** es un requisito previo para una operación de acceso a Filas, pero se realiza por separado de la operación de acceso a Filas en sí.
- Por el contrario, ***acceso, detección y restauración*** son tres fases diferentes que se realizan en secuencia para la operación de acceso a filas.

# Fase de precarga del amplificador de detección

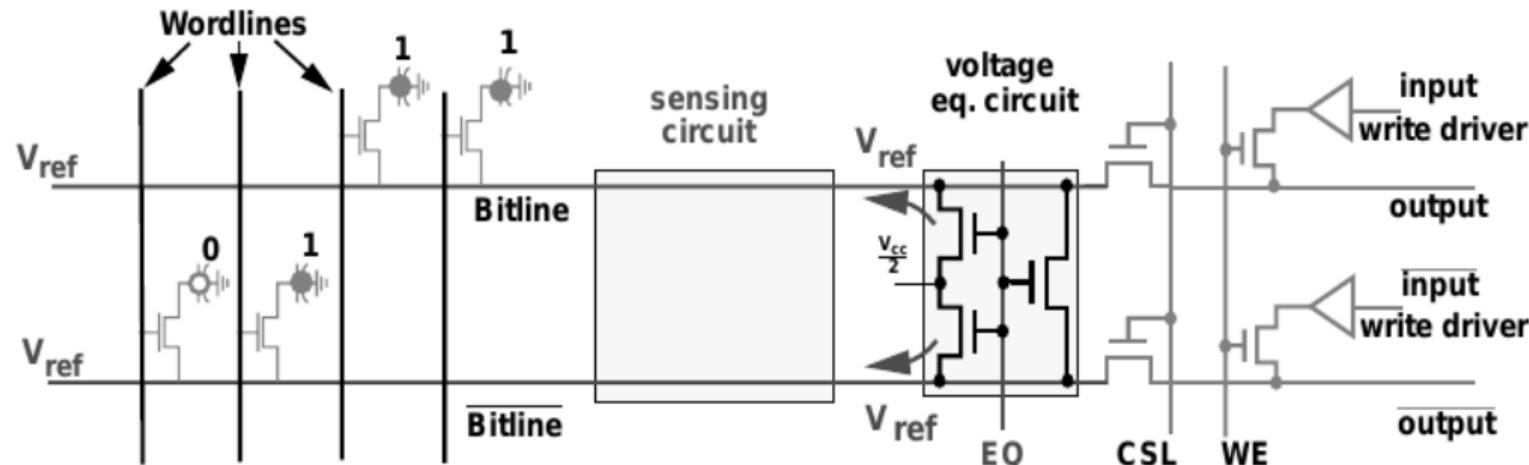


# Fase de precarga del amplificador de detección



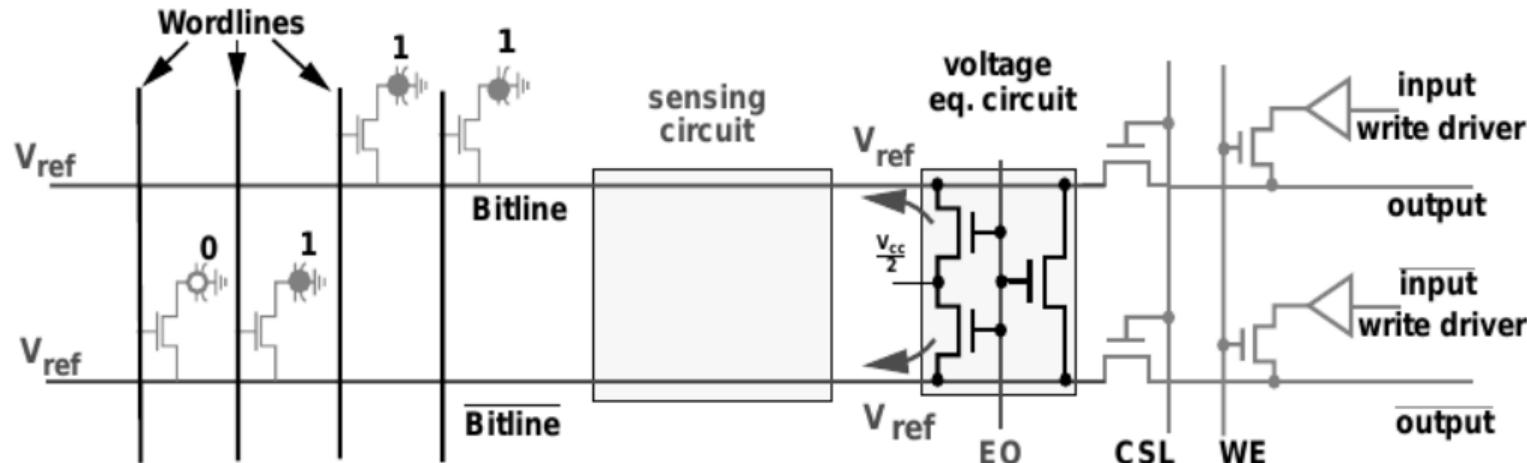
- Antes de que pueda comenzar el proceso de lectura de datos de una matriz DRAM, las Bitlines en una matriz DRAM se precargan a una tensión de referencia,  $V_{ref}$ .

# Fase de precarga del amplificador de detección



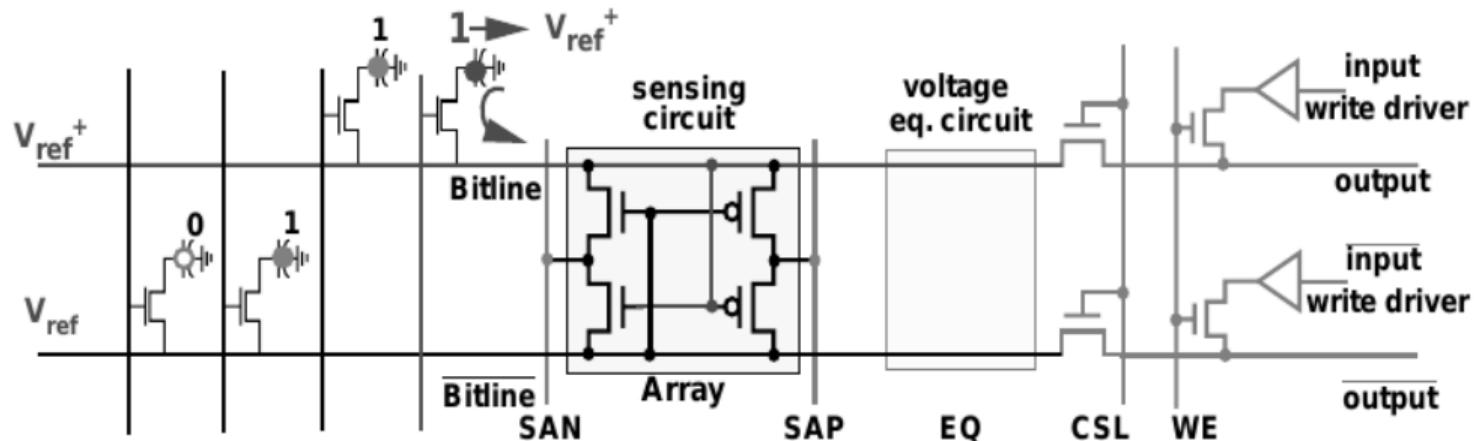
- Antes de que pueda comenzar el proceso de lectura de datos de una matriz DRAM, las Bitlines en una matriz DRAM se precargan a una tensión de referencia,  $V_{ref}$ .
- Para ello, se activa el circuito de ecualización que coloca la tensión de referencia en las Bitlines, que se precargan a  $V_{ref}$ .

# Fase de precarga del amplificador de detección

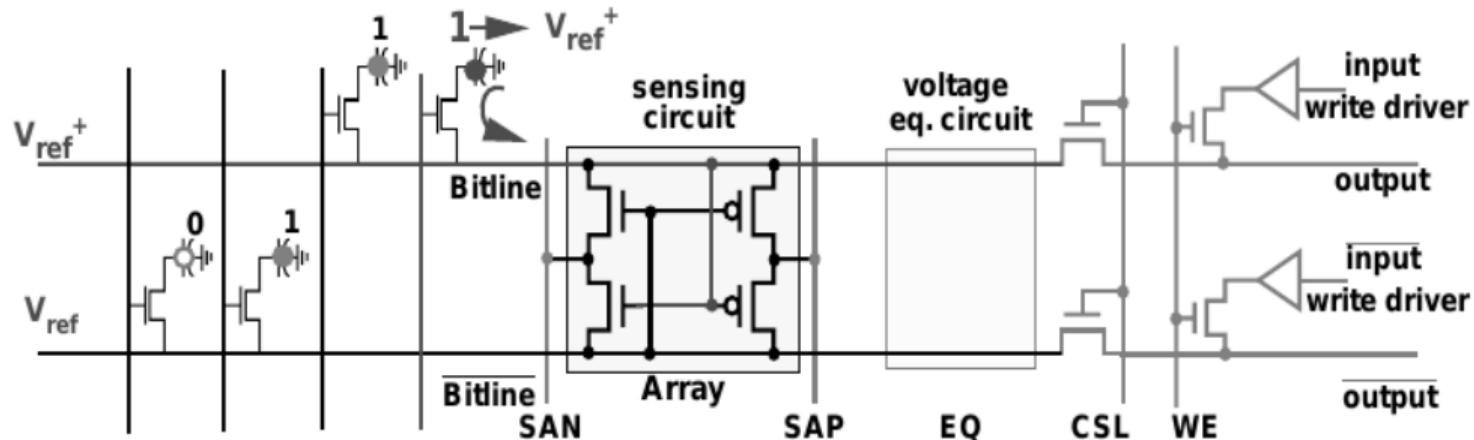


- Antes de que pueda comenzar el proceso de lectura de datos de una matriz DRAM, las Bitlines en una matriz DRAM se precargan a una tensión de referencia,  $V_{ref}$ .
- Para ello, se activa el circuito de ecualización que coloca la tensión de referencia en las Bitlines, que se precargan a  $V_{ref}$ .
- En muchos dispositivos DRAM modernos,  $V_{cc}/2$ , la tensión a medio camino entre la tensión de la fuente de alimentación y tierra, se usa como tensión de referencia.

# Fase de Acceso al amplificador de detección

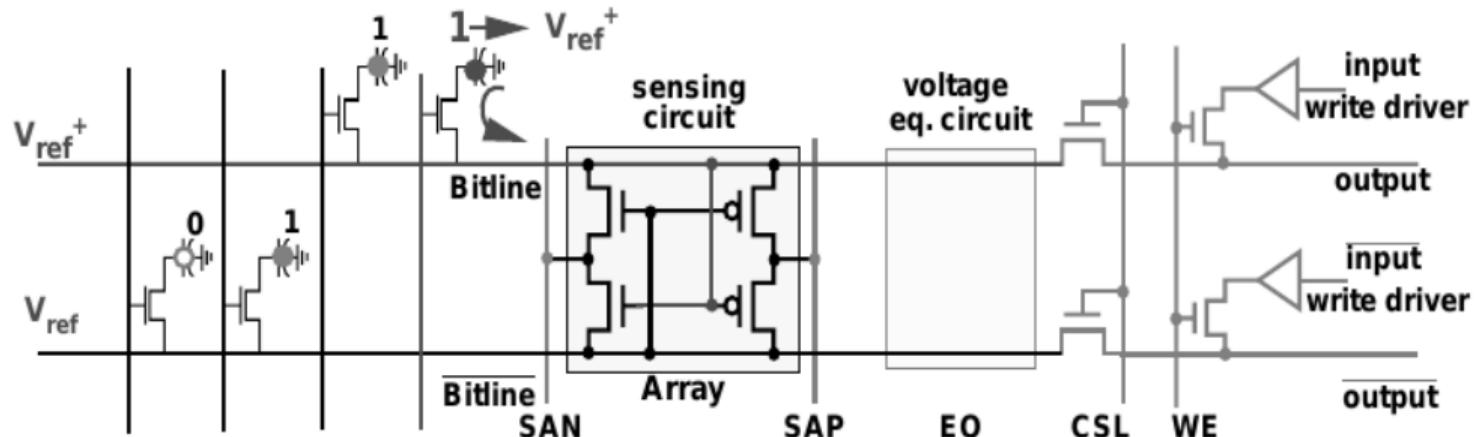


# Fase de Acceso al amplificador de detección



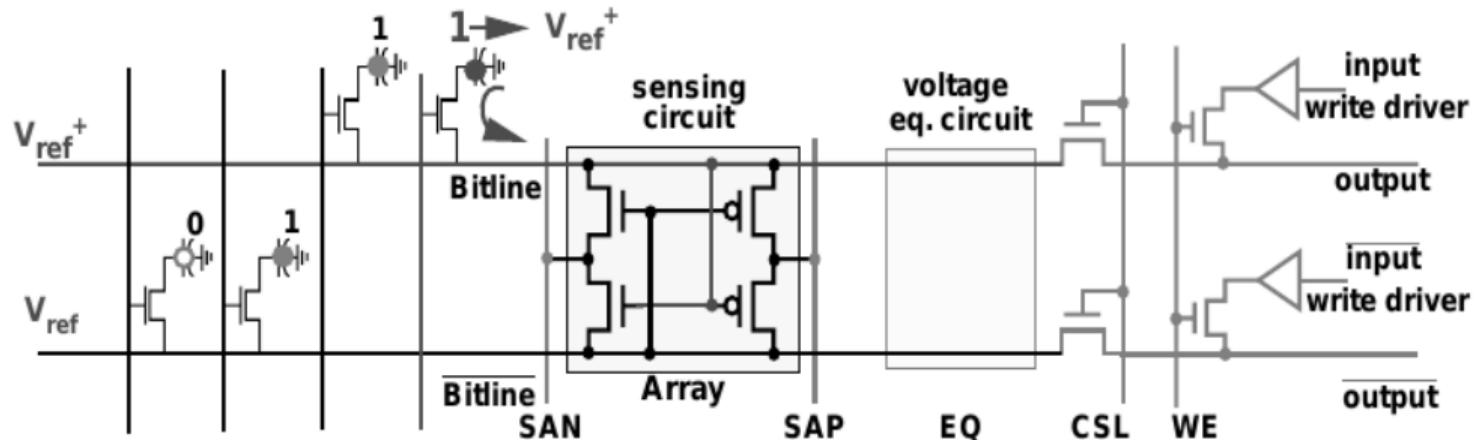
- Cuando se aplica una tensión a una wordline, ésta se sobrecarga a una tensión de al menos  $V_t$  por encima de  $V_{cc}$ . La tensión en la wordline activa los transistores de acceso y las celdas de almacenamiento seleccionadas descargan su contenido en las respectivas Bitlines.

# Fase de Acceso al amplificador de detección



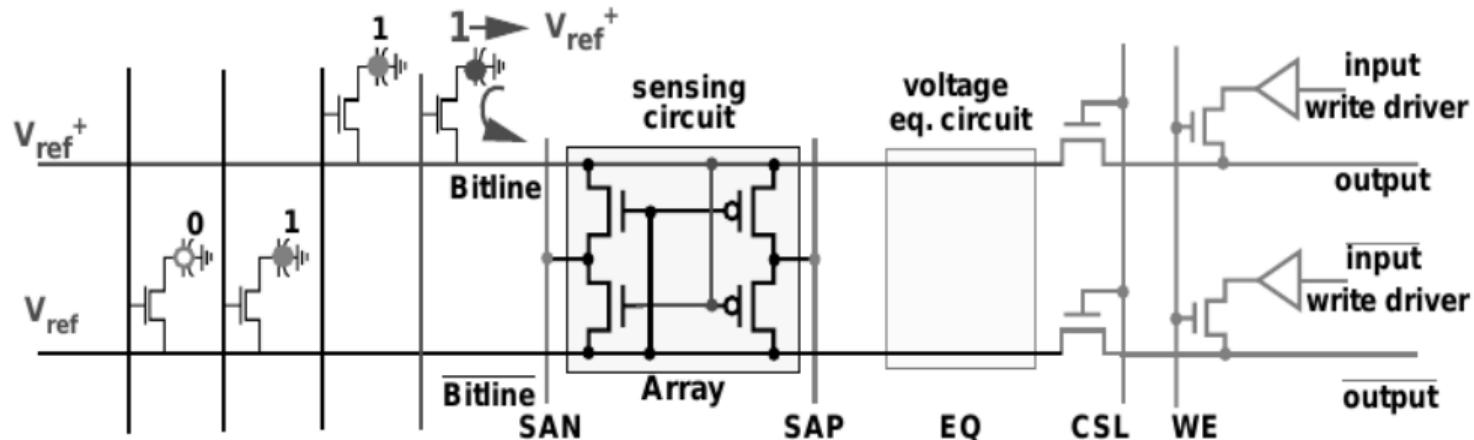
- Cuando se aplica una tensión a una wordline, ésta se sobrecarga a una tensión de al menos  $V_t$  por encima de  $V_{cc}$ . La tensión en la wordline activa los transistores de acceso y las celdas de almacenamiento seleccionadas descargan su contenido en las respectivas Bitlines.
- En este caso, dado que la tensión en la celda de almacenamiento representa un "1", el proceso de carga aumenta un pequeñísimo  $\delta$  la tensión en la Bitline de  $V_{ref}$  a  $V_{ref} + \delta$ .

# Fase de Acceso al amplificador de detección



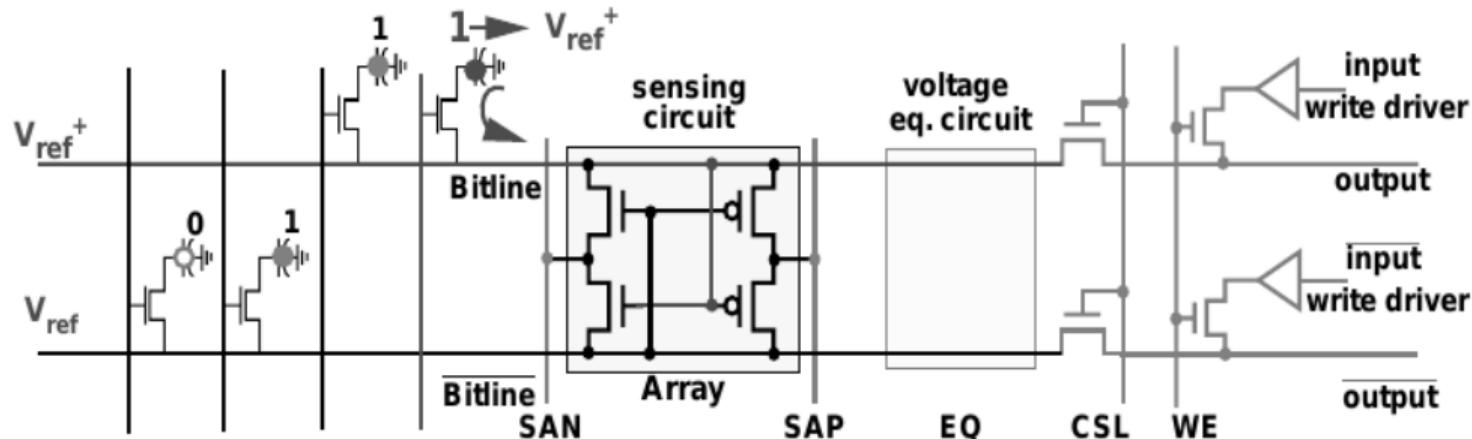
- A medida que cambia la tensión en la Bitline, afecta la operación del circuito de detección.

# Fase de Acceso al amplificador de detección



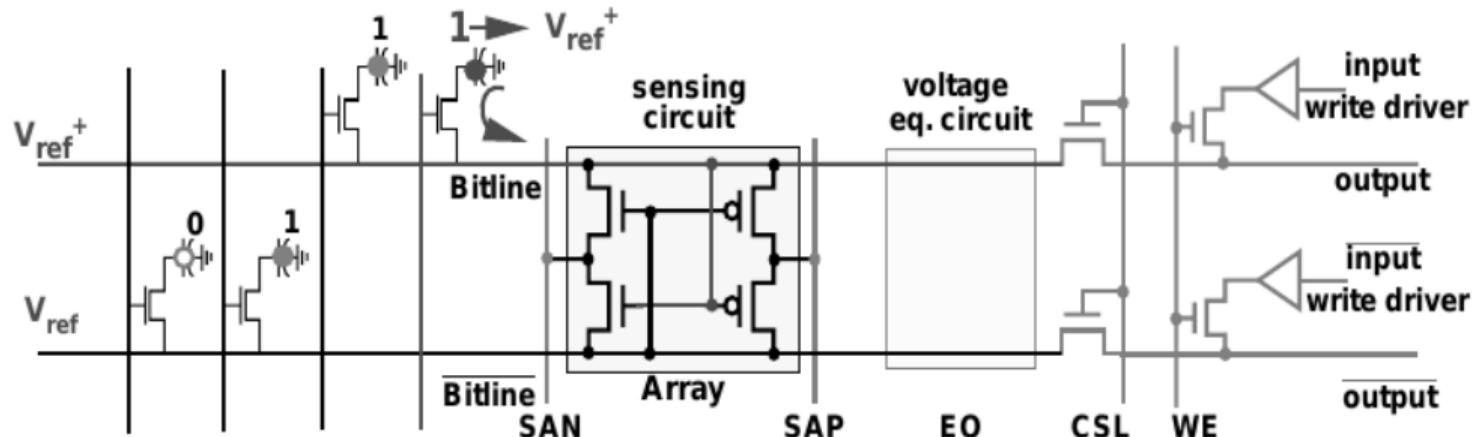
- A medida que cambia la tensión en la Bitline, afecta la operación del circuito de detección.
- En este caso, el incremento diferencial de tensión en Bitline hace que el NFet inferior sea más conductor que el NFet superior.

# Fase de Acceso al amplificador de detección



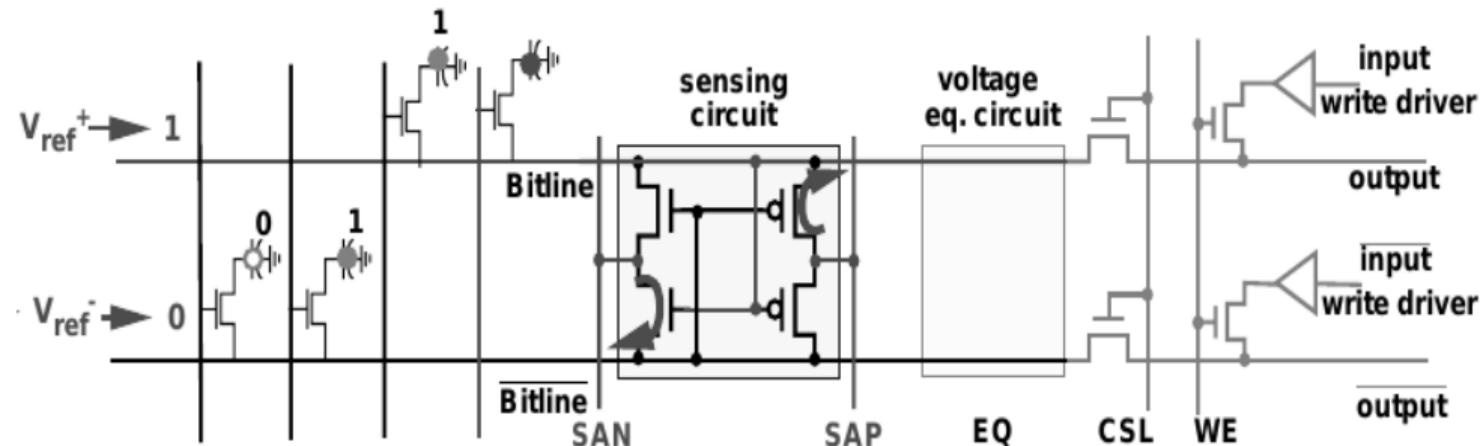
- A medida que cambia la tensión en la Bitline, afecta la operación del circuito de detección.
- En este caso, el incremento diferencial de tensión en Bitline hace que el NFet inferior sea más conductor que el NFet superior.
- Esa mínima diferencia de tensión también hace que el PFet inferior sea menos conductor que el PFet superior.

