

# Clase 21 - Coherencia Caché

---

Profesor: **IIC2343 - Arquitectura de Computadores**

- Felipe Valenzuela González

Correo:

frvalenzuela@alumni.uc.cl


# **Resumen de la clase pasada**

# Arquitectura de Computadores: Mejoras y extensiones

En lo que hemos visto hasta ahora

- Una máquina programable que ejecuta programas
- Interacción con dispositivos de entrada y salida

Lo que nos queda ver **mejoras**:

- Mejora los accesos a memoria 
- Mejora en lo que respecta a la acceso a memoria en más de un procesador
- Mejora en lo que respecta a la ejecución de instrucciones en la CPU.

## Tipos de arquitectura paralela - Memoria compartida

- Una política de escritura **write-through**, no habrá problemas ya que la memoria será consistente para todos los procesadores (más tiempo)
- Actualmente se opta por velocidad por tanto hay se sigue el protocolo de escritura **write-back**, pero de este nace un nuevo problema: la **coherencia de caché**

## Coherencia de caché - Protocolo MSI

- **Modified:** Indica que el contenido de la línea fue modificado y no es consistente con el bloque de la memoria principal. Este puede existir solo en una caché para un mismo bloque compartido.
- **Shared:** Indica que el contenido de la línea se encuentra en al menos una memoria caché, pero no se ha modificado.
- **Invalid:** Indica que el contenido de la línea es inválido, que puede ser por dato no existente o por solicitud desde el bus compartido.

# Coherencia de caché - Protocolo MSI

Para entender el protocolo, se definen los siguientes eventos:

- Eventos del procesador actual
  - **PrRd**: Solicitud de lectura (Read - Rd).
  - **PrWr**: Solicitud de escritura (Write - Wr).
- Eventos del bus compartido -
  - **BusRd**: Solicitud de un bloque de datos **sin** intención de modificarlo.
  - **BusRdX**: Solicitud de un bloque de datos **con** intención de modificarlo.
  - **Flush**: Transferencia de línea de caché a la memoria principal.

# Protocolo MSI - Diagrama General

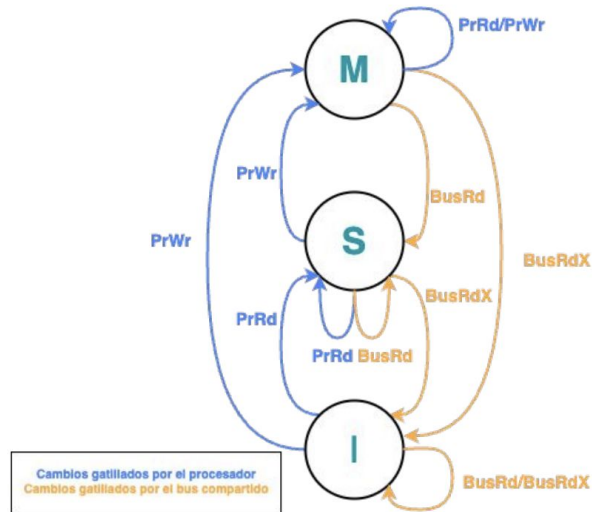


Diagrama de estado de una línea de caché para el protocolo MSI.

**¿Dudas?**



**A new foe has appeared!**

**CHALLENGER APPROACHING**



## Coherencia de caché - Protocolo MESI

- En este protocolo, el estado **Shared** cambia su uso y se agrega otro nuevo:
- **Shared**: Indica que el contenido de la línea se encuentra **en más de una caché** (al menos dos)
- **Exclusive**: Indica que el contenido de la línea se encuentra **exclusivamente en dicha caché**, sin modificaciones

## Coherencia de caché - Protocolo MESI

se añade la confirmación (o no) de que el contenido de una línea de caché exista o no en otra