# Fase de Acceso al amplificador de detección

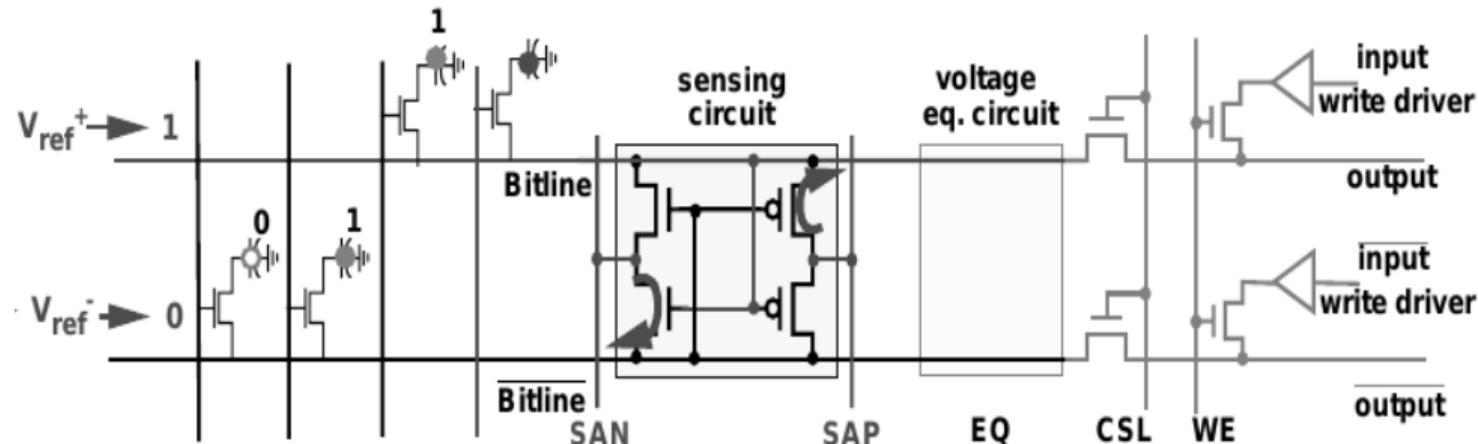


- A medida que cambia la tensión en la Bitline, afecta la operación del circuito de detección.
- En este caso, el incremento diferencial de tensión en Bitline hace que el NFet inferior sea más conductor que el NFet superior.
- Esa mínima diferencia de tensión también hace que el PFet inferior sea menos conductor que el PFet superior.
- La tensión de la Bitline polariza así el circuito de detección para la fase de detección.

# Fase de Detección al amplificador de detección

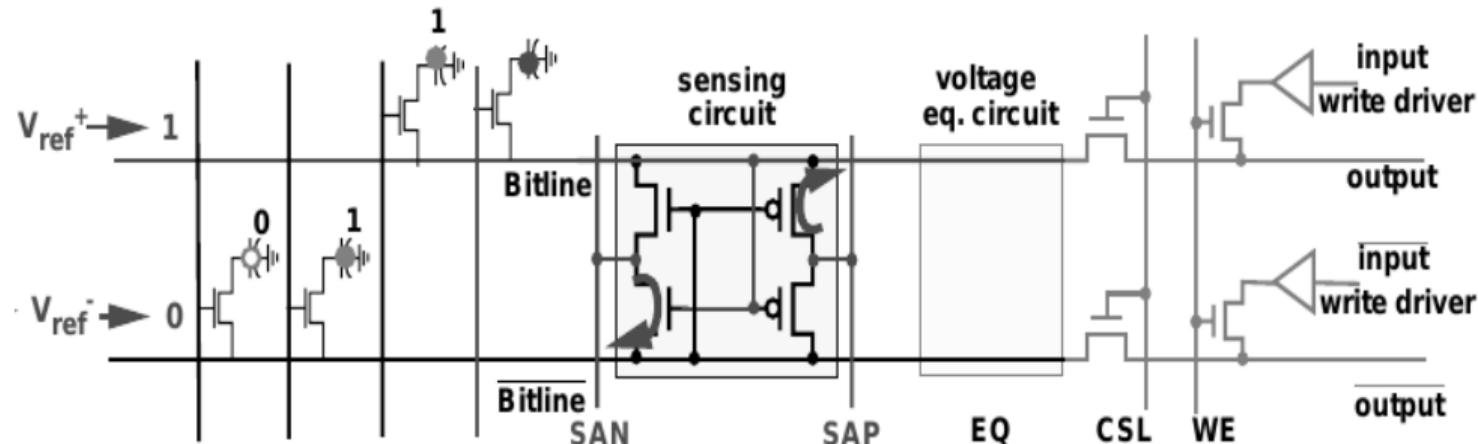


# Fase de Detección al amplificador de detección



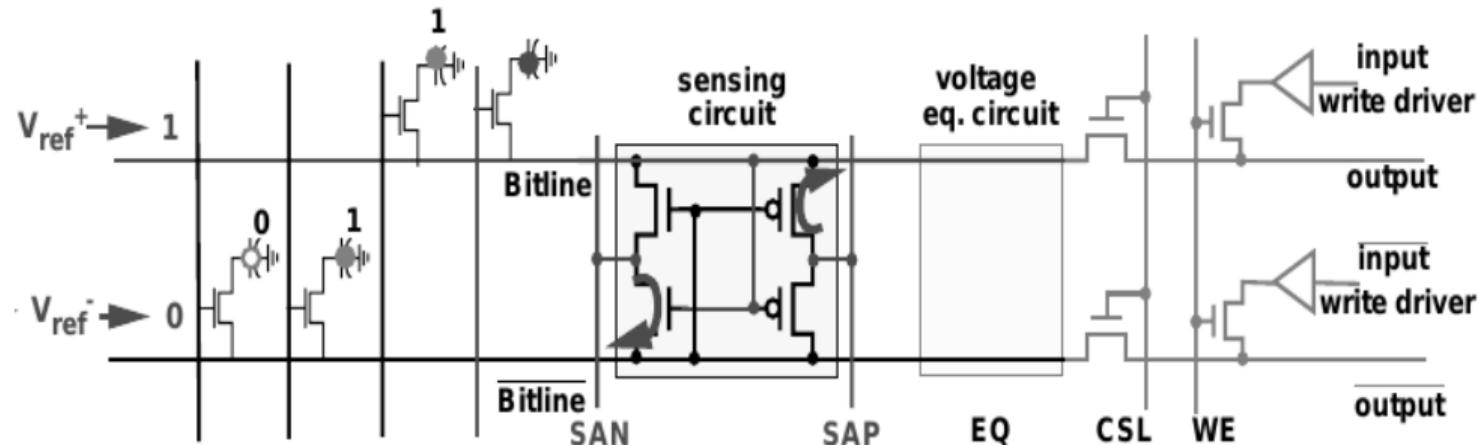
- A medida que el diferencial de tensión genera una polarización en el circuito de detección interconectado, se activa **SAN**, la señal de control del NFET del amplificador de detección del dispositivo DRAM, y disminuye la tensión en la Bitline inferior.

# Fase de Detección al amplificador de detección



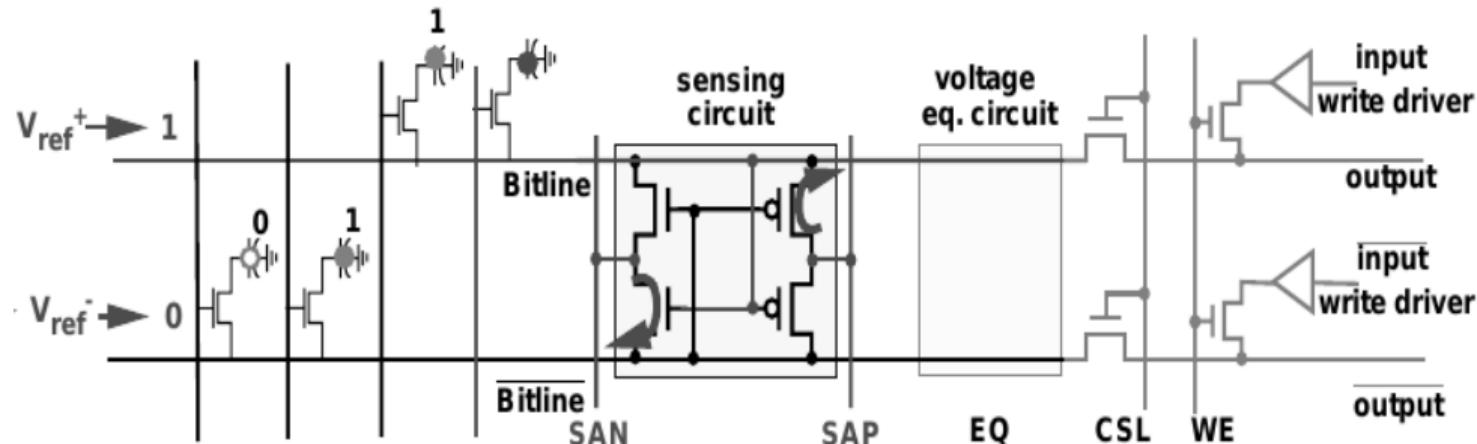
- A medida que el diferencial de tensión genera una polarización en el circuito de detección interconectado, se activa **SAN**, la señal de control del NFET del amplificador de detección del dispositivo DRAM, y disminuye la tensión en la Bitline inferior.
- La figura muestra que a medida que **SAN** se activa, el NFet inferior más conductivo permite que SAN baje la tensión de la Bitline inferior de  $V_{ref}$  a tierra.

# Fase de Detección al amplificador de detección



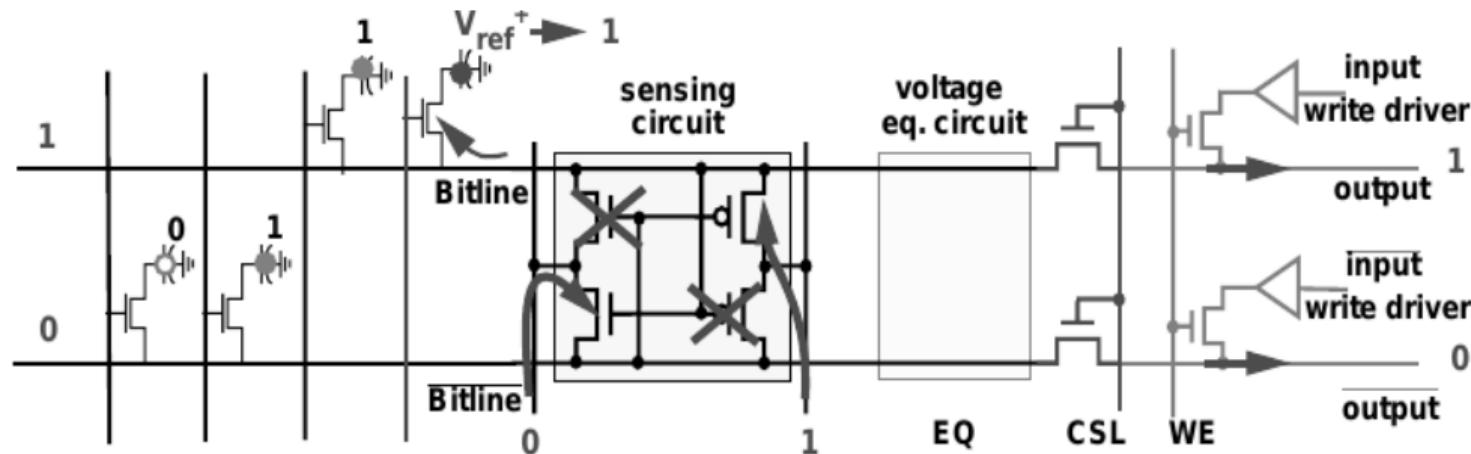
- De manera similar, **SAP**, la señal de control del amplificador de detección de PFet, impulsa la Bitline a un valor de tensión que representa el valor digital de "1".

# Fase de Detección al amplificador de detección

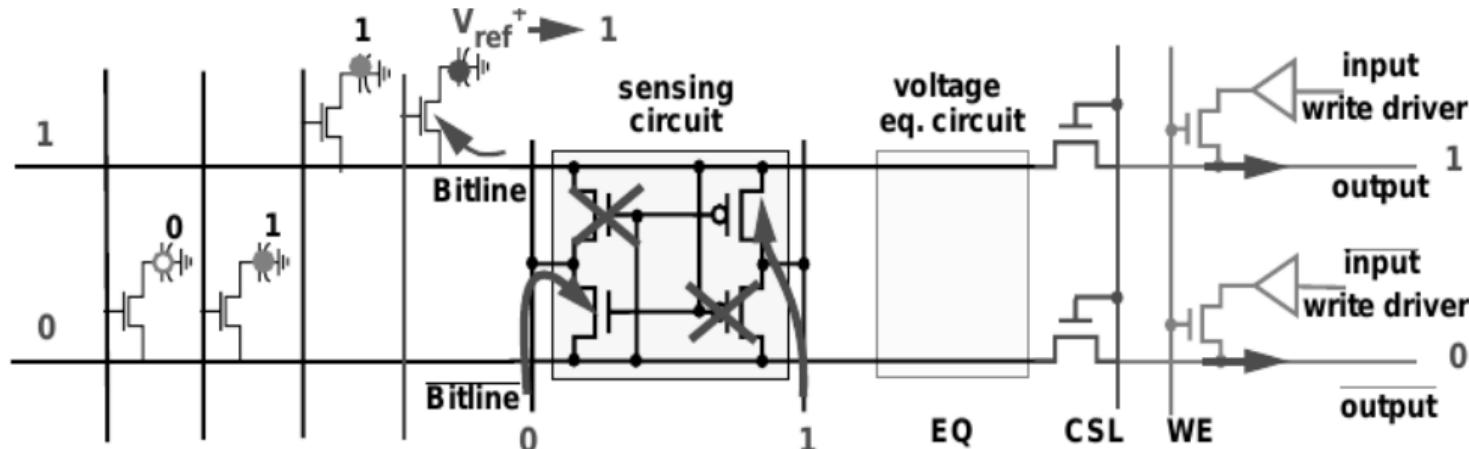


- De manera similar, **SAP**, la señal de control del amplificador de detección de PFet, impulsa la Bitline a un valor de tensión que representa el valor digital de "1".
- Las señales de control **SAN**y **SAP** fuerzan en conjunto al circuito amplificador de detección biestable a sus valores de tensión máximo o mínimo respectivos.

# Fase de Restore del amplificador de detección

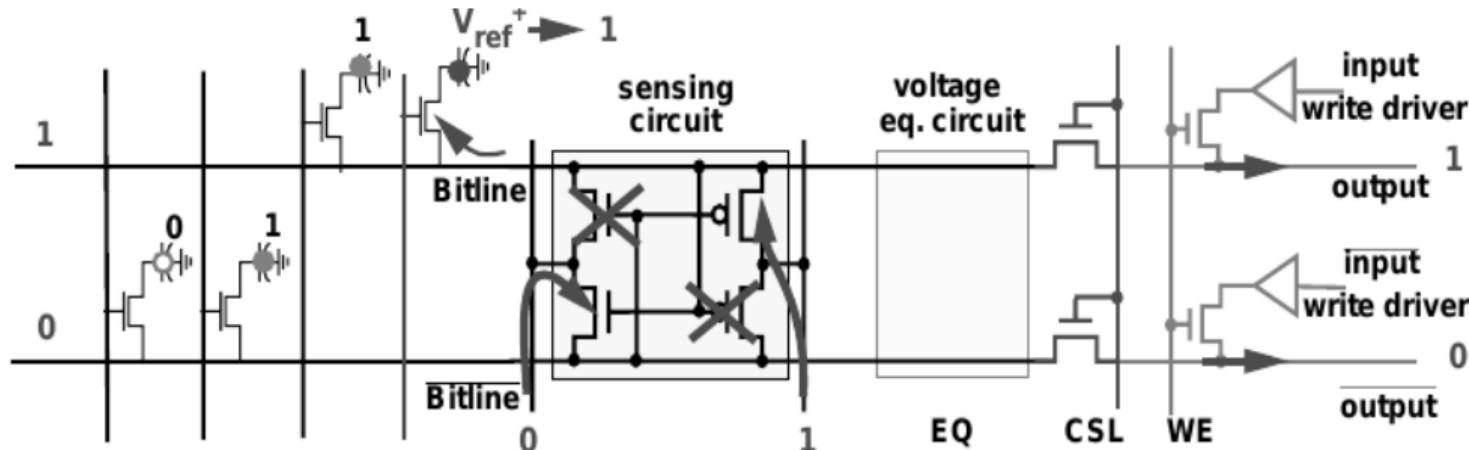


# Fase de Restore del amplificador de detección



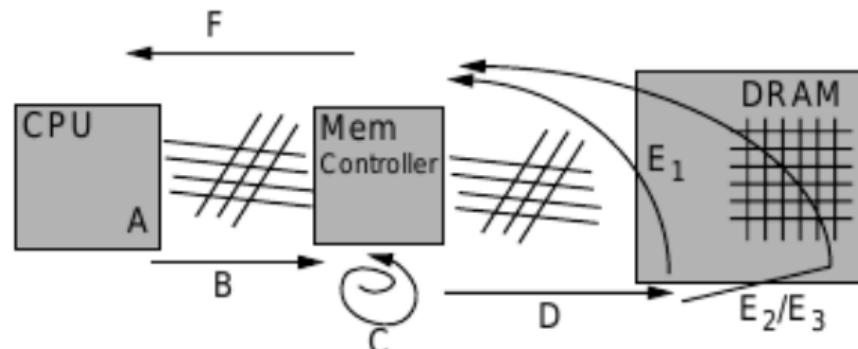
- Una vez que las Bitlines se conducen a los valores de tensión máximo o mínimo respectivos, la Word-line sobreexcitada permanece activa y la tensión de la Bitline activa restaura la carga en el capacitor de almacenamiento a través del transistor de acceso.

# Fase de Restore del amplificador de detección



- Una vez que las Bitlines se conducen a los valores de tensión máximo o mínimo respectivos, la Word-line sobreexcitada permanece activa y la tensión de la Bitline activa restaura la carga en el capacitor de almacenamiento a través del transistor de acceso.
- Al mismo tiempo, el valor de tensión en la Bitline puede extraerse del circuito amplificador de detección para entregar el valor del dato. De esta manera, se puede acceder al contenido de una fila DRAM simultáneamente con el proceso de restore de la fila.

# Diagrama general y transacciones



- El procesador ordena y encola los requerimientos de acceso a memoria y los envía al controlador.

A: Transaction request may be delayed in Queue

B: Transaction request sent to Memory Controller

C: Transaction converted to Command Sequences  
(may be queued)

D: Command/s Sent to DRAM

$E_1$ : Requires only a **CAS** or

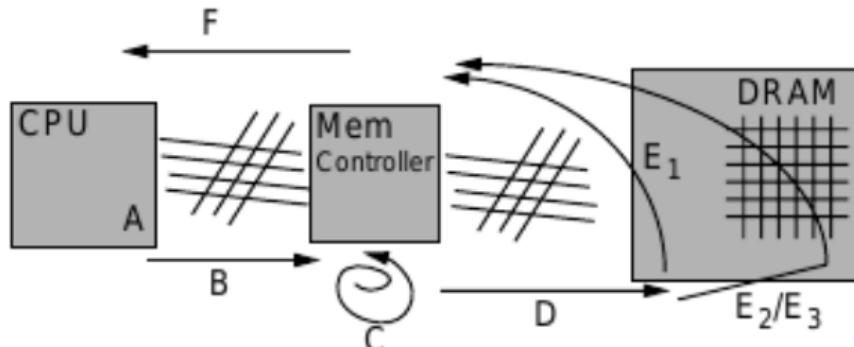
$E_2$ : Requires **RAS + CAS** or

$E_3$ : Requires **PRE + RAS + CAS**

F: Transaction sent back to CPU

$$\text{DRAM Latency} = A + B + C + D + E + F$$

# Diagrama general y transacciones



- El procesador ordena y encola los requerimientos de acceso a memoria y los envía al controlador.
- El controlador los encola hasta que la DRAM esté lista y se hayan resuelto los requerimientos en curso.

A: Transaction request may be delayed in Queue

B: Transaction request sent to Memory Controller

C: Transaction converted to Command Sequences  
(may be queued)

D: Command/s Sent to DRAM

E<sub>1</sub>: Requires only a **CAS** or

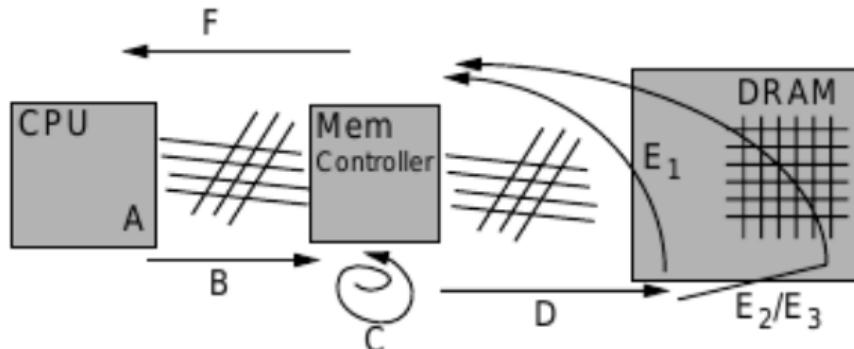
E<sub>2</sub>: Requires **RAS + CAS** or

E<sub>3</sub>: Requires **PRE + RAS + CAS**

F: Transaction sent back to CPU

$$\text{DRAM Latency} = A + B + C + D + E + F$$

# Diagrama general y transacciones



A: Transaction request may be delayed in Queue  
 B: Transaction request sent to Memory Controller  
 C: Transaction converted to Command Sequences (may be queued)

D: Command/s Sent to DRAM

E<sub>1</sub>: Requires only a **CAS** or

E<sub>2</sub>: Requires **RAS + CAS** or

E<sub>3</sub>: Requires **PRE + RAS + CAS**

F: Transaction sent back to CPU

$$\text{DRAM Latency} = A + B + C + D + E + F$$

- El procesador ordena y encola los requerimientos de acceso a memoria y los envía al controlador.
- El controlador los encola hasta que la DRAM esté lista y se hayan resuelto los requerimientos en curso.
- La interfaz del controlador de DRAM con los DIMM o con los DRAM devices, es mucho mas compleja que la de por ejemplo una memoria estática. En principio convierte dirección física en códigos de CAS y RAS. El detalle a continuación...

# Longitud de ráfaga (busrt) Programable

- Objetivo: Mejorar el ancho de banda de las transferencias de memoria.