- **BusRd(S)**: Indica que se realiza una lectura y se obtuvo una confirmación **afirmativa** de la existencia del contenido en otra caché
- **BusRd( $\neg$ S)**: Indica que se realiza una lectura y se obtuvo una confirmación **negativa** de la existencia del contenido en otra caché

## Protocolo MSI - Cambios por eventos del procesador

- Si se hace la lectura de un dato que no está en memoria y que se confirma que no está en ninguna otra caché, el estado de la línea pasa a estado **Exclusive**
- Si se realiza una lectura en este estado, se mantiene igual y **no se emiten señales**
- Si se realiza una escritura, pasa a estado **Modified** pero no envía ninguna señal al bus compartido

## Protocolo MSI - Cambios por eventos del BUS

- Si un controlador recibe la señal **BusRd** para una de sus líneas en estado **Exclusive**, entonces esta ya no es la única y por ende cambia a estado **Shared**
- Si en cambio recibe la señal **BusRdX**, pasa a estado **Invalid** ya que no solo **no será la única** caché que tiene el contenido, sino que no tendrá **la última versión de este**

**¿Dudas?**

# Protocolo MESI - Diagramas

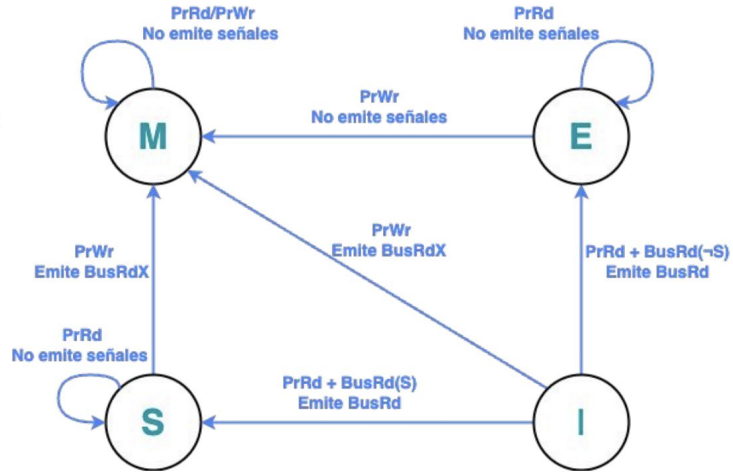


Diagrama de estado de una línea de caché para el protocolo MESI y eventos del procesador.

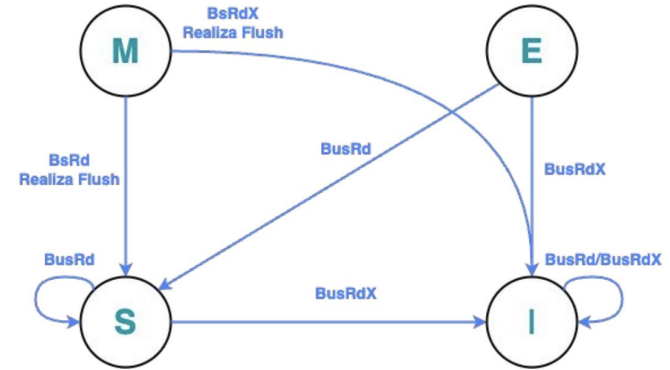


Diagrama de estado de una línea de caché para el protocolo MESI y eventos del bus compartido.

# Protocolo MESI - Diagrama General

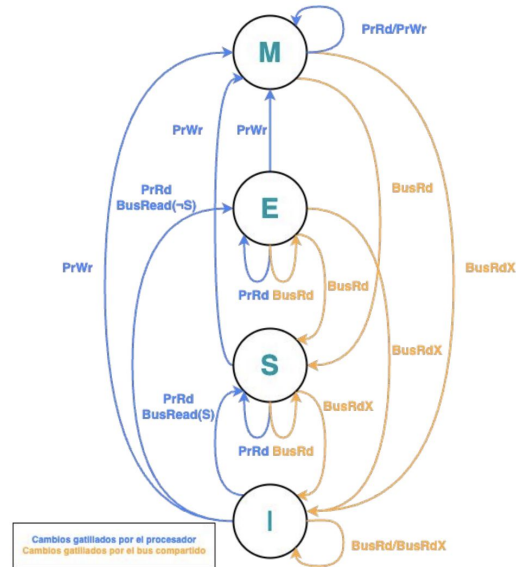


Diagrama de estado de una línea de caché para el protocolo MESI.