# Longitud de ráfaga (busrt) Programable

- Objetivo: Mejorar el ancho de banda de las transferencias de memoria.
- Los SDRAM disponen de un Registro de Modo en el que se programa entre otras cosas la cantidad de lecturas o escrituras en ráfaga (burst) que se llevarán a cabo una vez activada la línea CAS.

# Longitud de ráfaga (busrt) Programable

- Objetivo: Mejorar el ancho de banda de las transferencias de memoria.
- Los SDRAM disponen de un Registro de Modo en el que se programa entre otras cosas la cantidad de lecturas o escrituras en ráfaga (burst) que se llevarán a cabo una vez activada la línea CAS.
- En lugar de comutar la señal CAS para transmitir en cada flanco un paquete de datos, como las BEDO, las SDRAM comutan una vez la línea CAS, y se transmiten tantos paquetes de datos en cada flanco ascendente de la señal de clock como indique su Registro de Modo.

# Longitud de ráfaga (busrt) Programable

- Objetivo: Mejorar el ancho de banda de las transferencias de memoria.
- Los SDRAM disponen de un Registro de Modo en el que se programa entre otras cosas la cantidad de lecturas o escrituras en ráfaga (burst) que se llevarán a cabo una vez activada la línea CAS.
- En lugar de comutar la señal CAS para transmitir en cada flanco un paquete de datos, como las BEDO, las SDRAM comutan una vez la línea CAS, y se transmiten tantos paquetes de datos en cada flanco ascendente de la señal de clock como indique su Registro de Modo.
- Al no tener que comutar la línea CAS el controlador de memoria, el controlador puede enviar requerimientos por el bus de control a otros DRAM Devices aumentando el ancho de banda del bus.

# Longitud de ráfaga (busrt) Programable

- Objetivo: Mejorar el ancho de banda de las transferencias de memoria.
- Los SDRAM disponen de un Registro de Modo en el que se programa entre otras cosas la cantidad de lecturas o escrituras en ráfaga (burst) que se llevarán a cabo una vez activada la línea CAS.
- En lugar de comutar la señal CAS para transmitir en cada flanco un paquete de datos, como las BEDO, las SDRAM comutan una vez la línea CAS, y se transmiten tantos paquetes de datos en cada flanco ascendente de la señal de clock como indique su Registro de Modo.
- Al no tener que comutar la línea CAS el controlador de memoria, el controlador puede enviar requerimientos por el bus de control a otros DRAM Devices aumentando el ancho de banda del bus.
- Una vez terminado el burst en este DRAM Device, habrá otros Devices ya precargados y disponibles para transmitir datos en ráfaga.

# CAS Latency Programable

- CAS Latency o Delay: Cantidad de ciclos de clock que transcurren desde que se activa CAS hasta que el dato está disponible.

# CAS Latency Programable

- CAS Latency o Delay: Cantidad de ciclos de clock que transcurren desde que se activa CAS hasta que el dato está disponible.
- Se lo solía referir también como tiempo de acceso.

## CAS Latency Programable

- CAS Latency o Delay: Cantidad de ciclos de clock que transcurren desde que se activa CAS hasta que el dato está disponible.
- Se lo solía referir también como tiempo de acceso.
- Las SDRAM permiten programar esta cantidad de ciclo de clock en su Registro de Modo.

## CAS Latency Programable

- CAS Latency o Delay: Cantidad de ciclos de clock que transcurren desde que se activa CAS hasta que el dato está disponible.
- Se lo solía referir también como tiempo de acceso.
- Las SDRAM permiten programar esta cantidad de ciclo de clock en su Registro de Modo.
- De este modo se facilita la coexistencia en un mismo sistema de memoria de SDRAM devices de diferentes características en lo que a tiempo de acceso se refiere.

# Temario

## 1 El sistema de Memoria

- Antecedentes
- Rol de la memoria en un computador
- Jerarquía de memoria

## 2 Tecnologías de Memoria

- Clasificación
- Memorias No volátiles
- Memorias Volátiles
- Memorias y velocidad del Procesador

## 3 Memoria Cache

- Principio de Funcionamiento
- Métricas y Performance
- Hardware dedicado = + complejidad
- organización de un cache

## 4 Cache en Sistemas Multiprocesador

## ● Organización de Sistemas Multiprocesador

- Coherencia de un cache
- Protocolos de Coherencia para Multicore
- Casos del Mundo real

## 5 Memorias Dinámicas

- Introducción
- Organización interna

## 6 Standards

- Estado del arte
- JEDEC SDRAM

## 7 Controladores de Memoria

- Introducción General
- Arquitectura

## 8 Configuración

- Configuración del DRAM Device

## ● Entrada Salida de Datos

## 9 Integración con el sistema Cache

- Evitar cuellos de botella es la clave

## 10 Casos Prácticos

- Beagle Bone Black
- Memorias DDR en la BBB
- Controlador de DDRn SDRAM en la BBB

## 11 SRAM Cuestiones de Implementación

- Vistazo introductorio
- Decodificación
- Implementación

## 12 DRAM Detalles de implementación

- Acceso a las celdas
- JEDEC DDR SDRAM
- Protocolo de acceso

# Dual Edge Clocking

- Las DDR SDRAM conservan las características de las SDRAM agregando nuevas prestaciones.

# Dual Edge Clocking

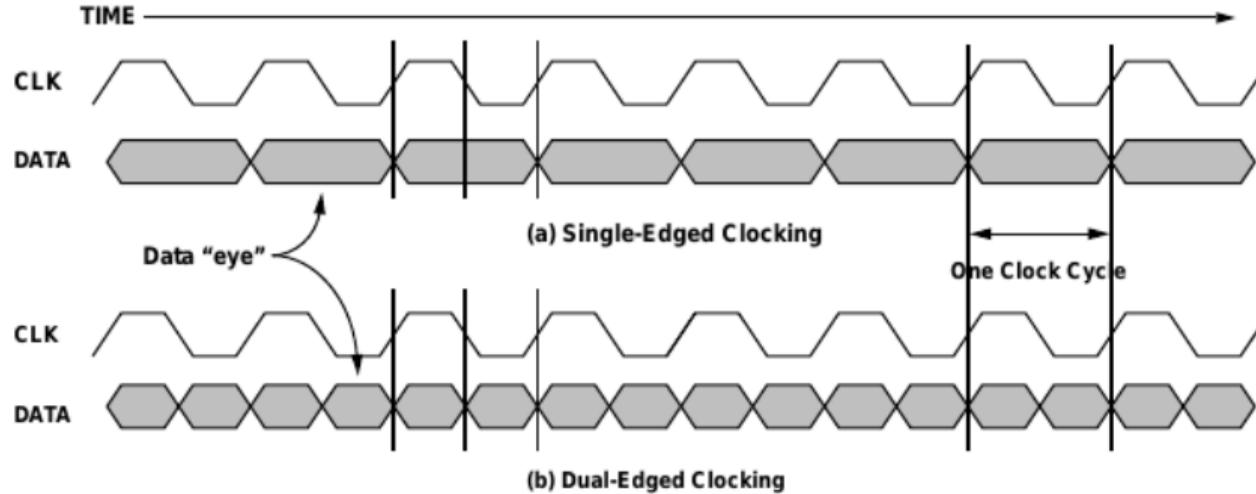
- Las DDR SDRAM conservan las características de las SDRAM agregando nuevas prestaciones.
- Las transacciones de direcciones y control se sincronizan con el flanco ascendente de la señal de clock igual que las SDRAM.

# Dual Edge Clocking

- Las DDR SDRAM conservan las características de las SDRAM agregando nuevas prestaciones.
- Las transacciones de direcciones y control se sincronizan con el flanco ascendente de la señal de clock igual que las SDRAM.
- Las transmisiones de datos por su parte usan un clock dual edge.

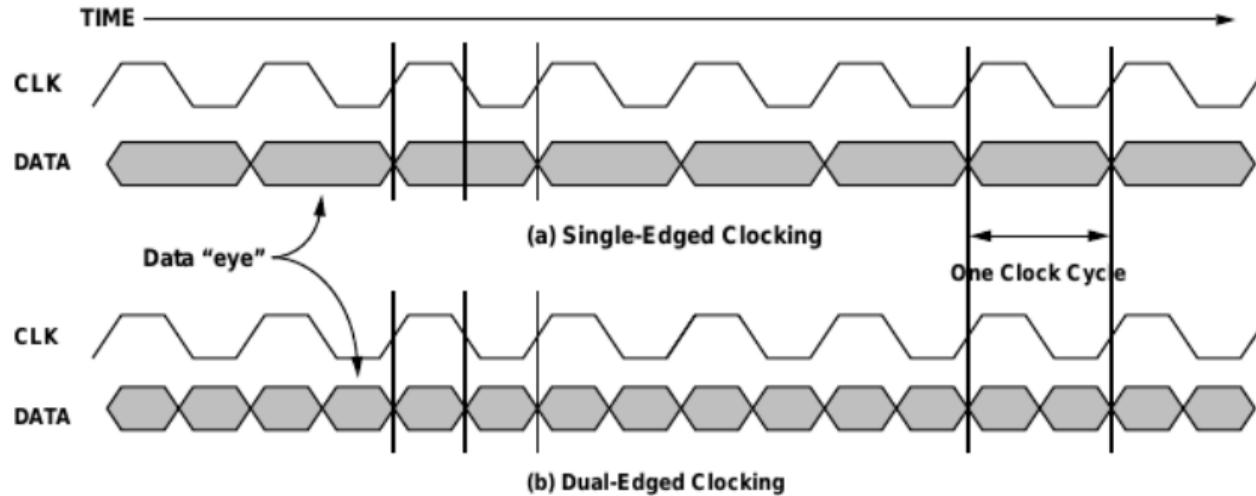
# Dual Edge Clocking

- Las DDR SDRAM conservan las características de las SDRAM agregando nuevas prestaciones.
- Las transacciones de direcciones y control se sincronizan con el flanco ascendente de la señal de clock igual que las SDRAM.
- Las transmisiones de datos por su parte usan un clock dual edge.



# Dual Edge Clocking

- Las DDR SDRAM conservan las características de las SDRAM agregando nuevas prestaciones.
- Las transacciones de direcciones y control se sincronizan con el flanco ascendente de la señal de clock igual que las SDRAM.
- Las transmisiones de datos por su parte usan un clock dual edge.



- Requieren la mitad de ciclos de clock para acceder a la misma cantidad de datos.

# Dual Edge Clocking

- El tamaño de la ventana válida (también llamada “ojo de datos”) es la mitad que en una SDRAM.

# Dual Edge Clocking

- El tamaño de la ventana válida (también llamada “ojo de datos”) es la mitad que en una SDRAM.
- El vuelco de los datos al bus (Read), y el muestreo a la entrada de la DRAM (write), tienen ahora diferente relación con el flanco de la señal de clock.

# Dual Edge Clocking

- El tamaño de la ventana válida (también llamada “ojo de datos”) es la mitad que en una SDRAM.
- El vuelco de los datos al bus (Read), y el muestreo a la entrada de la DRAM (write), tienen ahora diferente relación con el flanco de la señal de clock.
- En la lectura del DRAM device, el flanco dispara el vuelco de los datos hacia el bus. Se refiere como “alineada al flanco de los datos”.

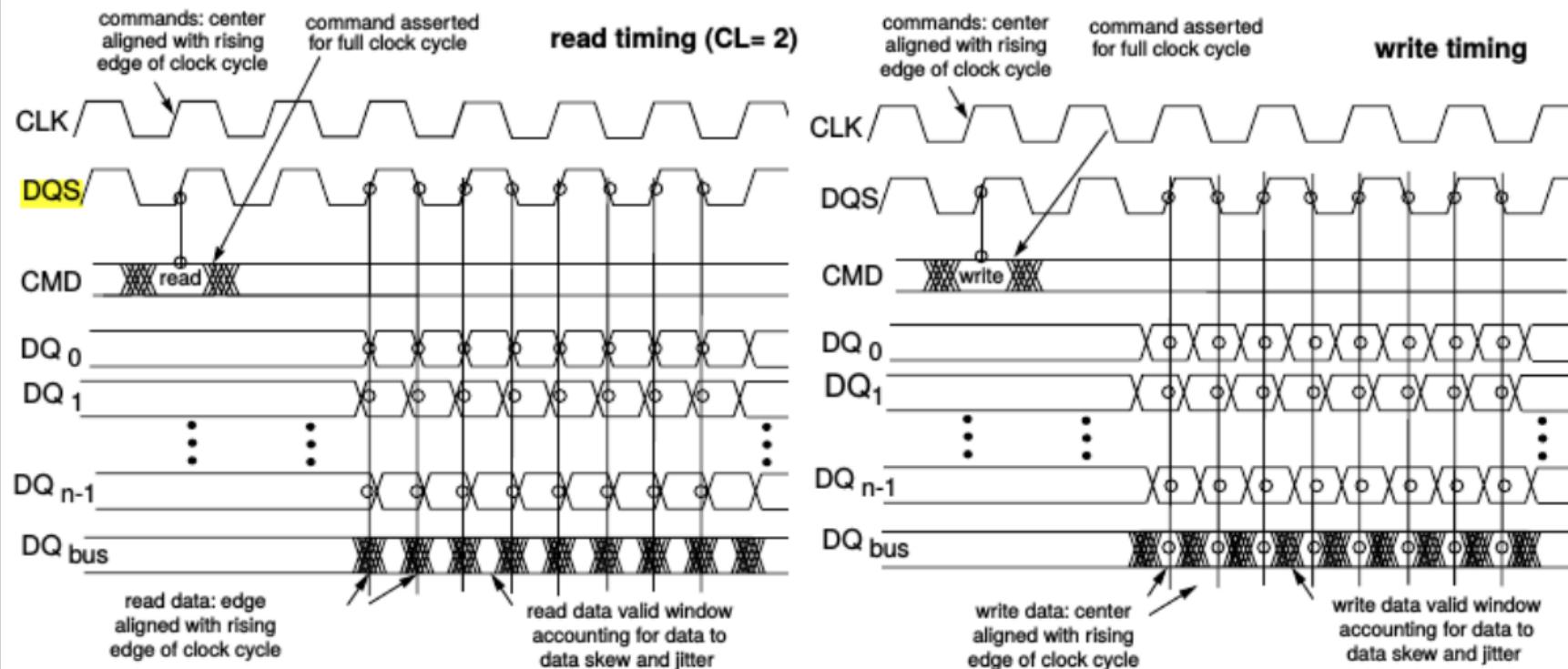
# Dual Edge Clocking

- El tamaño de la ventana válida (también llamada “ojo de datos”) es la mitad que en una SDRAM.
- El vuelco de los datos al bus (Read), y el muestreo a la entrada de la DRAM (write), tienen ahora diferente relación con el flanco de la señal de clock.
- En la lectura del DRAM device, el flanco dispara el vuelco de los datos hacia el bus. Se refiere como “alineada al flanco de los datos”.
- En la escritura del DRAM device, se necesita alinear el flanco con el centro de los datos para asegurarse que la habilitación de los circuitos de entrada del DRAM device encuentren señales válidas en el bus de datos.

# Dual Edge Clocking

- El tamaño de la ventana válida (también llamada “ojo de datos”) es la mitad que en una SDRAM.
- El vuelco de los datos al bus (Read), y el muestreo a la entrada de la DRAM (write), tienen ahora diferente relación con el flanco de la señal de clock.
- En la lectura del DRAM device, el flanco dispara el vuelco de los datos hacia el bus. Se refiere como “alineada al flanco de los datos”.
- En la escritura del DRAM device, se necesita alinear el flanco con el centro de los datos para asegurarse que la habilitación de los circuitos de entrada del DRAM device encuentren señales válidas en el bus de datos.
- Se requiere asegurar precisión en ambas operaciones.

# DQS en un CAS para Lectura y para Escritura



## Señal adicional: DQS

- A medida que aumenta la velocidad, los datos en las entradas pueden desfasarse en el tiempo respecto de otras señales que corren libremente como el **Clock**.

## Señal adicional: DQS

- A medida que aumenta la velocidad, los datos en las entradas pueden desfasarse en el tiempo respecto de otras señales que corren libremente como el **Clock**.
- En las entradas de datos de un DRAM Device, pueden existir tambien desfasajes por variaciones de temperatura, tensión, o en la carga del dispositivo.

## Señal adicional: DQS

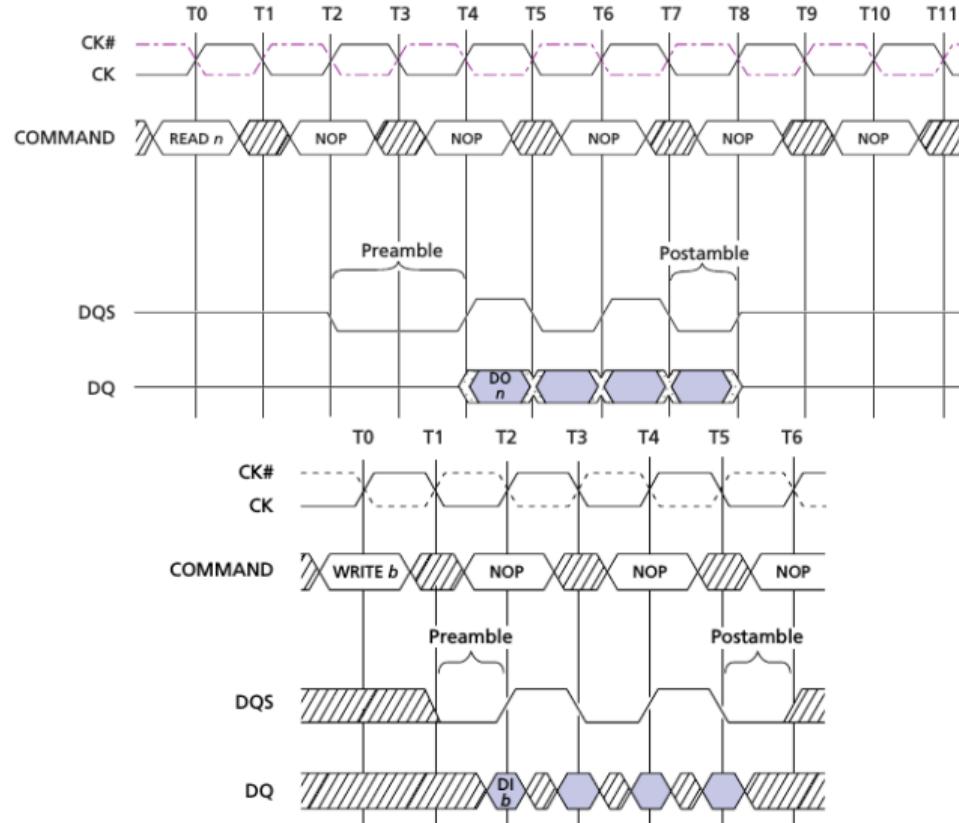
- A medida que aumenta la velocidad, los datos en las entradas pueden desfasarse en el tiempo respecto de otras señales que corren libremente como el **Clock**.
- En las entradas de datos de un DRAM Device, pueden existir tambien desfasajes por variaciones de temperatura, tensión, o en la carga del dispositivo.
- Estos efectos se acentúan en los dispositivos DDR. Por lo tanto se necesita usar señales de strobe, generadas por el controlador de memoria, en lugar de señales libres como **Clock**.

## Señal adicional: DQS

- A medida que aumenta la velocidad, los datos en las entradas pueden desfasarse en el tiempo respecto de otras señales que corren libremente como el **Clock**.
- En las entradas de datos de un DRAM Device, pueden existir tambien desfasajes por variaciones de temperatura, tensión, o en la carga del dispositivo.
- Estos efectos se acentúan en los dispositivos DDR. Por lo tanto se necesita usar señales de strobe, generadas por el controlador de memoria, en lugar de señales libres como **Clock**.

Por ello, los controladores de DRAM DDR tienen una señal DQS, que se encarga que las operaciones READ se generen con las señales de datos alienadas con el flanco ascendente y las WRITE con el flanco descendente centrado en la señal de datos.

# Señal adicional: DQS



# Temario

- 1 El sistema de Memoria
  - Antecedentes
  - Rol de la memoria en un computador
  - Jerarquía de memoria
- 2 Tecnologías de Memoria
  - Clasificación
  - Memorias No volátiles
  - Memorias Volátiles
  - Memorias y velocidad del Procesador
- 3 Memoria Cache
  - Principio de Funcionamiento
  - Métricas y Performance
  - Hardware dedicado = + complejidad
  - organización de un cache
- 4 Cache en Sistemas Multiprocesador

- 5 Organización de Sistemas Multiprocesador
  - Coherencia de un cache
  - Protocolos de Coherencia para Multicore
  - Casos del Mundo real
- 6 Memorias Dinámicas
  - Introducción
  - Organización interna
- 7 Standards
  - Estado del arte
  - JEDEC SDRAM
- 8 Controladores de Memoria
  - Introducción General
  - Arquitectura
- 9 Configuración
  - Configuración del DRAM Device
- 10 Entrada Salida de Datos
  - Integración con el sistema Cache
  - Evitar cuellos de botella es la clave
- 11 Casos Prácticos
  - Beagle Bone Black
  - Memorias DDR en la BBB
  - Controlador de DDRn SDRAM en la BBB
- 12 SRAM Cuestiones de Implementación
  - Vistazo introductorio
  - Decodificación
  - Implementación
- 13 DRAM Detalles de implementación
  - Acceso a las celdas
  - JEDEC DDR SDRAM
  - Protocolo de acceso

# Funciones que se resuelven

- Un protocolo de acceso a memoria define los comandos y restricciones de temporización que un controlador de memoria debe utilizar para manejar sus transferencias de datos con la memoria DRAM

# Funciones que se resuelven

- Un protocolo de acceso a memoria define los comandos y restricciones de temporización que un controlador de memoria debe utilizar para manejar sus transferencias de datos con la memoria DRAM
- Nos proponemos analizar un protocolo de acceso genérico aplicable a SDRAM y DDRx SDRAM.

# Funciones que se resuelven

- Un protocolo de acceso a memoria define los comandos y restricciones de temporización que un controlador de memoria debe utilizar para manejar sus transferencias de datos con la memoria DRAM
- Nos proponemos analizar un protocolo de acceso genérico aplicable a SDRAM y DDRx SDRAM.
- Examinar en detalle cualquier protocolo de acceso a una SDRAM es sumamente complejo.

# Funciones que se resuelven

- Un protocolo de acceso a memoria define los comandos y restricciones de temporización que un controlador de memoria debe utilizar para manejar sus transferencias de datos con la memoria DRAM
- Nos proponemos analizar un protocolo de acceso genérico aplicable a SDRAM y DDRx SDRAM.
- Examinar en detalle cualquier protocolo de acceso a una SDRAM es sumamente complejo.
- Esta complejidad proviene de la gran variedad de comandos en los sistemas de memoria modernos.

# Funciones que se resuelven

- Un protocolo de acceso a memoria define los comandos y restricciones de temporización que un controlador de memoria debe utilizar para manejar sus transferencias de datos con la memoria DRAM
- Nos proponemos analizar un protocolo de acceso genérico aplicable a SDRAM y DDRx SDRAM.
- Examinar en detalle cualquier protocolo de acceso a una SDRAM es sumamente complejo.
- Esta complejidad proviene de la gran variedad de comandos en los sistemas de memoria modernos.
- Sin embargo es posible definir un set de comandos básicos que es posible encontrar en la mayoría de los sistemas SDRAM modernos.

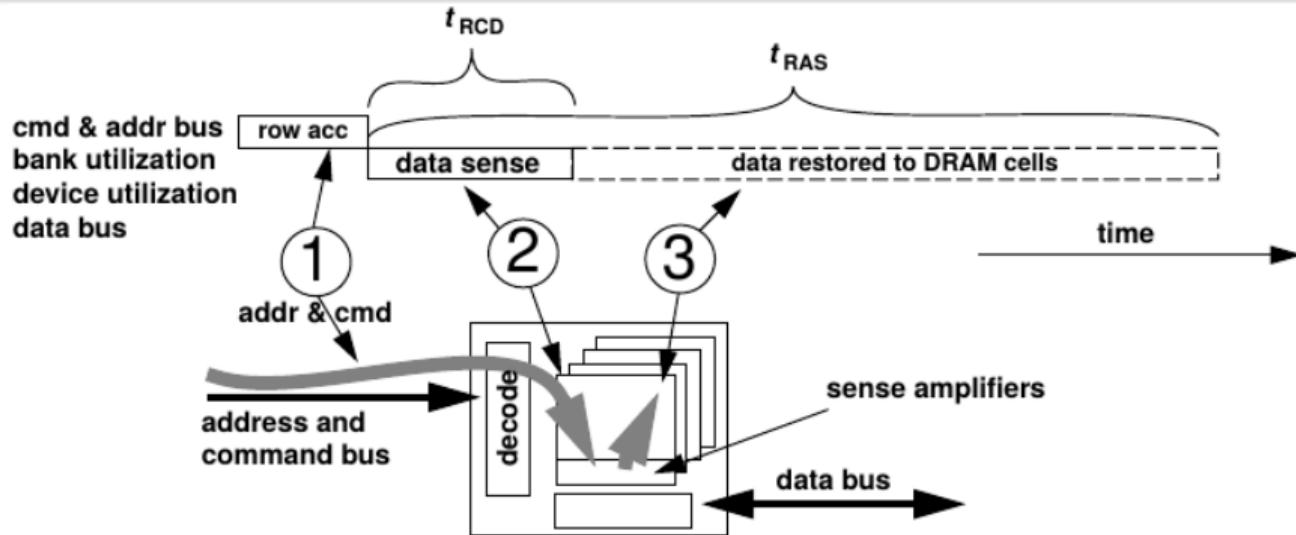
# Valores estándar de Timing

Parametro	Descripción
$t_{AL}$	<b>Added Latency</b> en accesos a columnas. Se usa en DDRx SDRAM devices para comandos CAS enviados.
$t_{BURST}$	<b>Duración del Burst de Datos</b> . Período de tiempo en que el <i>data burst</i> ocupa el bus de datos. Tipicamente 4 u 8 <i>beats</i> de datos. En DDR SDRAM, 4 <i>beats</i> de datos consumen 2 ciclos completos de clock.
$t_{CAS}$	<b>Column Access Strobe latency</b> . Intervalo de Tiempo entre el comando de acceso a columna y el inicio del retorno de datos desde el/los DRAM device/s. Normalmente nombrado como $t_{CL}$ .
$t_{CCD}$	<b>Column-to-Column Delay</b> . Mínimo timing en comandos de columna, determinado por la longitud del <i>burst</i> interno ( <i>prefetch</i> ). Las lecturas de columnas utilizan múltiples bursts internos para formar un burst mas largo. $t_{CCD}$ es 2 <i>beats</i> (1 ciclo) en SDRAM DDR, 4 (2 ciclos) en DDR2, y 8 (4 ciclos) en DDR3.
$t_{CMD}$	<b>Tiempo de Transporte de Comando</b> . Tiempo durante el cual el comando ocupa el bus de Comandos y es transportado desde el Controlador de DRAM hasta el/los DRAM device/s.
$t_{CWD}$	<b>Column Write Delay</b> . Tiempo entre el envío de un comando <i>column-write</i> y el vuelco de datos al bus por parte del controlador de DRAM.
$t_{FAW}$	<b>Four (row) bank Activation Window</b> . Ranura de tiempo en la cual se pueden activar un máximo de cuatro bancos. Limita el perfil de corriente pico en DDR2 y DDR3 DRAMs con mas de 4 bancos.
$t_{OST}$	<b>ODT Switching Time</b> . Tiempo para comutar el control ODT, de rango a rango.

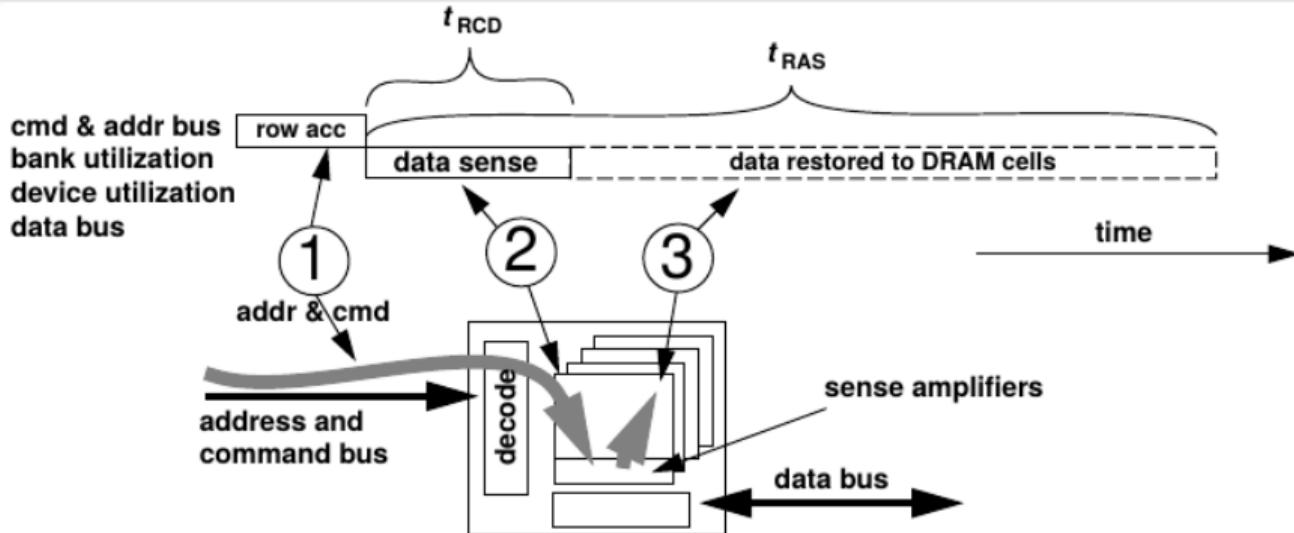
# Valores estándar de Timing

Parametro	Descripción
$t_{RAS}$	<b>Row Access Strobe.</b> Tiempo entre el comando <i>row access</i> y la regeneración de datos en un DRAM array. Un banco de DRAM no se puede precargar hasta el último $t_{RAS}$ luego de la activación del banco previa.
$t_{RC}$	<b>Row Cycle.</b> Tiempo entre accesos a diferentes filas en un banco. $t_{RC} = t_{RAS} + t_{RP}$ .
$t_{RCD}$	<b>Row to Column command Delay.</b> Tiempo entre un <i>row access</i> y los datos listos en los amplificadores de detección.
$t_{RFC}$	<b>Refresh Cycle time.</b> Tiempo entre los comandos <i>Refresh</i> y <i>Activation</i> .
$t_{RP}$	<b>Row Precharge.</b> Tiempo que le toma a un DRAM array precargarse para otro <i>row access</i> .
$t_{RRD}$	<b>Row activation to Row activation Delay.</b> Tiempo mínimo entre dos comandos de activación de fila en el mismo DRAM device. Limita el perfil de corriente pico.
$t_{RTP}$	<b>Read to Precharge.</b> Tiempo entre un comando de lectura y otro de precarga.
$t_{RTRS}$	<b>Rank-to-rank switching time.</b> Un ciclo completo en DDR SDRAM. No se usa en sistemas SDRAM ni RDRAM Directo.
$t_{WR}$	<b>Write Recovery time.</b> Mínimo intervalo de tiempo entre el fin de una escritura burst y el inicio de un comando de precarga. Permite a los amplificadores de detección restaurar la carga en las celdas.
$t_{WTR}$	<b>Write To Read delay time.</b> Tiempo mínimo entre el final de una escritura burst y el inicio de un comando <i>column read</i> . Permite a la E/S forzar a los amplificadores de detección a saturar su salida antes de que se ejecute el comando Read.

# Comando Row Access: Formato y timing.

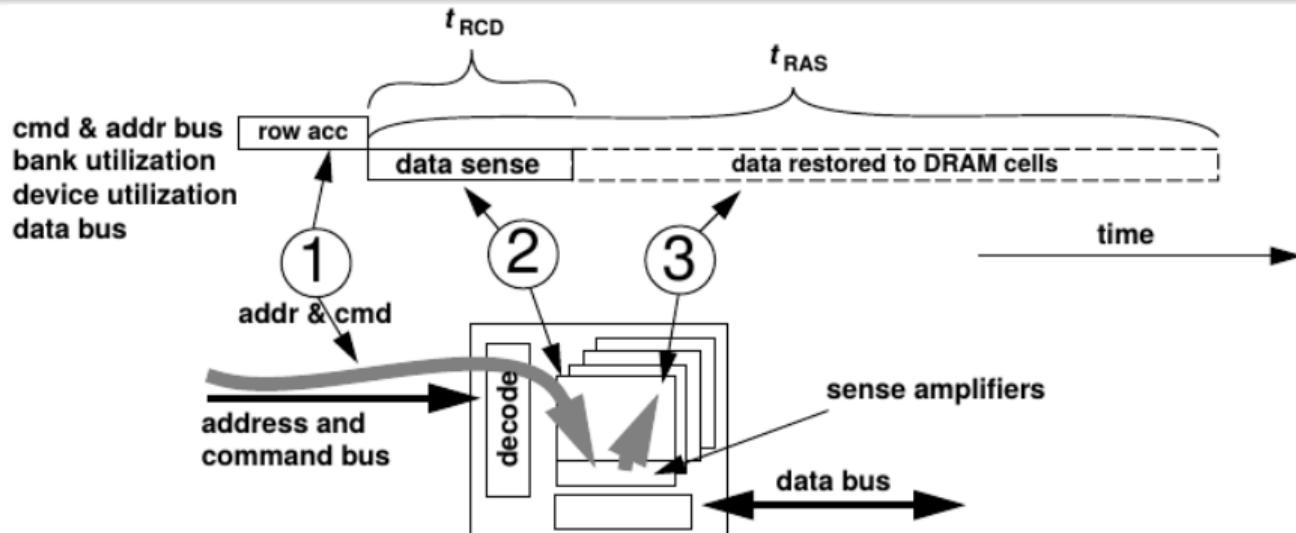


# Comando Row Access: Formato y timing.



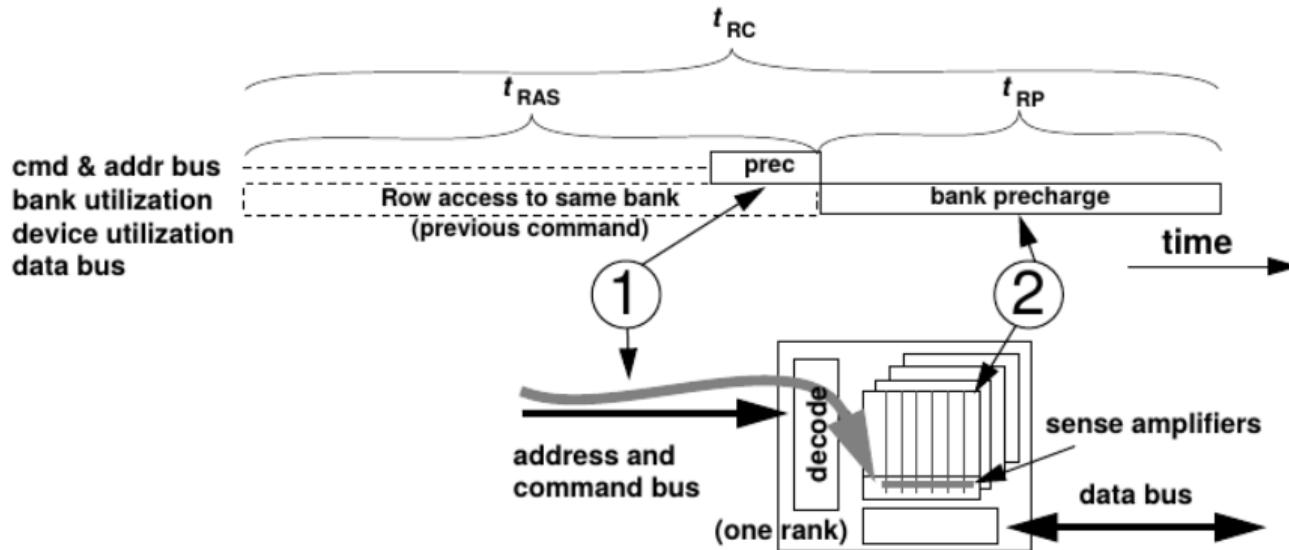
- Luego de  $t_{RCD}$  (**Row to Column command Delay**) los datos están listos en los amplificadores de detección.

# Comando Row Access: Formato y timing.

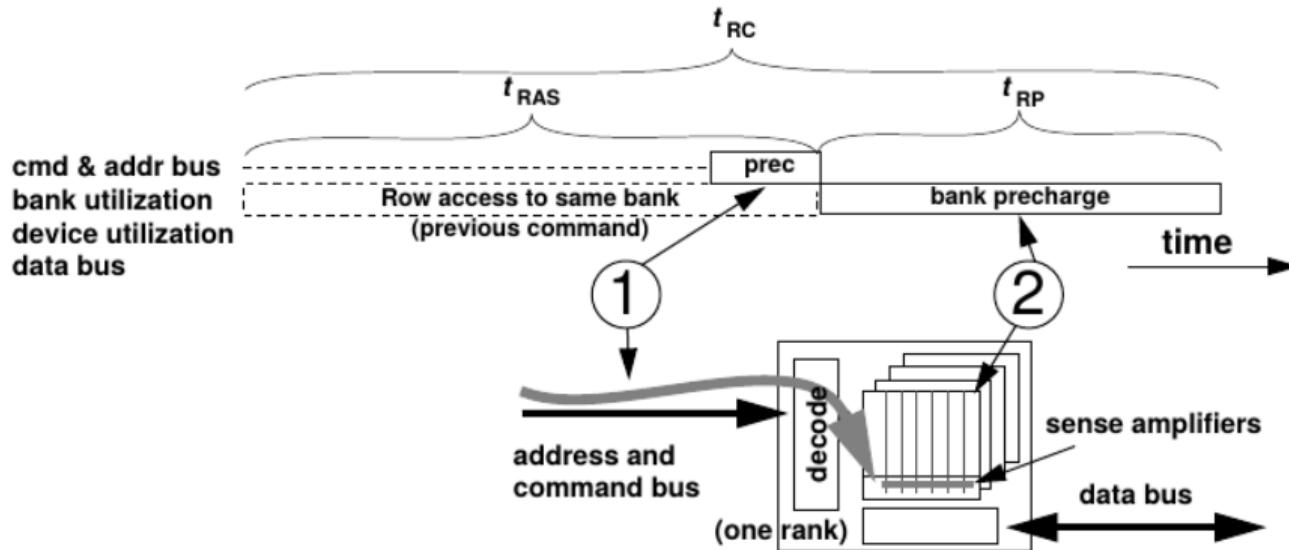


- Luego de  $t_{RCD}$  (**Row to Column command Delay**) los datos están listos en los amplificadores de detección.
- El tiempo total entre el comando Row Access y la realimentación de los datos en el DRAM array es  $t_{RAS}$  (**Row Access Strobe latency**). Un banco DRAM no se puede precargar al menos hasta el último  $t_{RAS}$  posterior a la activación previa del banco.

# Comando Precharge: Formato y timing.

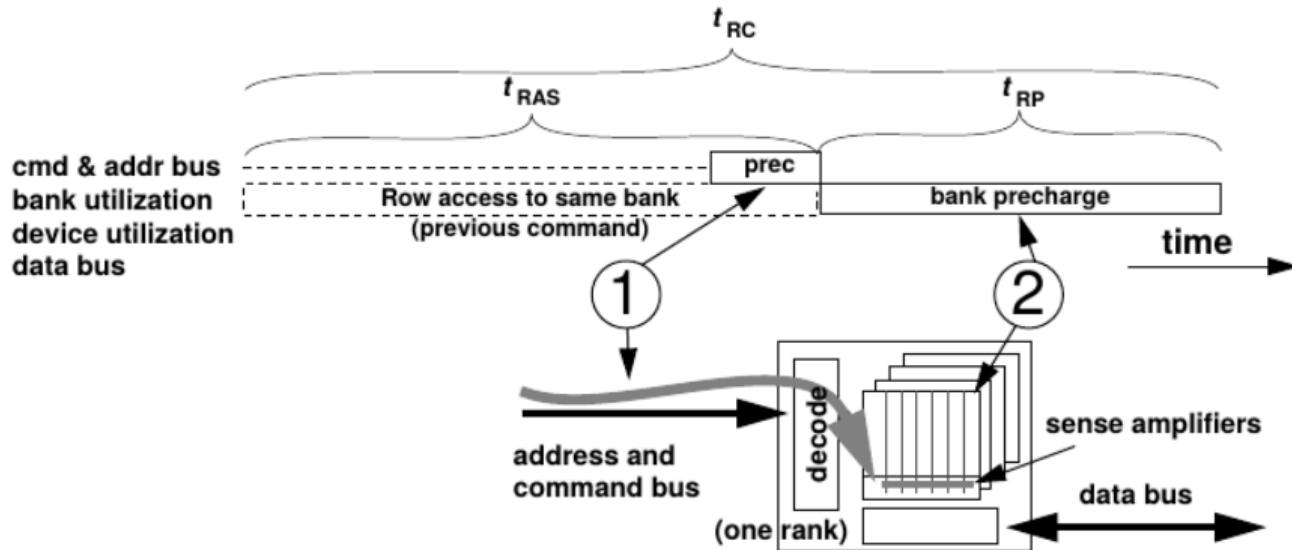


# Comando Precharge: Formato y timing.



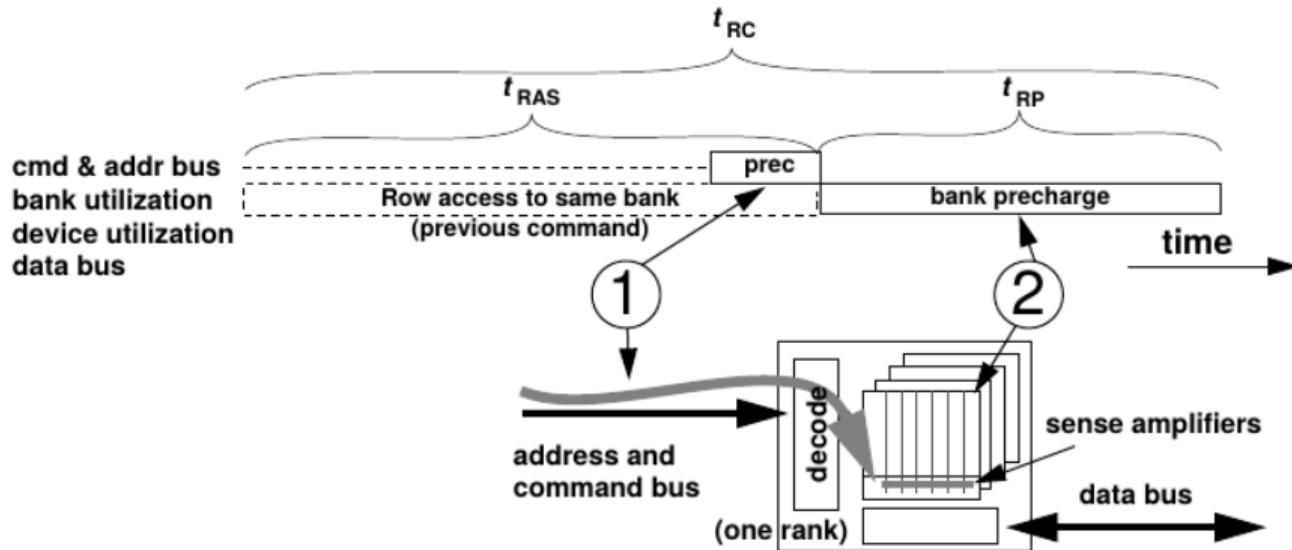
- El acceso de datos en una DRAM es un proceso de dos pasos:

# Comando Precharge: Formato y timing.



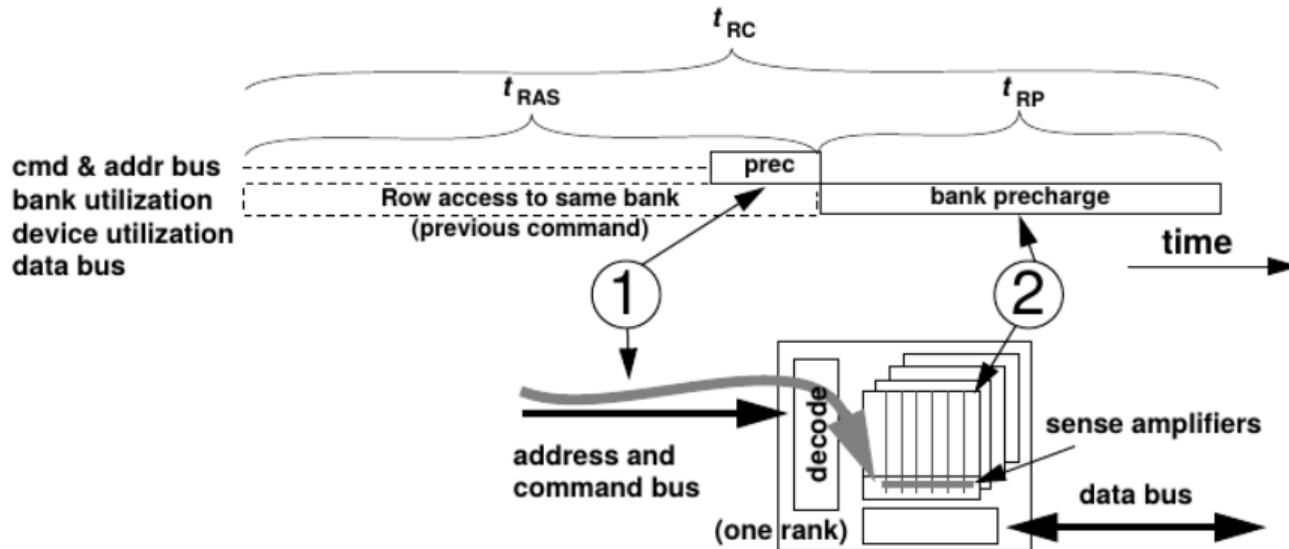
- El acceso de datos en una DRAM es un proceso de dos pasos:
  - Un comando *Row Access* para mover los datos desde el array de celdas de memoria al array de amplificadores de detección.