# Ejercicios

## Protocolo MESI - EJERCICIO

- Considere la siguiente memoria compartida con los datos señalados en la tabla
- Esta es compartida por tres CPUs que ejecutan simultáneamente los siguientes programas

Dirección	Label	Valor
0xCD	i	5
0xCE	temp	-45
0xCF	var1	2
0xD0	var2	9
0xD1	arr	20
0xD2		-10
0xD3		3
0xD4		9

CPU0

```
MOV A,(var1)
MOV B,(var2)
ADD A,B
MOV (arr),A
ADD A,(i)
MOV (temp),A
```

CPU1

```
MOV A,arr
ADD A,1
MOV (temp),A
MOV B,(temp)
MOV B,(B)
MOV (var1),B
```

CPU2

```
MOV B,(i)
MOV A,(arr)
ADD A,B
MOV (i),A
INC B
MOV (var2),B
```

# Protocolo MESI - EJERCICIO

- Indique los valores finales de las variables para cada caché privada y complete las siguientes tablas según cómo se actualiza cada celda a partir del protocolo MESI (indicando con una letra qué bit está activo). Puede asumir que cada variable se almacena en una línea de la caché.

	CPU0							
Instrucción	i	temp	var1	var2	arr[0]	arr[1]	arr[2]	arr[3]
1								
2								
3								
4								
5								
6								

	CPU1							
Instrucción	i	temp	var1	var2	arr[0]	arr[1]	arr[2]	arr[3]
1								
2								
3								
4								
5								
6								

	CPU2							
Instrucción	i	temp	var1	var2	arr[0]	arr[1]	arr[2]	arr[3]
1								
2								
3								
4								
5								
6								

# Protocolo MESI - EJERCICIO

- Se destaca que las variables **rojas** señalan que la línea quedó con el contenido de estas, pero fue invalidado por modificaciones de otras caché

CPU0	CPU1	CPU2
i = 25 temp = 36 var1 = 2 var2 = 9 arr[0] = 11 arr[1] = - A = 36 B = 9	i = - temp = 0xD2 var1 = -10 var2 = - arr[0] = - arr[1] = -10 A = 0xD2 B = -10	i = 25 temp = - var1 = - var2 = 6 arr[0] = 20 arr[1] = - A = 25 B = 6

CPU0							
i	temp	var1	var2	arr[0]	arr[1]	arr[2]	arr[3]
S	M	I	I	M	I	I	I
CPU1							
i	temp	var1	var2	arr[0]	arr[1]	arr[2]	arr[3]
I	I	M	I	I	E	I	I
CPU2							
i	temp	var1	var2	arr[0]	arr[1]	arr[2]	arr[3]
S	I	I	M	I	I	I	I

**¿Dudas?**

# Ejercicio prueba I3 - 2024-2

Suponga que posee una arquitectura MIMD de memoria compartida con 3 CPU que poseen su propia caché y que siguen el protocolo MESI para asegurar consistencia. La memoria contiene los siguientes datos de variables; mientras que CPU0, CPU1 y CPU2 ejecutan los siguientes programas:

Dirección	Label	Valor
0x00	var1	255
0x01	var2	253
...	...	...
0xFE		0
0xFF		0

// CPU0	// CPU1	// CPU2
MOV A, (var1)	MOV A, 0	MOV B, 0
MOV B, (var2)	NOP	PUSH B
AND B,A	PUSH A	INC B
INC B	POP B	PUSH B
MOV B, (B)		POP A
MOV (B), B	MOV B,(B)	