# Comando Precharge: Formato y timing.



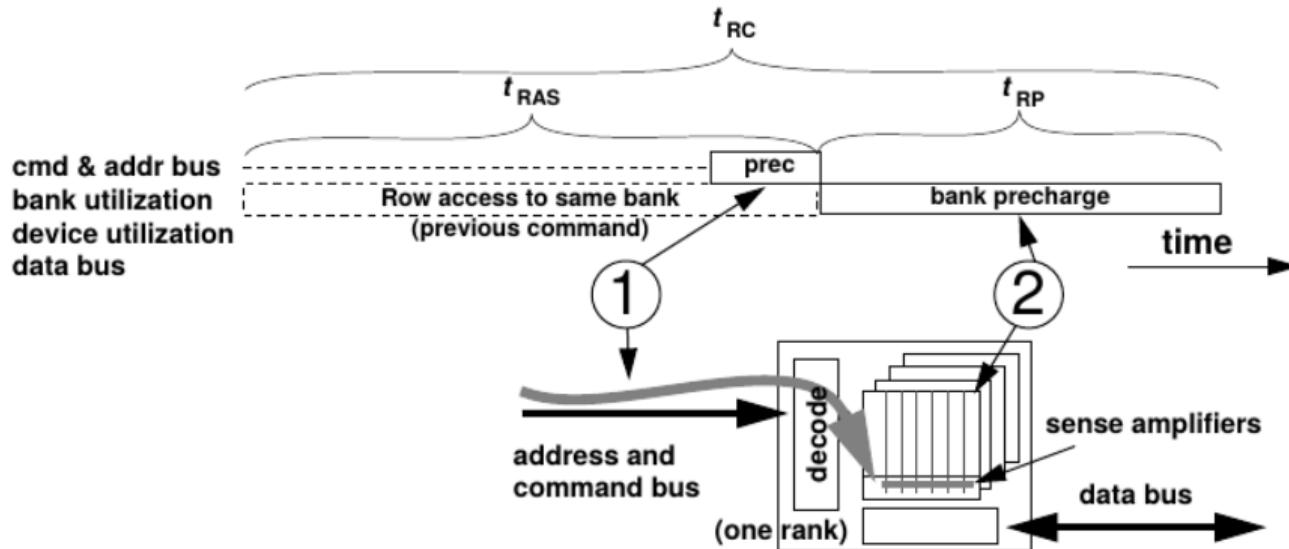
- El acceso de datos en una DRAM es un proceso de dos pasos:
  - Un comando *Row Access* para mover los datos desde el array de celdas de memoria al array de amplificadores de detección.
  - Se cachean los datos en el amplificador de detección para los *Column Access* posteriores

# Comando Precharge: Formato y timing.



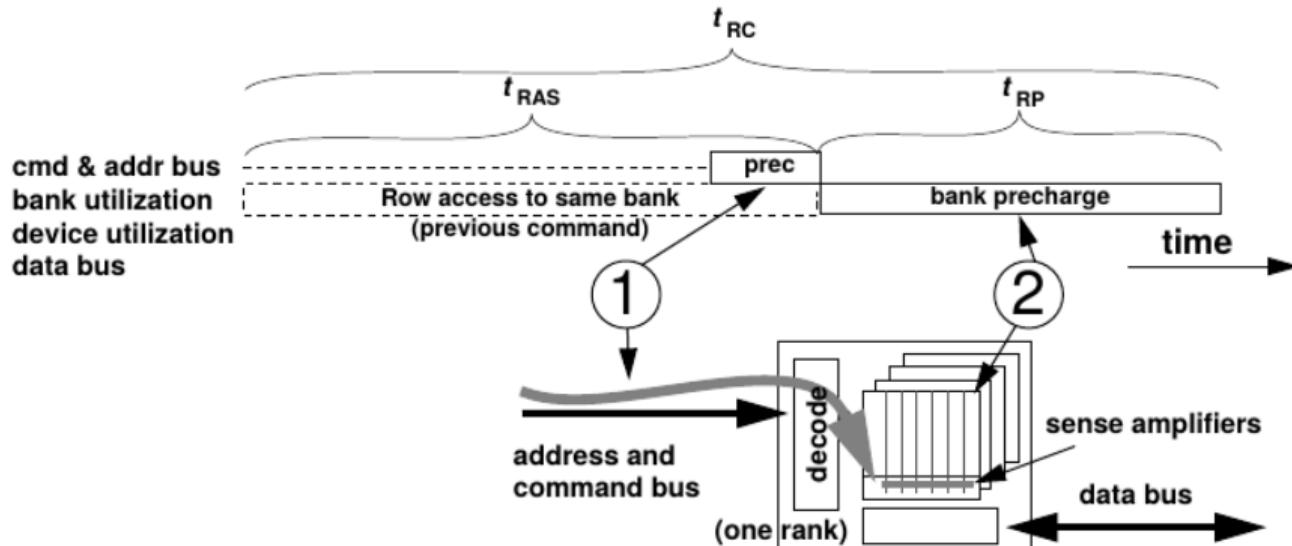
- **Precharge**, completa la secuencia de *Row Access*, reseteando los amplificadores de detección y las bitlines de modo de dejarlas listas para un nuevo *Row Access* al mismo array de celdas.

# Comando Precharge: Formato y timing.



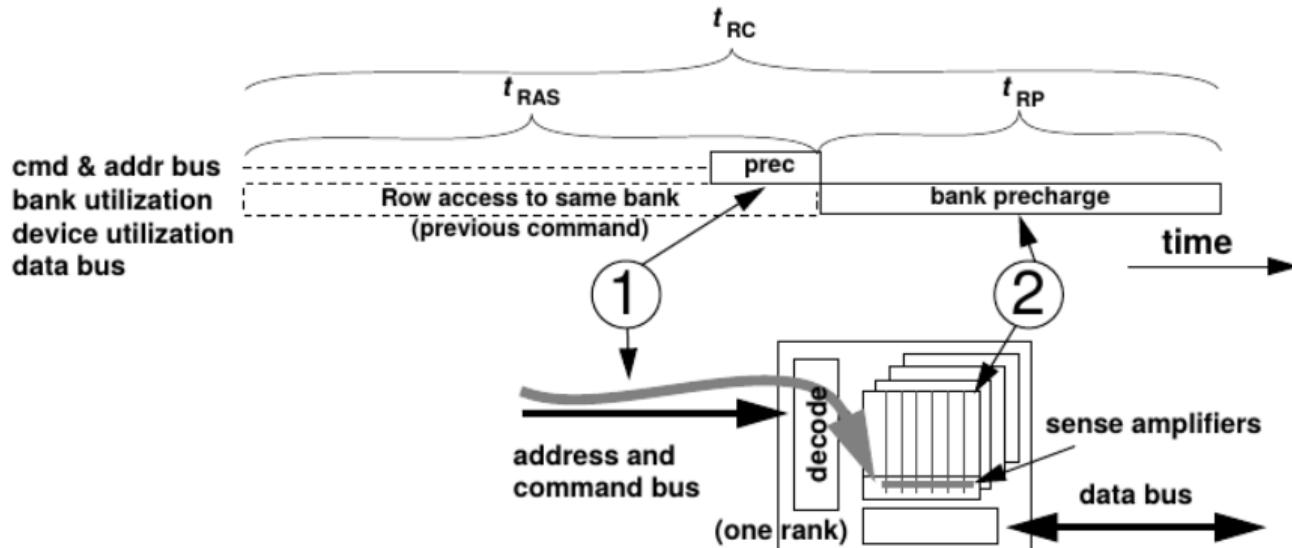
- *Precharge*, completa la secuencia de *Row Access*, reseteando los amplificadores de detección y las bitlines de modo de dejarlas listas para un nuevo *Row Access* al mismo array de celdas.
- El tiempo que toma precargar un DRAM array para otro *Row Access* es  $t_{RP}$  (**Row Precharge**).

# Comando Precharge: Formato y timing.



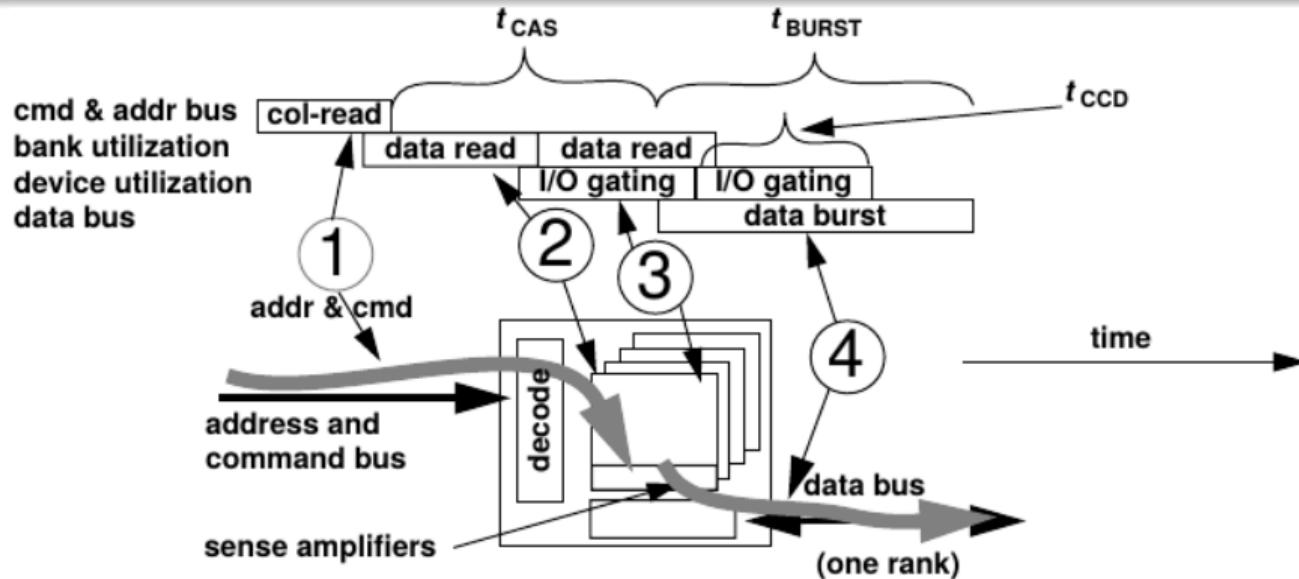
- $t_{RP}$  y  $t_{RAS}$  se combinan formando  $t_{RC}$  (**Row Cycle Time**), que indica el tiempo mínimo que necesita un DRAM device para poner disponibles en los amplificadores de detección los datos de los arrays de celdas, restaurar los datos de esas celdas, y precargar los bitlines al nivel de tensión de referencia para un nuevo comando Row Access.

# Comando Precharge: Formato y timing.

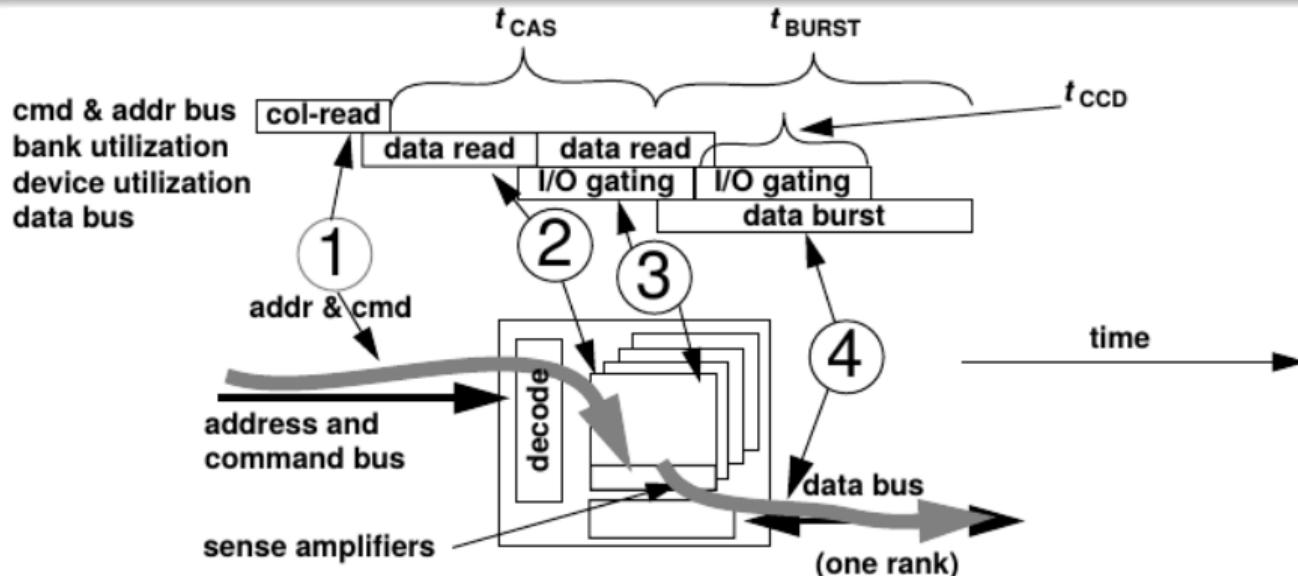


- $t_{RC}$  es la limitación fundamental de la velocidad a la que se pueden obtener datos de filas diferentes del mismo banco de DRAM. También se lo conoce como *random row-cycle time* de un DRAM device.

# Comando Column Read: Formato y timing

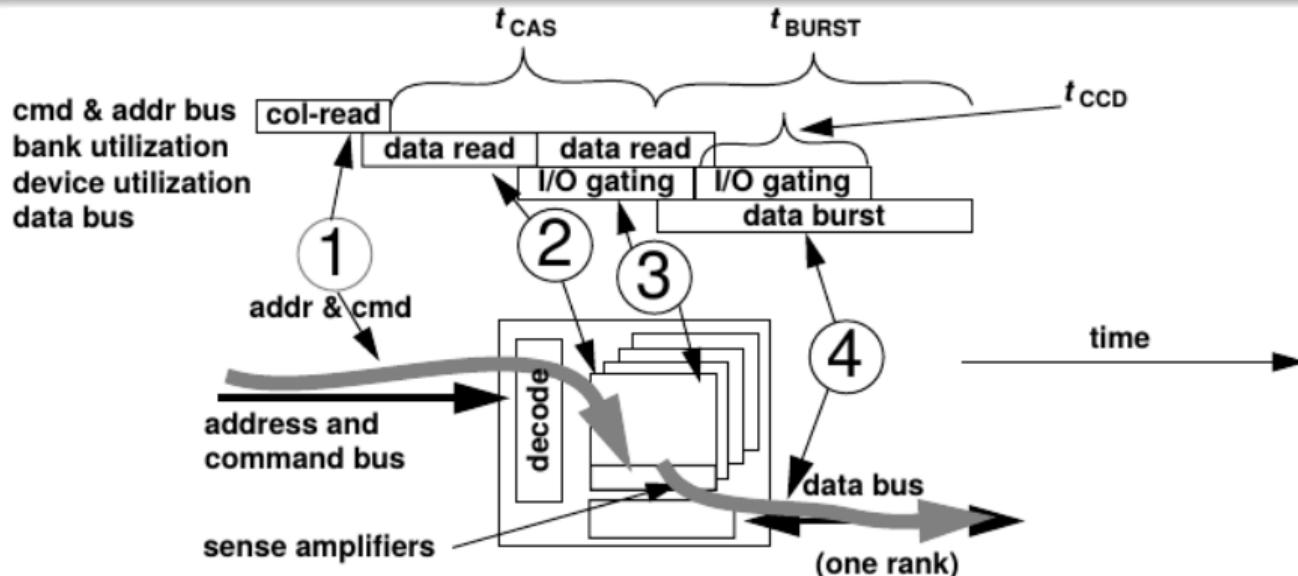


# Comando Column Read: Formato y timing



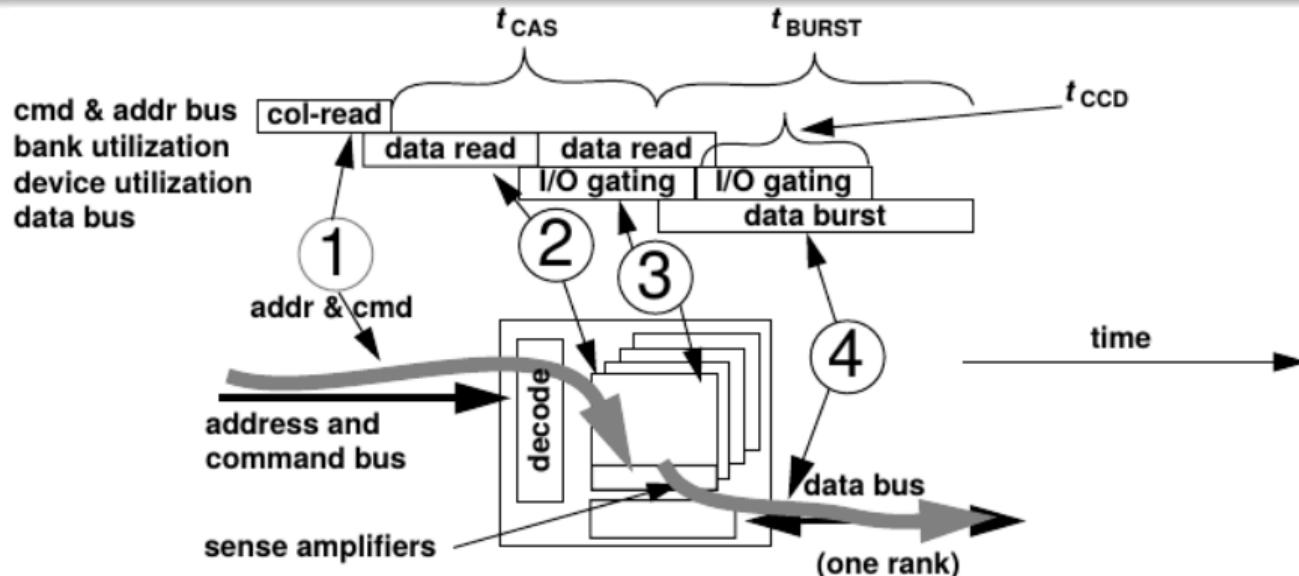
- Mueve los datos desde los amplificadores de detección hasta el controlador de memoria.

# Comando Column Read: Formato y timing



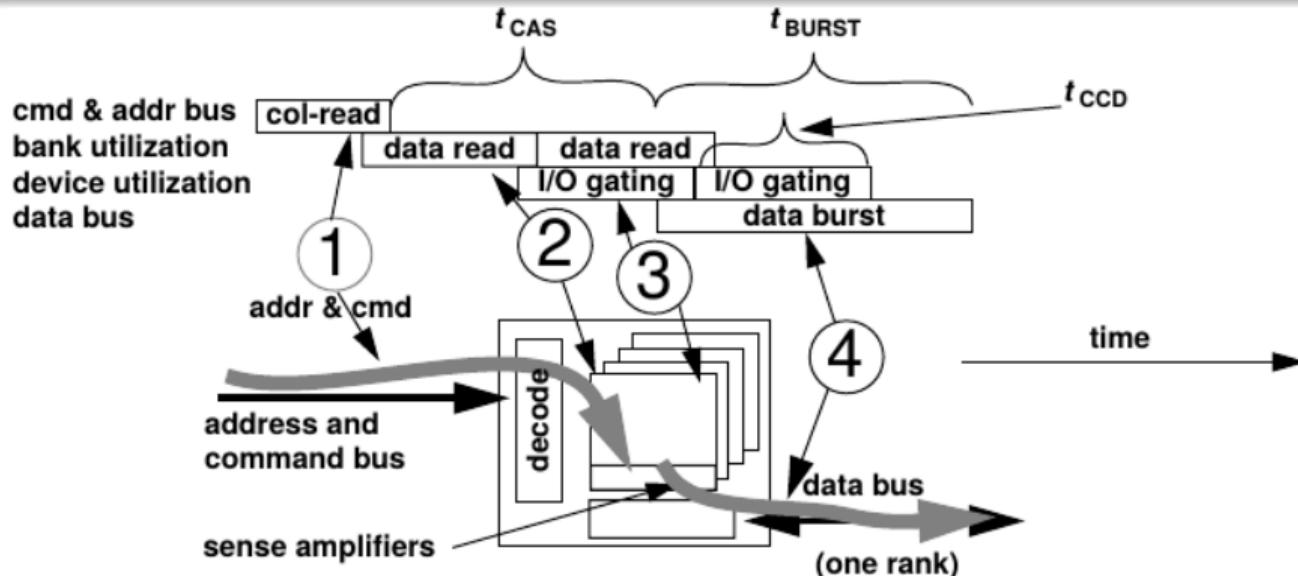
- Mueve los datos desde los amplificadores de detección hasta el controlador de memoria.
- $t_{CAS}$  **Column Access Strobe latency**, también llamado  $t_{CL}$ , es el tiempo que le demanda al DRAM device poner el dato en el bus luego de recibir el comando Column Read.

# Comando Column Read: Formato y timing



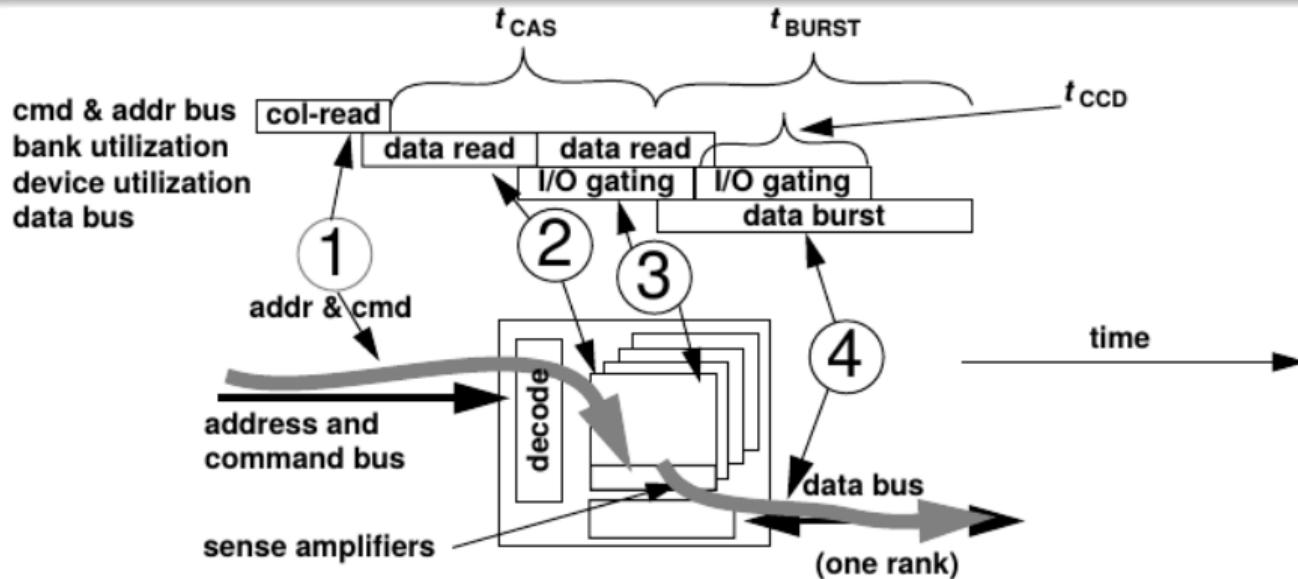
- Los DDR SDRAM devices mueven los datos internamente en pequeñas ráfagas (burst) a alta velocidad (n-prefetch).

# Comando Column Read: Formato y timing



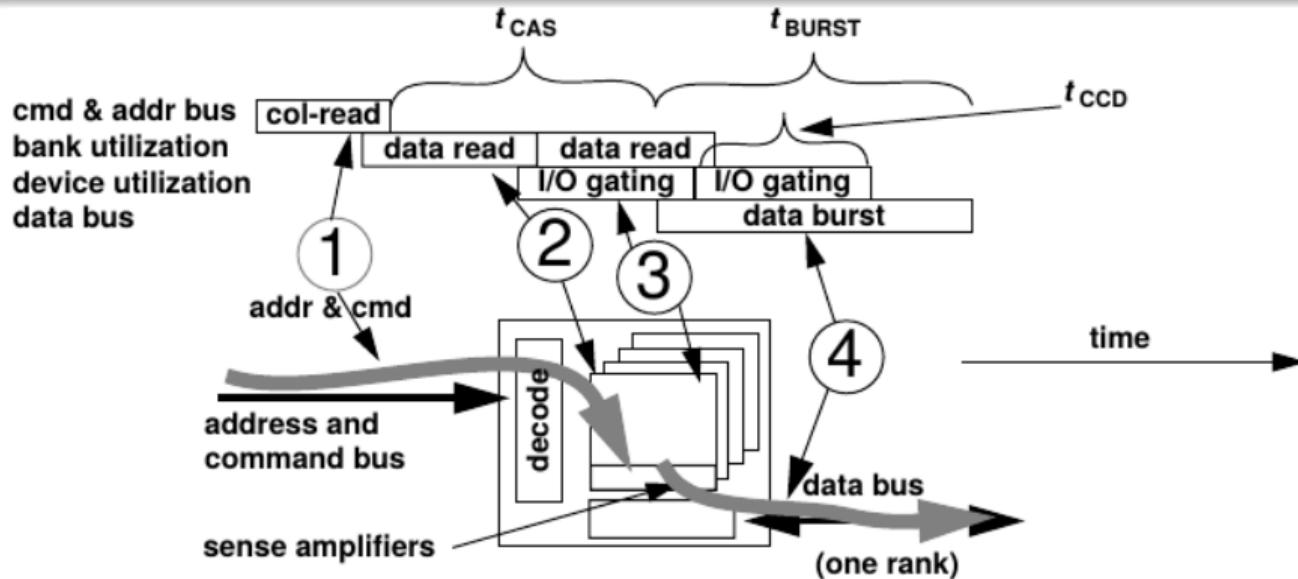
- Los DDR SDRAM devices mueven los datos internamente en pequeñas ráfagas (burst) a alta velocidad (n-prefetch).
- En el caso del gráfico se muestra un device que entrega datos en dos bursts rápidos (2n-prefetch), y los vuelca al bus de datos en un solo burst.

# Comando Column Read: Formato y timing



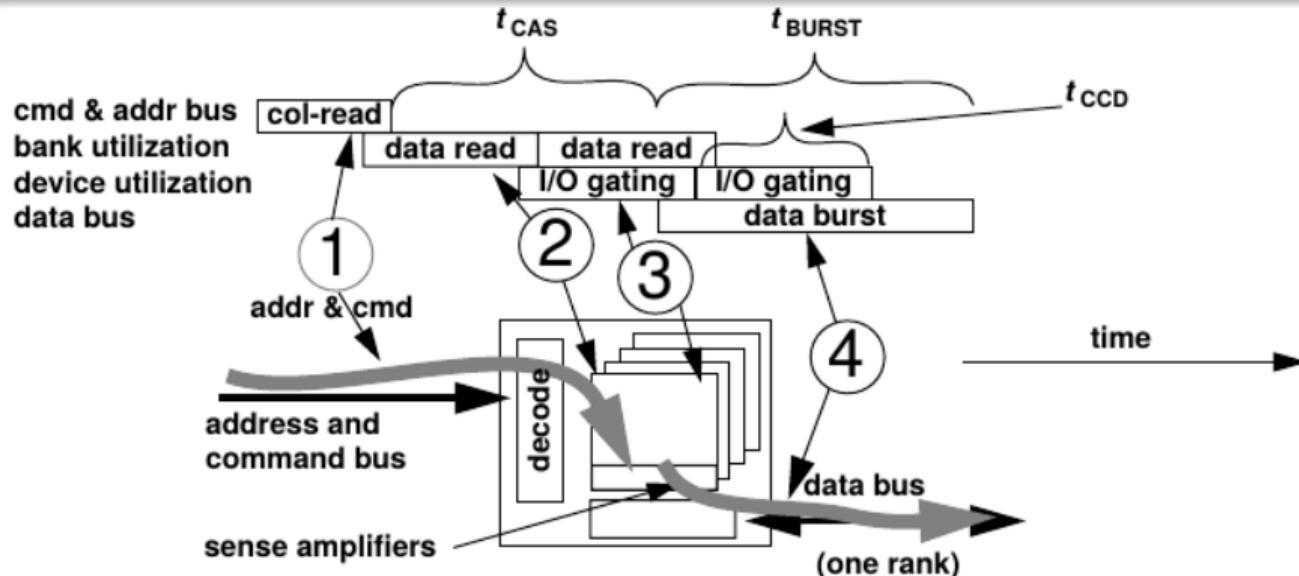
- $t_{CCD}$  **Column-to-Column Delay** es la duración de un pulso burst interno. Su valor mínimo está determinado por el valor de bits de prebúsqueda. En este ejemplo se usan 2 bits de prebúsqueda, por lo tanto,  $t_{CCD}$  será de un ciclo de clock.

# Comando Column Read: Formato y timing



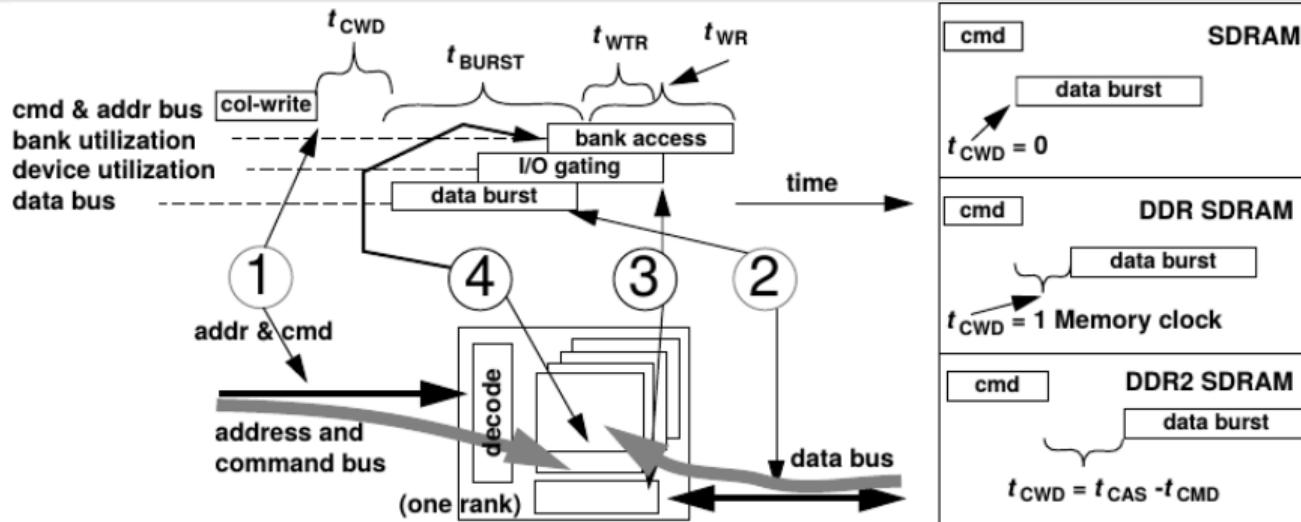
- La duración de un burst completo de datos en el bus es  $t_{BURST}$  **Data burst duration**.

# Comando Column Read: Formato y timing

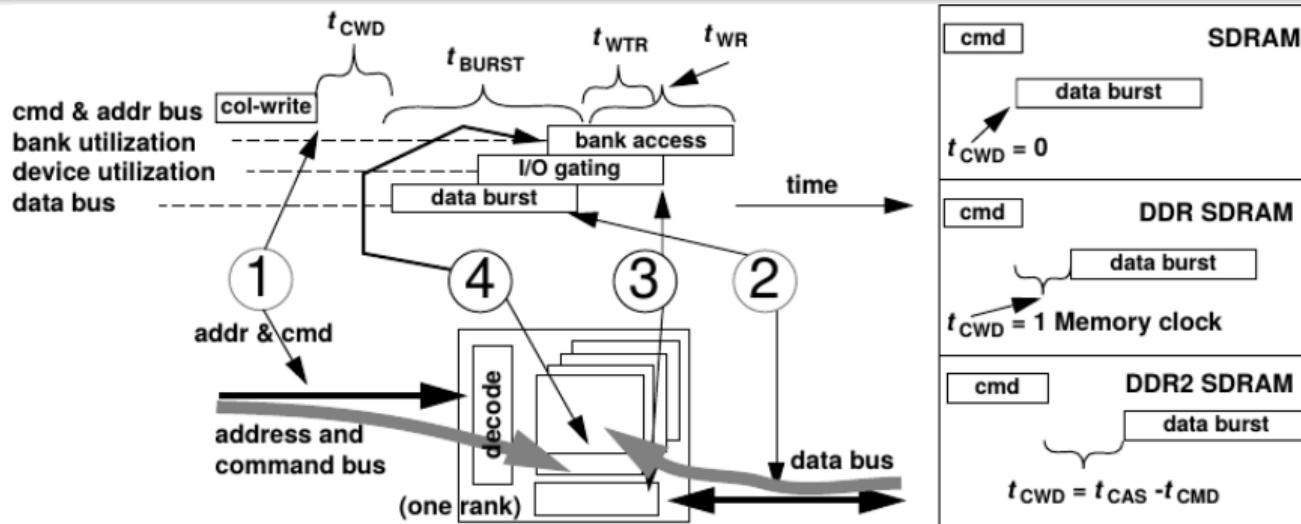


- La duración de un burst completo de datos en el bus es  $t_{BURST}$  **Data burst duration**.
- Usar bits de prebúsqueda mejora el tiempo total de Column Read si se logra:  $t_{CCD} < t_{BURST}$

# Comando Column Write: Formato y timing.

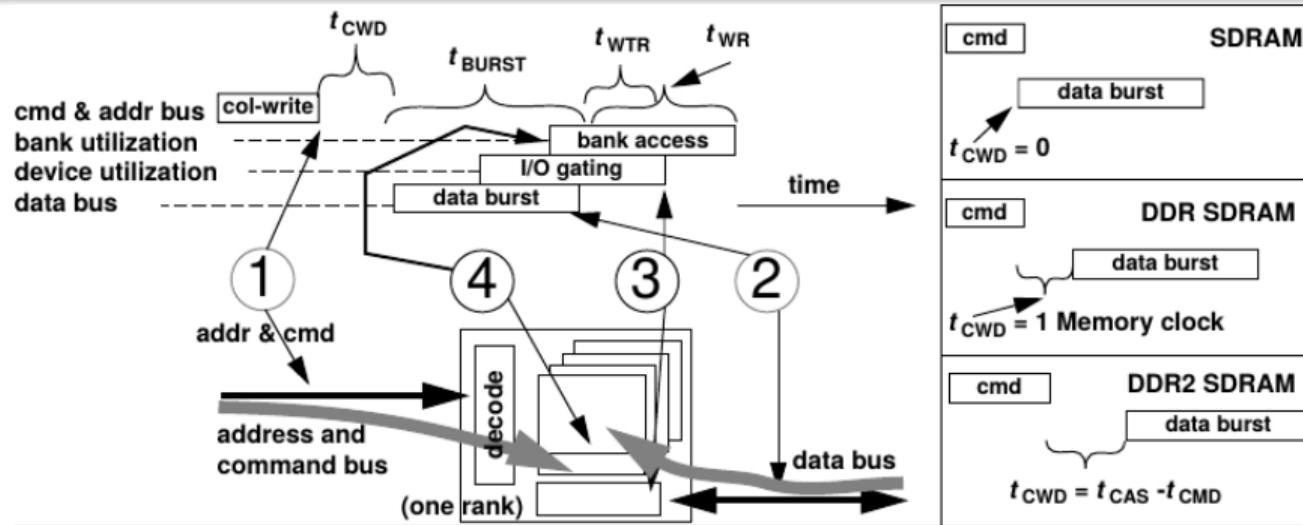


# Comando Column Write: Formato y timing.



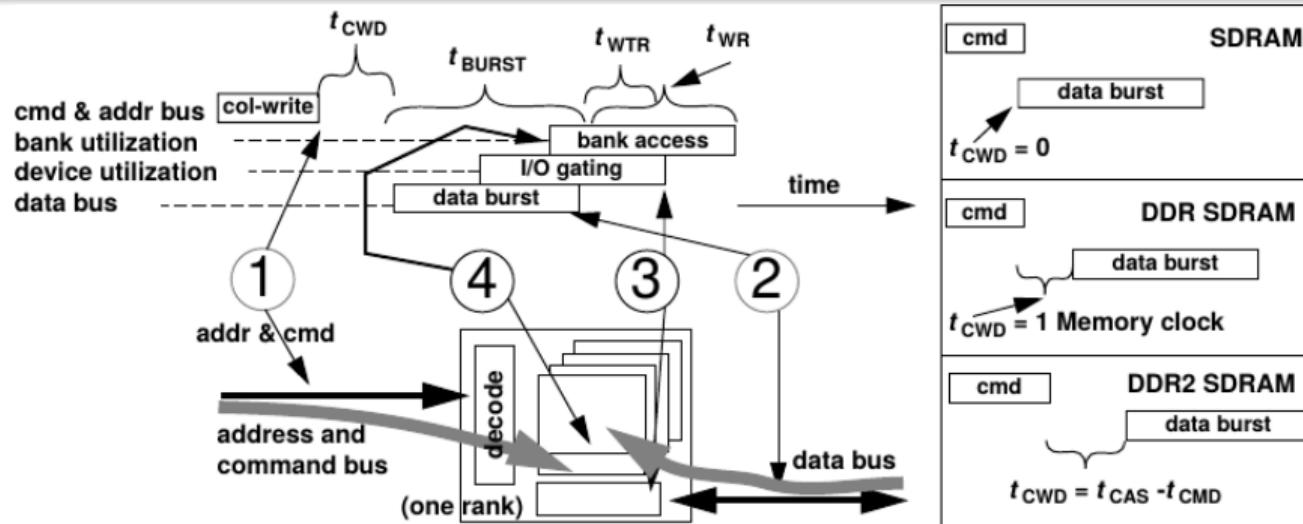
- Mueve los datos desde el controlador de memoria hasta los amplificadores de detección.

# Comando Column Write: Formato y timing.



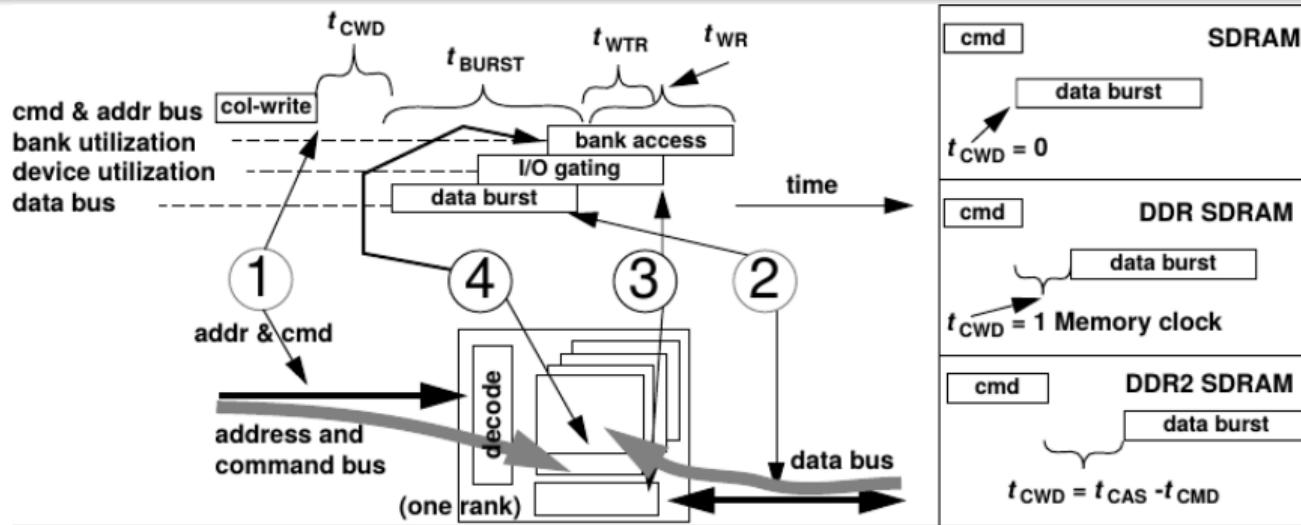
- Mueve los datos desde el controlador de memoria hasta los amplificadores de detección.
- El controlador de memoria pone el comando Column Write en el bus de comandos, y luego de un tiempo  $t_{CWD}$  (**Column Write Delay**), pone los datos en el bus de datos. El valor de este tiempo depende de la tecnología de memoria (lado derecho de la figura).

# Comando Column Write: Formato y timing.



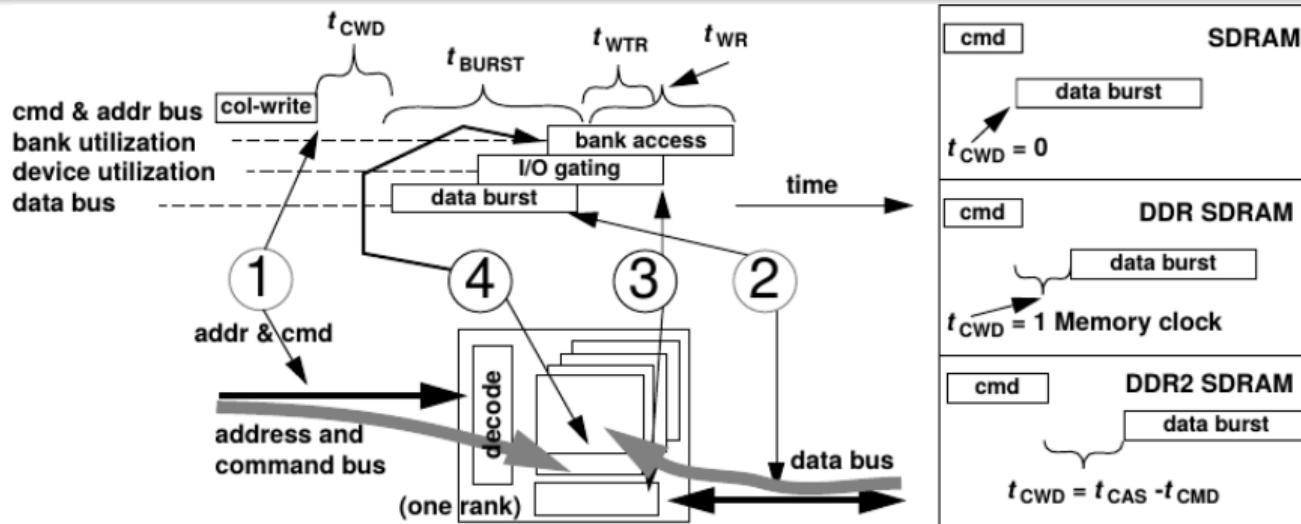
- La ráfaga (burst) completa de datos en el bus lleva un tiempo  $t_{BURST}$  (**Data burst duration**).

# Comando Column Write: Formato y timing.



- La ráfaga (burst) completa de datos en el bus lleva un tiempo  $t_{BURST}$  (**Data burst duration**).
- El DRAM device emplea un tiempo  $t_{WTR}$  (**Write To Read delay time**) en liberar los recursos de gating de E/S. Este tiempo debe ser especialmente tenido en cuenta cuando al comando Write lo sigue un comando Read.

# Comando Column Write: Formato y timing.



- Los datos tomados por el gating de E/S se propagan hasta los gate arrays luego de un tiempo  $t_{WR}$  (**Write Recovery time**). Este tiempo debe ser especialmente tenido en cuenta si al comando Write lo sigue un comando Precharge.

# Refresco de memoria

# Refresco de memoria

- Tiene por objeto asegurar la integridad de la información, ya que los capacitores inexorablemente encontrarán un camino de impedancia (alta pero no infinita) a través del cual descargarse.

## Refresco de memoria

- Tiene por objeto asegurar la integridad de la información, ya que los capacitores inexorablemente encontrarán un camino de impedancia (alta pero no infinita) a través del cual descargarse.
- Refrescar implica leer para regenerar la carga. Esto consume ancho de banda en el bus que une el controlador y la memoria DRAM.

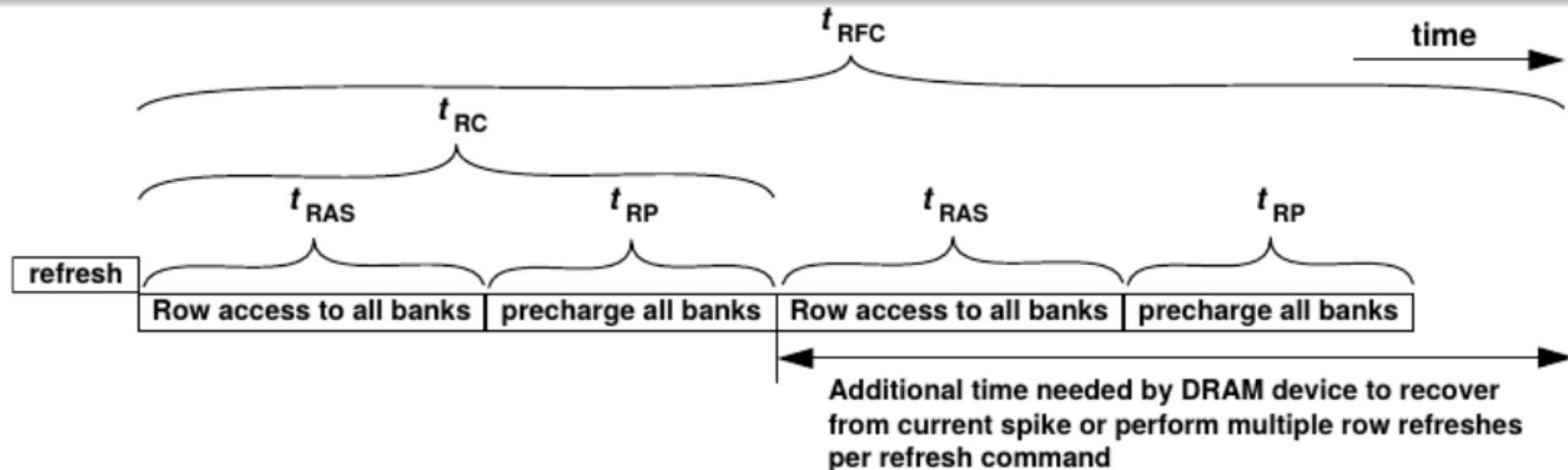
## Refresco de memoria

- Tiene por objeto asegurar la integridad de la información, ya que los capacitores inexorablemente encontrarán un camino de impedancia (alta pero no infinita) a través del cual descargarse.
- Refrescar implica leer para regenerar la carga. Esto consume ancho de banda en el bus que une el controlador y la memoria DRAM.
- Algunos controladores son diseñados para minimizar la complejidad del mecanismo de refresco, de memoria, otros para optimizar el uso de ancho de banda del bus en esta operación, y otros intentan minimizar el consumo de energía consumida en esta operación

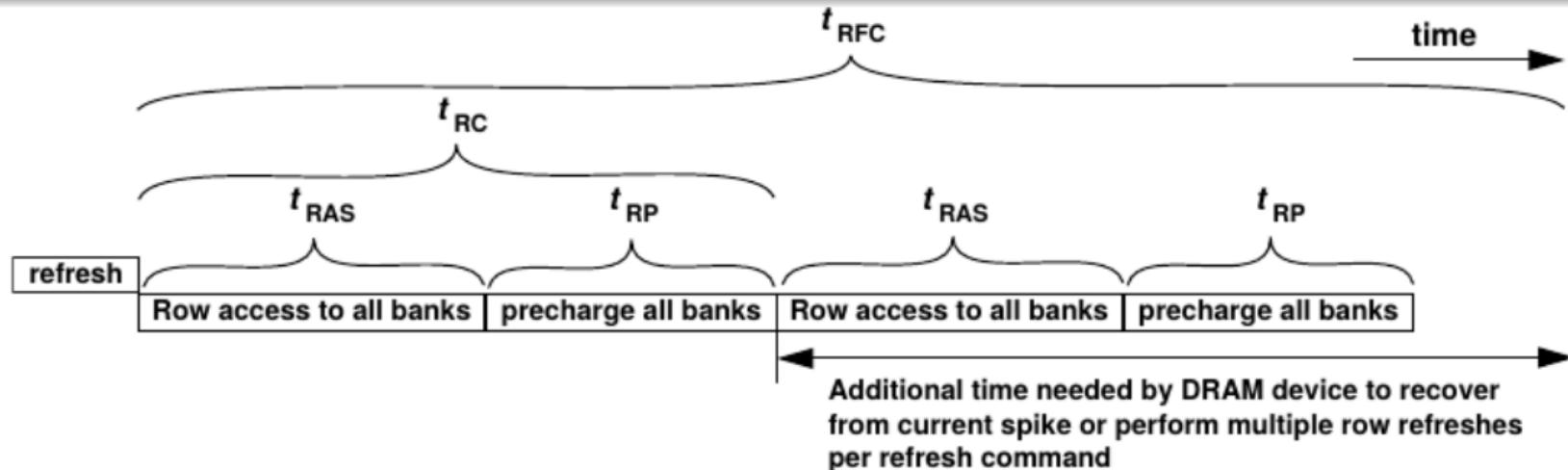
## Refresco de memoria

- Tiene por objeto asegurar la integridad de la información, ya que los capacitores inexorablemente encontrarán un camino de impedancia (alta pero no infinita) a través del cual descargarse.
- Refrescar implica leer para regenerar la carga. Esto consume ancho de banda en el bus que une el controlador y la memoria DRAM.
- Algunos controladores son diseñados para minimizar la complejidad del mecanismo de refresco, de memoria, otros para optimizar el uso de ancho de banda del bus en esta operación, y otros intentan minimizar el consumo de energía consumida en esta operación
- Por lo general todos los controladores contienen un Refresh Row Register al que utilizan para refrescar una fila completa de celdas con un solo comando.