Asumiendo que cada dirección se almacena en una línea distinta, y que el contador SP posee un valor inicial igual a 255 en **todas** las CPU, indique el estado de cada línea de las caché (M/E/S/I) para cada ciclo completando la tabla adjunta, asumiendo que todas las líneas parten en estado I (ciclo 0):

# Ejercicio prueba I3 - 2024-2

[illegible]

# Ejercicio prueba I3 - 2024-2

**Solución:** A continuación se muestra la tabla final esperada por cada iteración:

	CPU0				CPU1				CPU2			
Ciclo	0x00	0x01	0xFE	0xFF	0x00	0x01	0xFE	0xFF	0x00	0x01	0xFE	0xFF
0	I	I	I	I	I	I	I	I	I	I	I	I
1	E	I	I	I	I	I	I	I	I	I	I	I
2	E	E	I	I	I	I	I	I	I	I	I	M
3	E	E	I	I	I	I	I	M	I	I	I	I
4	E	E	I	I	I	I	I	M	I	I	M	I
5	E	E	S	I	I	I	I	M	I	I	S	I
6	S	M	S	I	S	I	I	M	I	I	S	I



# Ejercicio prueba I3 - 2024-2

Para llegar al estado final correcto, es importante haber tomado en cuenta lo siguiente:

- Si se realiza la operación `AND B,A` con 8 bits para cada registro, se llega al resultado `B = 253`, por lo que la CPU0 luego accede a la dirección de memoria `254 = 0xFE` considerando la ejecución de la instrucción `INC B`.
- La ejecución de `POP A` y `POP B` toma dos ciclos. La lectura del valor de memoria se realiza en el **segundo ciclo**, dado que el primero lo único que hace es incrementar el valor de `SP`.
- Al ser `SP` parte de una CPU, su valor es **independiente** de lo que ejecuten otras. Es decir, que una CPU realice un `PUSH` o un `POP` no afecta el valor de `SP` de otra.

**¿Dudas?**

# Bibliografía

- Apuntes históricos. Hans Löbel, Alejandro Echeverría

12 - Paralelismo Avanzado

- D. Patterson, Computer Organization and Design RISC-V. Edition: The Hardware Software Interface. Morgan Kaufmann, 2020. Capítulo 5.12. Página 459, 586 en PDF

# Bibliografía

- <https://github.com/IIC2343/Syllabus-antiores/tree/main/Enunciados>

The screenshot shows the GitHub interface for the repository `IIC2343 / Syllabus-antiores`. The left sidebar displays the file tree with the `Enunciados` directory selected. The main content area shows the commit history for this directory, listing 15 commits with their names, messages, and dates.

Name	Last commit message	Last commit date
...		
Enunciados 2019-1	feat: add more solutions to previous exams; add previous auxiliary cl...	2 years ago
Enunciados 2019-2	feat: add more solutions to previous exams; add previous auxiliary cl...	2 years ago
Enunciados 2020-1	feat: add more solutions to previous exams; add previous auxiliary cl...	2 years ago
Enunciados 2021-1	add enunciados antiguos	2 years ago
Enunciados 2021-2	feat: add assessments for past semester	2 years ago
Enunciados 2022-1	feat: add assessments for past semester	2 years ago
Enunciados 2022-2	feat: add more solutions to previous exams; add previous auxiliary cl...	2 years ago
Enunciados 2023-1	feat: add assessments for past semester	2 years ago
Enunciados 2023-2	add missing test solutions for last semester	last year
Enunciados 2024-1	remove unused readme	10 months ago
Enunciados 2024-2	Delete Enunciados/Enunciados 2024-2/a	4 months ago
Compilado_IIC2343.pdf	Agregando compilado de Felipe	5 years ago

# Clase 21 - Coherencia Caché

---

Profesor: **IIC2343 - Arquitectura de Computadores**

- Felipe Valenzuela González

Correo:

frvalenzuela@alumni.uc.cl