# Refresco de memoria

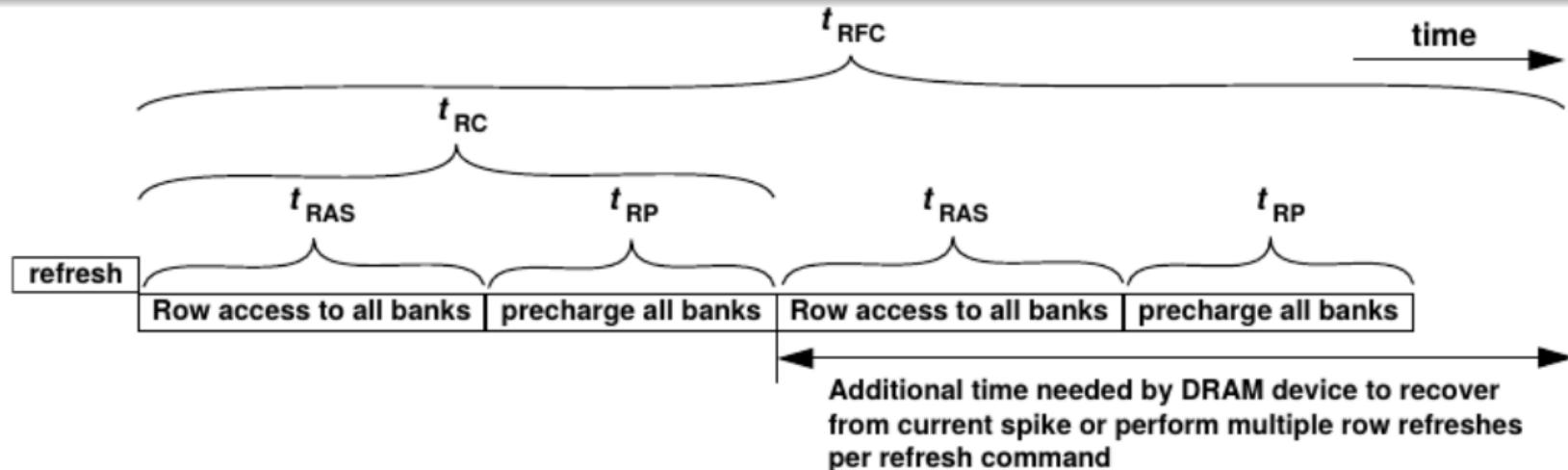


# Refresco de memoria



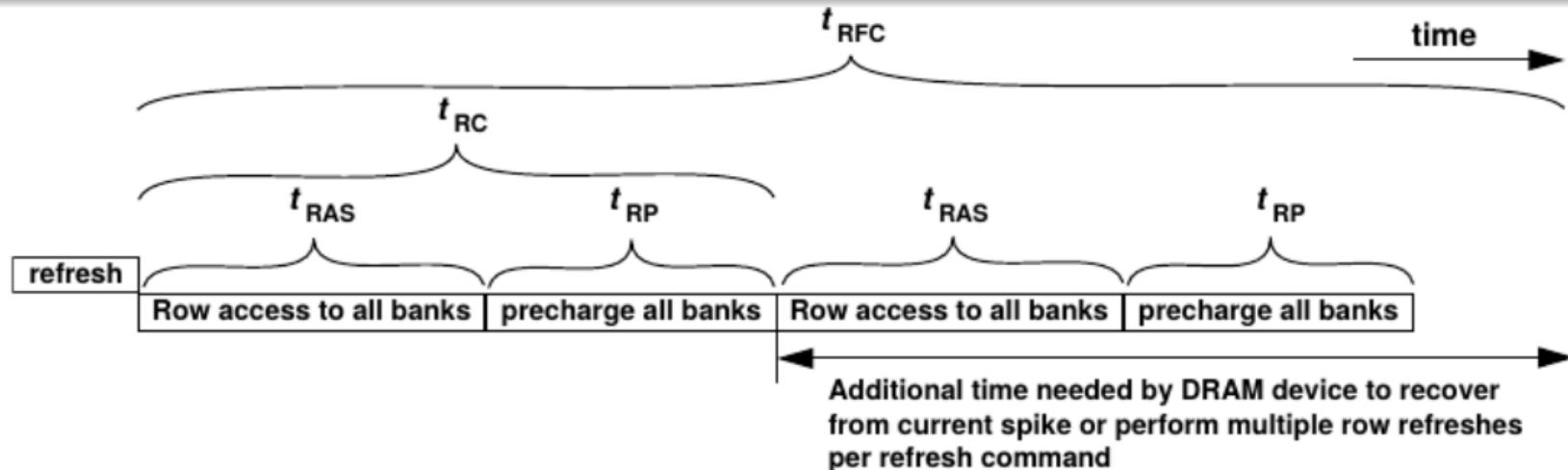
- El Refresh Row Register contiene siempre la Row Address de la última fila accedida.

# Refresco de memoria



- El Refresh Row Register contiene siempre la Row Address de la última fila accedida.
- Cada vez que el Controlador de Memoria envía un comando Refresh, el DRAM Device incrementa este registro e inicia un row cycle en cada fila de cada banco cuya dirección coincide con la del Refresh Row Register,

# Refresco de memoria



- El Refresh Row Register contiene siempre la Row Address de la última fila accedida.
- Cada vez que el Controlador de Memoria envía un comando Refresh, el DRAM Device incrementa este registro e inicia un row cycle en cada fila de cada banco cuya dirección coincide con la del Refresh Row Register,
- $t_{RFC}$  **Refresh Cycle time**. Intervalo de tiempo entre los comandos de Refresco y activación.

# Formato y timing de un comando Refresh

Familia DRAM	Tensión	Capacidad	# bancos	# Filas	Tamaño Fila	Cuenta de Refresco	$t_{RC}$	$t_{RFC}$
DDR	2.5V	256Mb	4	8192	1kB	8192	60 ns	67 ns
		512Mb	4	8192	2kB	8192	55 ns	70 ns
DDR2	1.8V	256Mb	4	8192	1kB	8192	55 ns	75 ns
		512Mb	4	16384	1kB	8192	5 ns	105 ns
		1024Mb	8	16384	1kB	8192	54 ns	127.5 ns
		2048Mb	8	32768	1kB	8192	-	197.5 ns
		4096Mb	8	65536	1kB	8192	-	327.5 ns

# Formato y timing de un comando Refresh

Familia DRAM	Tensión	Capacidad	# bancos	# Filas	Tamaño Fila	Cuenta de Refresco	$t_{RC}$	$t_{RFC}$
DDR	2.5V	256Mb	4	8192	1kB	8192	60 ns	67 ns
		512Mb	4	8192	2kB	8192	55 ns	70 ns
DDR2	1.8V	256Mb	4	8192	1kB	8192	55 ns	75 ns
		512Mb	4	16384	1kB	8192	5 ns	105 ns
		1024Mb	8	16384	1kB	8192	54 ns	127.5 ns
		2048Mb	8	32768	1kB	8192	-	197.5 ns
		4096Mb	8	65536	1kB	8192	-	327.5 ns

- La tabla muestra tendencias en los tiempos de refresco.

# Formato y timing de un comando Refresh

Familia DRAM	Tensión	Capacidad	# bancos	# Filas	Tamaño Fila	Cuenta de Refresco	$t_{RC}$	$t_{RFC}$
DDR	2.5V	256Mb	4	8192	1kB	8192	60 ns	67 ns
		512Mb	4	8192	2kB	8192	55 ns	70 ns
DDR2	1.8V	256Mb	4	8192	1kB	8192	55 ns	75 ns
		512Mb	4	16384	1kB	8192	5 ns	105 ns
		1024Mb	8	16384	1kB	8192	54 ns	127.5 ns
		2048Mb	8	32768	1kB	8192	-	197.5 ns
		4096Mb	8	65536	1kB	8192	-	327.5 ns

- La tabla muestra tendencias en los tiempos de refresco.
- A mayor capacidad del banco mayor cantidad de celdas, mayor tiempo de refresco.

# Formato y timing de un comando Refresh

Familia DRAM	Tensión	Capacidad	# bancos	# Filas	Tamaño Fila	Cuenta de Refresco	$t_{RC}$	$t_{RFC}$
DDR	2.5V	256Mb	4	8192	1kB	8192	60 ns	67 ns
		512Mb	4	8192	2kB	8192	55 ns	70 ns
DDR2	1.8V	256Mb	4	8192	1kB	8192	55 ns	75 ns
		512Mb	4	16384	1kB	8192	5 ns	105 ns
		1024Mb	8	16384	1kB	8192	54 ns	127.5 ns
		2048Mb	8	32768	1kB	8192	-	197.5 ns
		4096Mb	8	65536	1kB	8192	-	327.5 ns

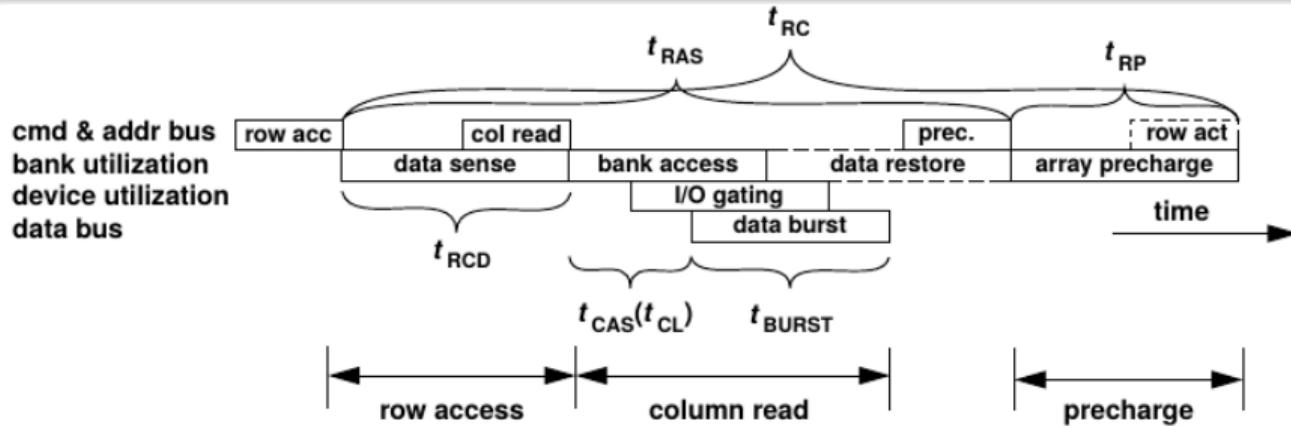
- La tabla muestra tendencias en los tiempos de refresco.
- A mayor capacidad del banco mayor cantidad de celdas, mayor tiempo de refresco.
- Da la impresión que de DDR2 en adelante el challenge es mantener el número de comandos de refresco en períodos de 64 mseg a pesar de duplicarse en número de filas. Esto implica para el controlador de memoria enviar 8192 comandos Refresh en 64 mseg.

# Formato y timing de un comando Refresh

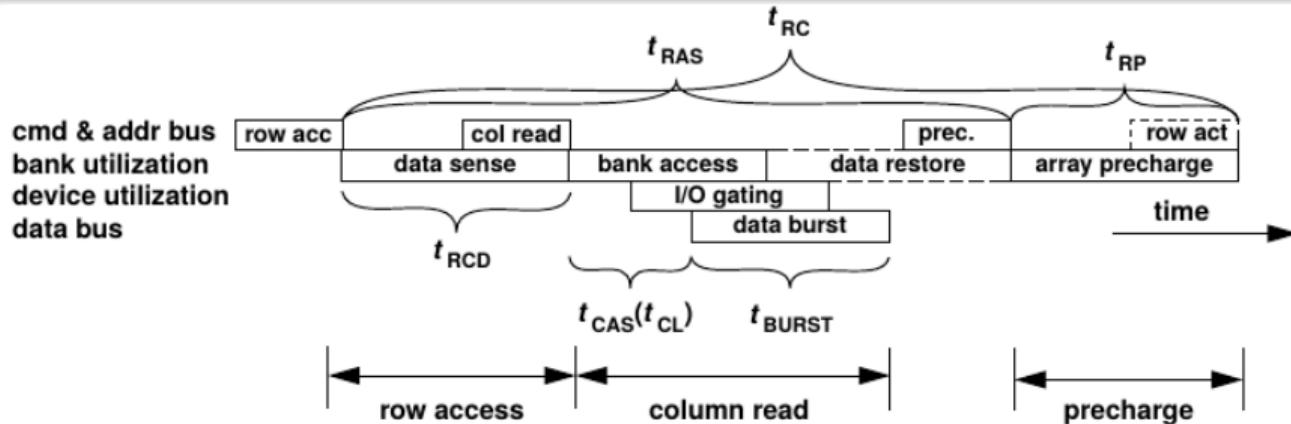
Familia DRAM	Tensión	Capacidad	# bancos	# Filas	Tamaño Fila	Cuenta de Refresco	$t_{RC}$	$t_{RFC}$
DDR	2.5V	256Mb	4	8192	1kB	8192	60 ns	67 ns
		512Mb	4	8192	2kB	8192	55 ns	70 ns
DDR2	1.8V	256Mb	4	8192	1kB	8192	55 ns	75 ns
		512Mb	4	16384	1kB	8192	5 ns	105 ns
		1024Mb	8	16384	1kB	8192	54 ns	127.5 ns
		2048Mb	8	32768	1kB	8192	-	197.5 ns
		4096Mb	8	65536	1kB	8192	-	327.5 ns

- La tabla muestra tendencias en los tiempos de refresco.
- A mayor capacidad del banco mayor cantidad de celdas, mayor tiempo de refresco.
- Da la impresión que de DDR2 en adelante el challenge es mantener el número de comandos de refresco en períodos de 64 mseg a pesar de duplicarse en número de filas. Esto implica para el controlador de memoria enviar 8192 comandos Refresh en 64 mseg.
- Los devices de alta capacidad tienen mas de 8192 filas con 2, 4, u 8 en filas en cada device.

# Ciclo de Lectura

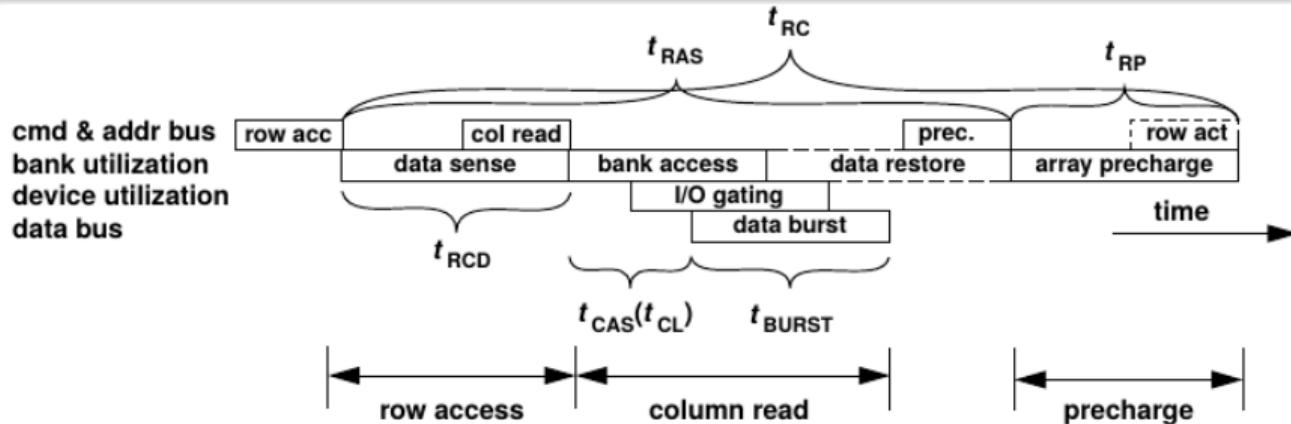


# Ciclo de Lectura



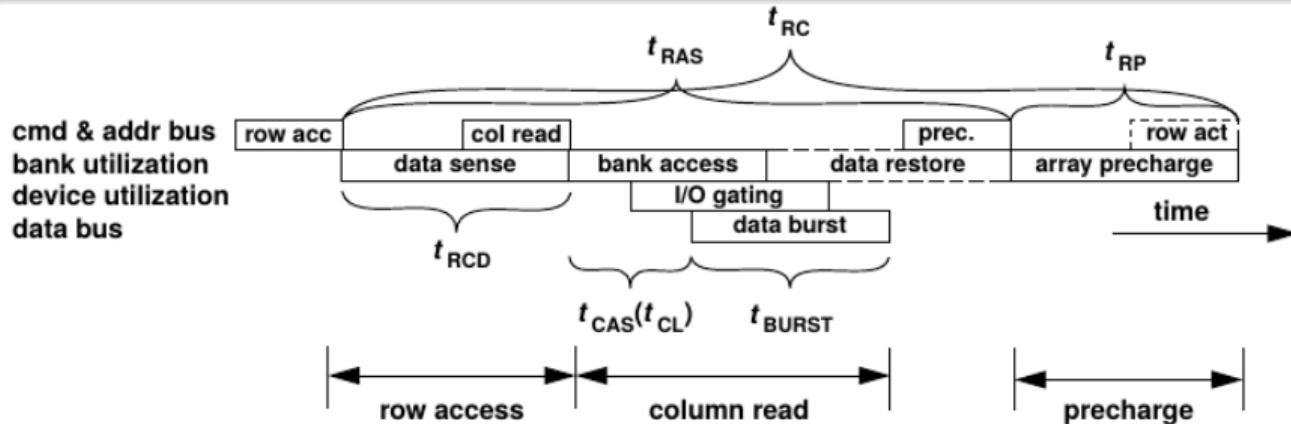
- El comando Row Access habilita muchos cientos de bits de datos al array de amplificadores de detección dentro de un banco.

# Ciclo de Lectura



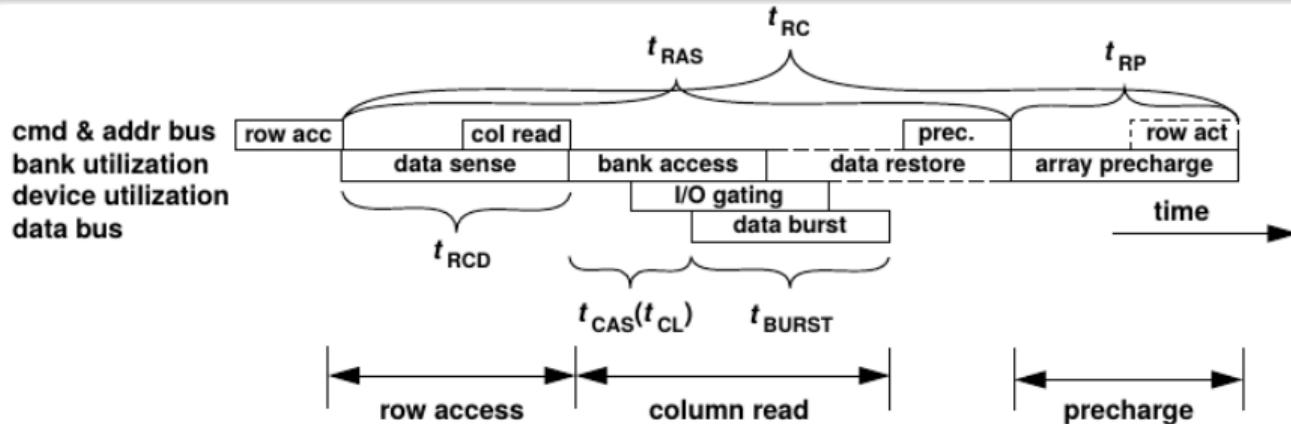
- El comando Row Access habilita muchos cientos de bits de datos al array de amplificadores de detección dentro de un banco.
- Luego Column Access selecciona decenas o algunos pocos cientos de esos bits.

# Ciclo de Lectura



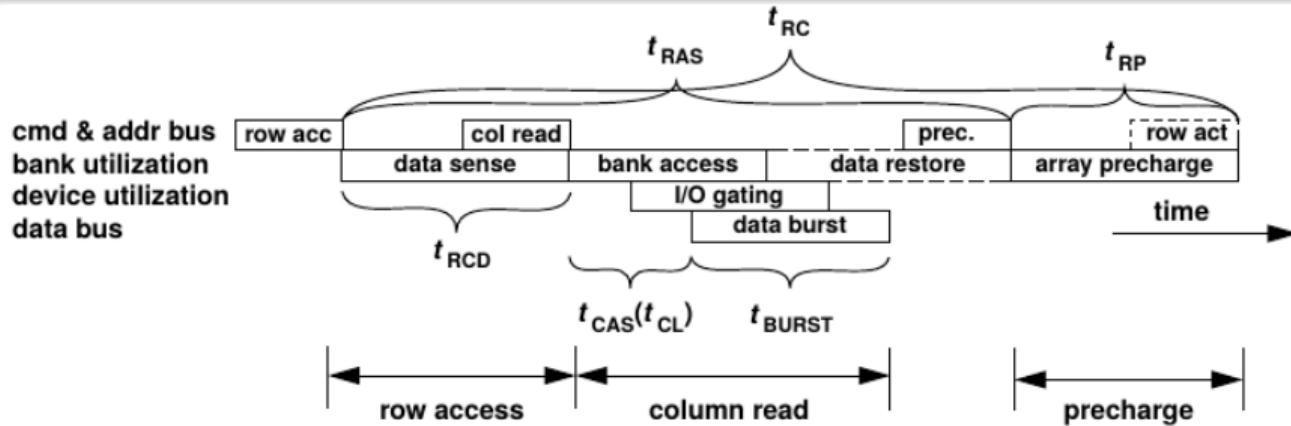
- El comando Row Access habilita muchos cientos de bits de datos al array de amplificadores de detección dentro de un banco.
- Luego Column Access selecciona decenas o algunos pocos cientos de esos bits.
- Si el acceso a memoria se efectúa en direcciones secuenciales el device no necesita precargar nuevamente la fila, con el consiguiente ahorro de tiempo de acceso y de energía.

# Ciclo de Lectura



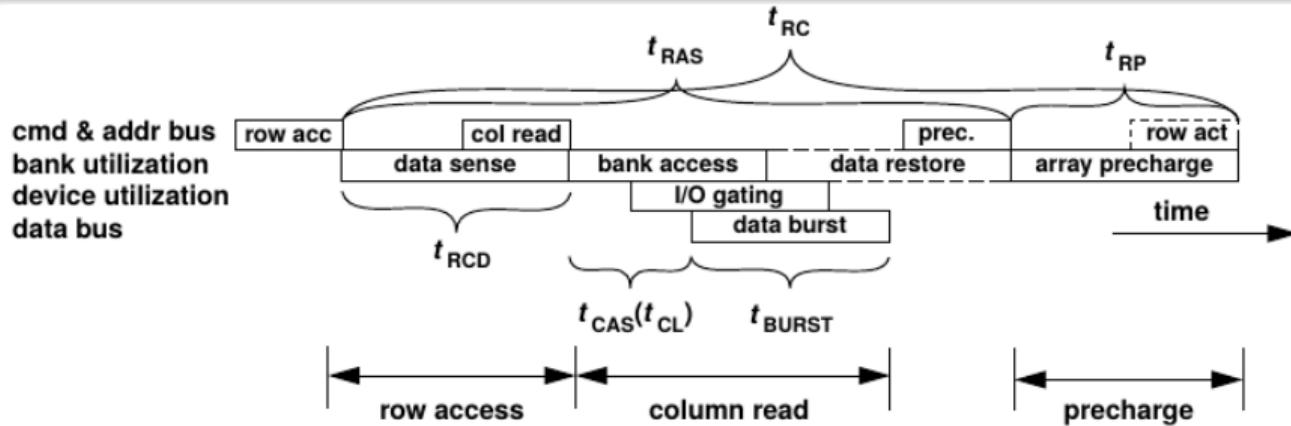
- Nuevamente el principio de vecindad en el acceso a memoria regula la eficiencia del sistema.

# Ciclo de Lectura



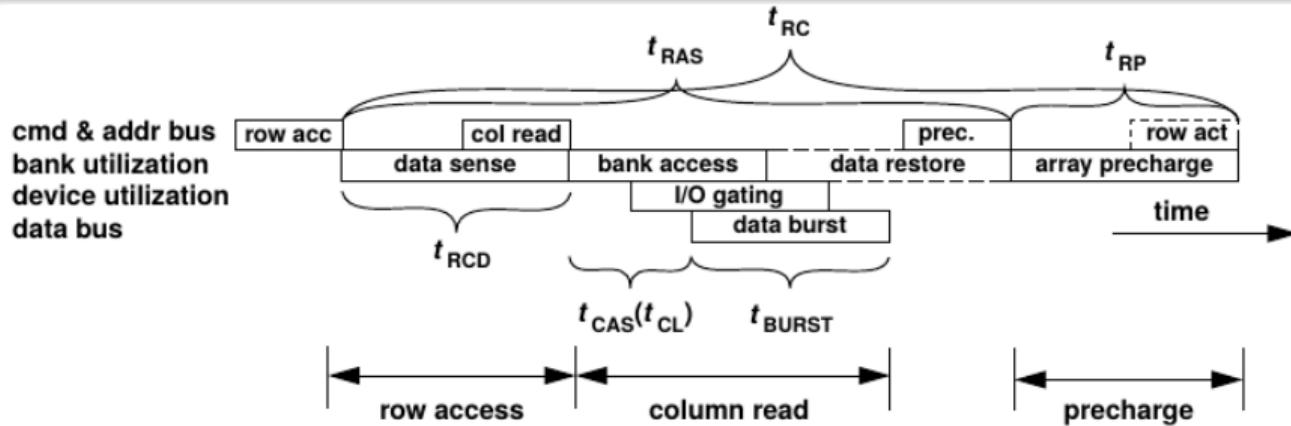
- Nuevamente el principio de vecindad en el acceso a memoria regula la eficiencia del sistema.
- Transcurrido  $t_{RCD}$  los datos de la fila seleccionada son precargados por los amplificadores de detección.

# Ciclo de Lectura



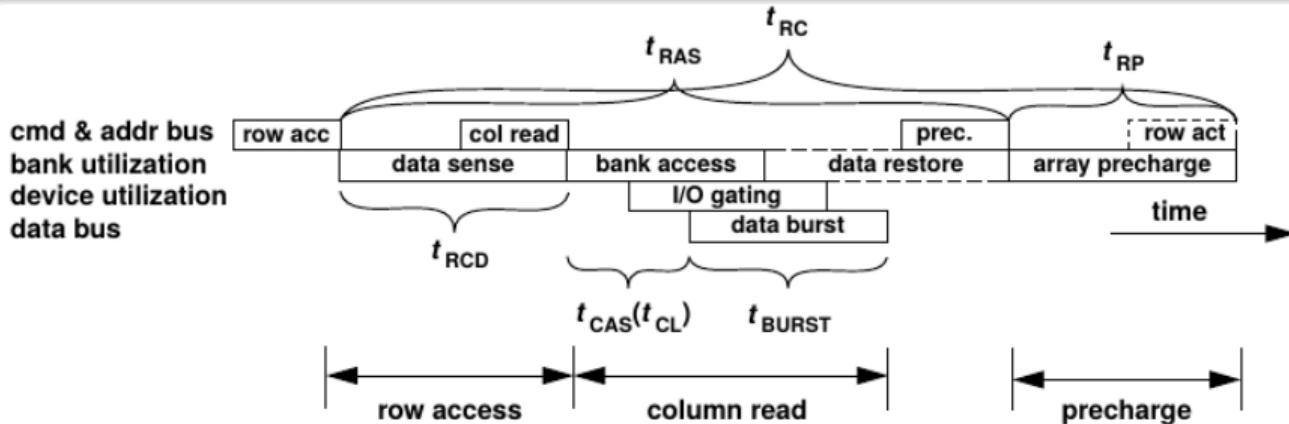
- Nuevamente el principio de vecindad en el acceso a memoria regula la eficiencia del sistema.
- Transcurrido  $t_{RCD}$  los datos de la fila seleccionada son precargados por los amplificadores de detección.
- El controlador de memoria envía el comando Column Read y los datos se transfieren al bus y se recargan en las celdas.

# Ciclo de Lectura



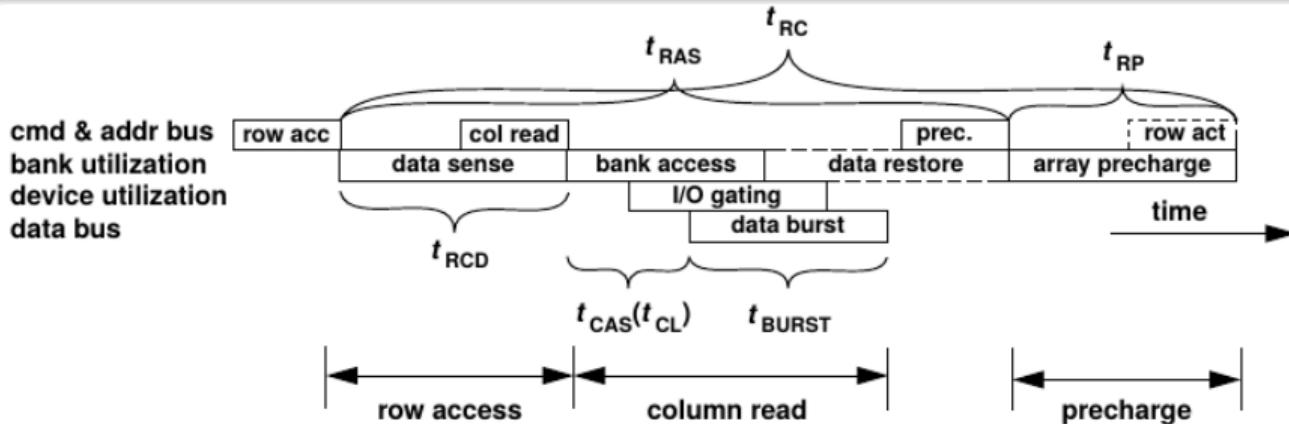
- Transcurrido  $t_{RAS}$  las celdas están listas para un comando Precharge que resetee las bitlines y los amplificadores de detección.

# Ciclo de Lectura



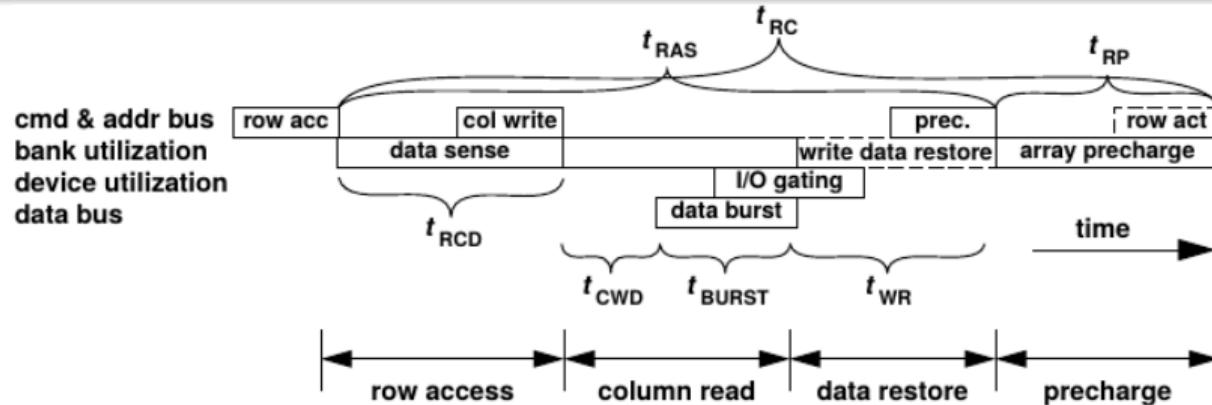
- Transcurrido  $t_{RAS}$  las celdas están listas para un comando Precharge que resetee las bitlines y los amplificadores de detección.
- Sistemas close-page: son los que precargan los bancos para acceder a una fila diferente luego del  $t_{RAS}$ .

# Ciclo de Lectura

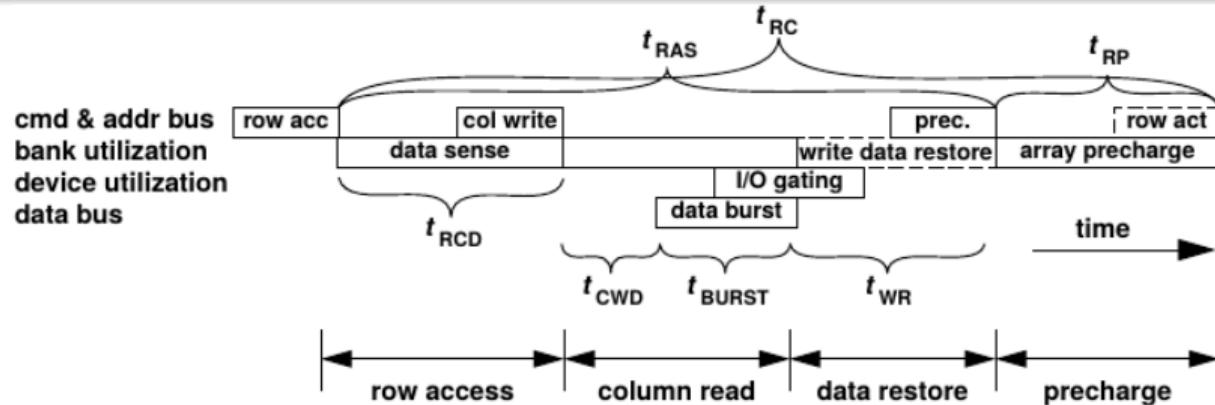


- Transcurrido  $t_{RAS}$  las celdas están listas para un comando Precharge que resetee las bitlines y los amplificadores de detección.
- Sistemas close-page: son los que precargan los bancos para acceder a una fila diferente luego del  $t_{RAS}$ .
- Sistemas open-page: son los que mantienen la fila activa y los amplificadores de detección luego del  $t_{RAS}$  especulando con accesos secuenciales desde el procesador (principio de vecindad).

# Ciclo de Escritura

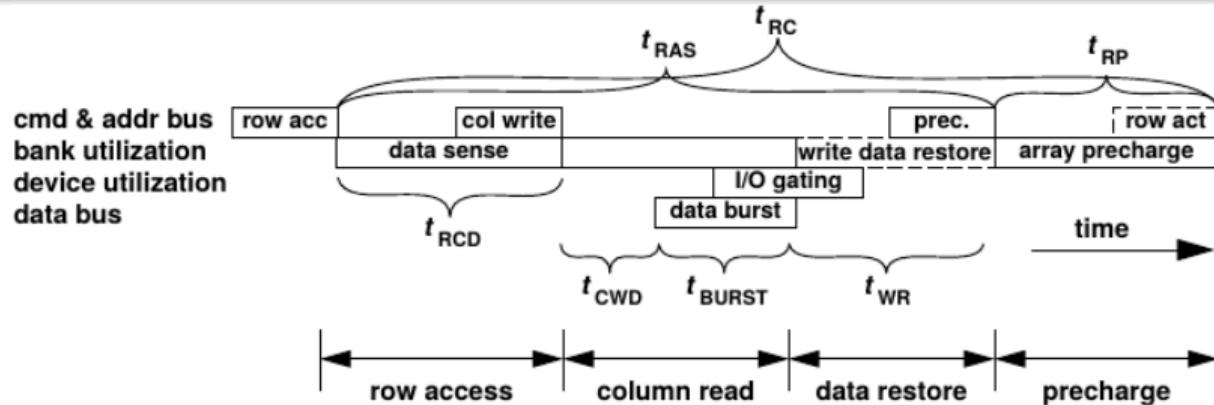


# Ciclo de Escritura



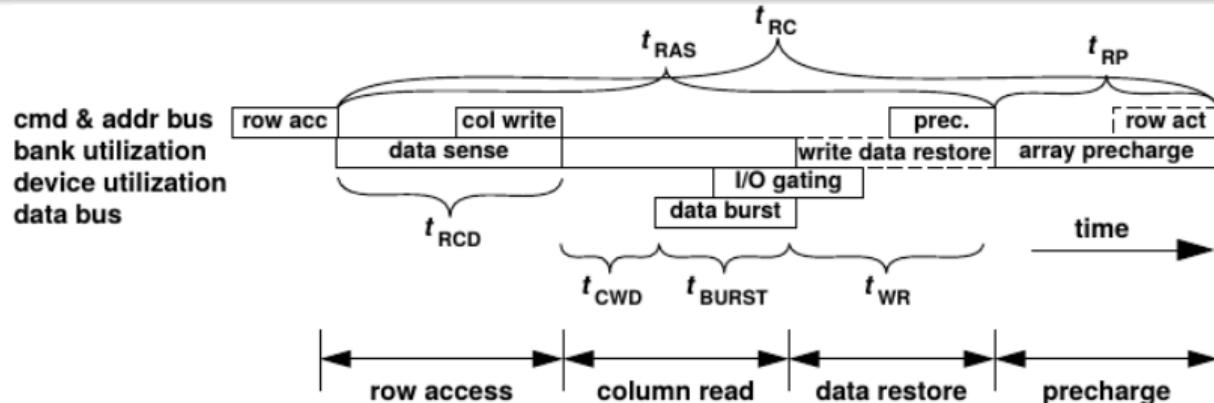
- En los dispositivos DRAM modernos el tiempo de ciclo de una fila está determinado por el tiempo de escritura.

# Ciclo de Escritura



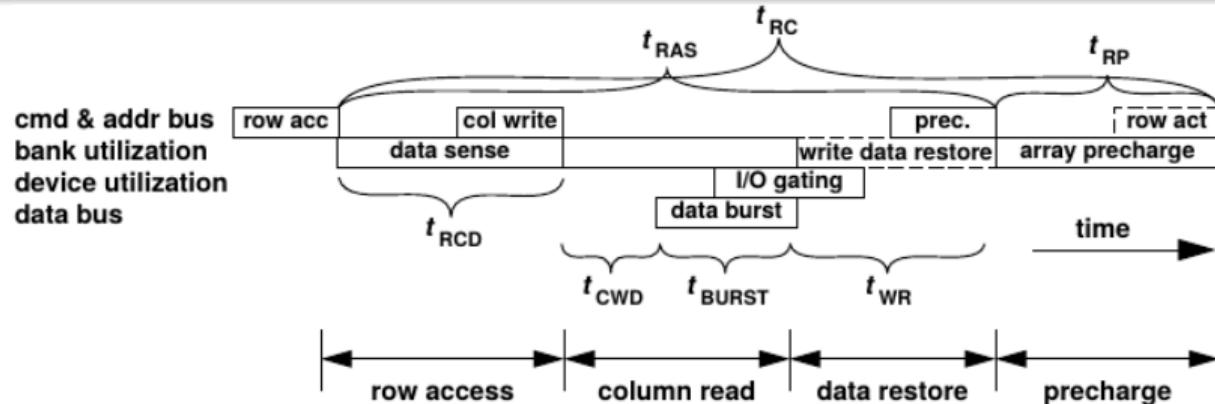
- En los dispositivos DRAM modernos el tiempo de ciclo de una fila está determinado por el tiempo de escritura.
- Es el tiempo que necesita un DRAM Device para proveer acceso a una fila en un determinado banco de celdas de DRAM.

# Ciclo de Escritura



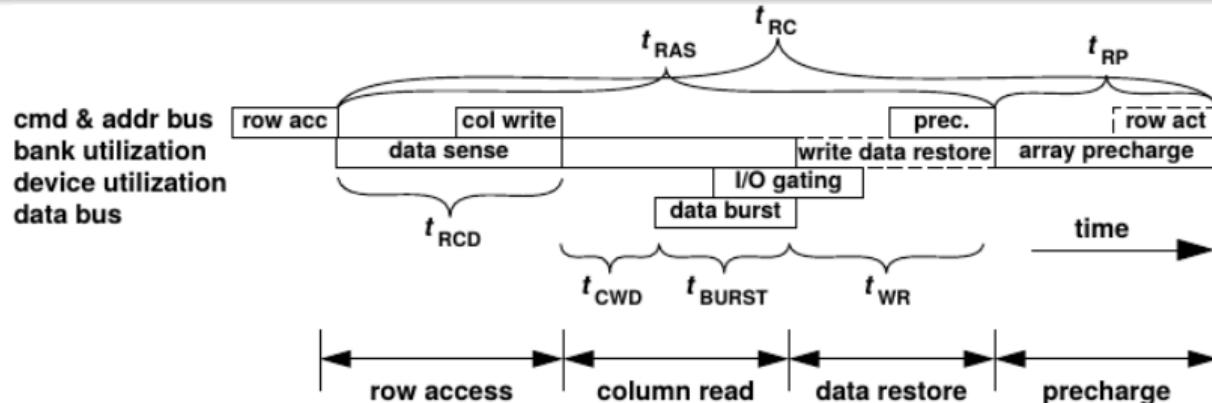
- En los dispositivos DRAM modernos el tiempo de ciclo de una fila está determinado por el tiempo de escritura.
- Es el tiempo que necesita un DRAM Device para proveer acceso a una fila en un determinado banco de celdas de DRAM.
- En el caso de un ciclo de escritura, los datos deben ser provistos por el controlador de memoria, viajan por el bus de datos, pasan por los multiplexores de disparo de E/S, saturan los amplificadores de detección, y finalmente llegan a la celda.

# Ciclo de Escritura



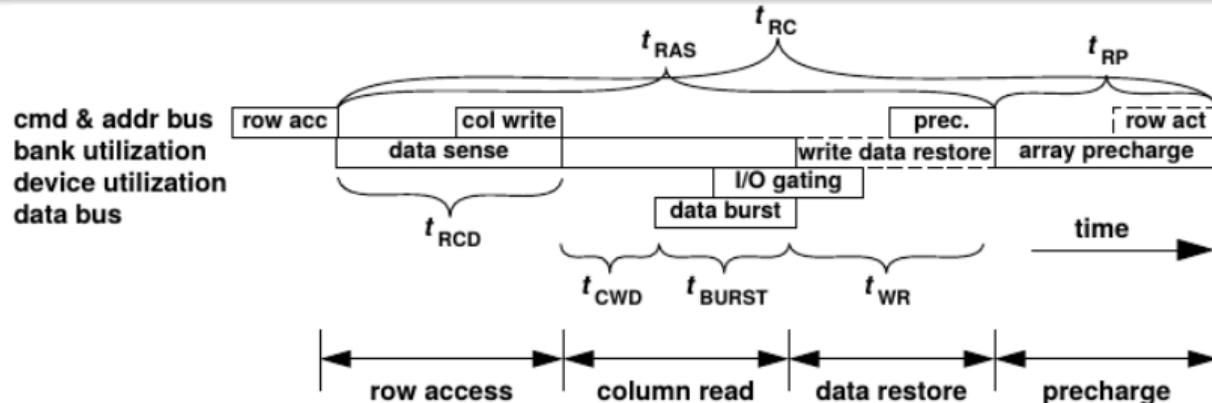
- Esta secuencia se debe completar antes de enviar el comando de precarga que completa el ciclo de Escritura.

# Ciclo de Escritura



- Esta secuencia se debe completar antes de enviar el comando de precarga que completa el ciclo de Escritura.
- De este modo el  $t_{RAS}$  debe ser suficientemente extenso como para abarcar  $t_{RCD}$ ,  $t_{CWD}$ ,  $t_{CCD}$  y  $t_{WR}$ .

# Ciclo de Escritura



- Esta secuencia se debe completar antes de enviar el comando de precarga que completa el ciclo de Escritura.
- De este modo el  $t_{RAS}$  debe ser suficientemente extenso como para abarcar  $t_{RCD}$ ,  $t_{CWD}$ ,  $t_{CCD}$  y  $t_{WR}$ .
- En la práctica debe setearse en un valor por lo menos igual a la suma de los otros cuatro.