DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (II/2026)

Ayudantía 13 - Coherencia de Caché

Ayudantes: Ignacio Gajardo (gajardo.ignacio@uc.cl), Mario Rojas (mario.denzel@estudiante.uc.cl), Nicolás Romo Aubele (nroma@uc.cl), Fernanda Escobar (fer_jez@uc.cl)

Pregunta 1: Preguntas conceptuales

(a) Te han encomendado liderar el diseño de hardware para dos tipos de servidores con propósitos muy distintos:

- **Servidor A**: Un servidor de bases de datos transaccionales en tiempo real que requiere un acceso altamente simétrico y predecible a un gran bloque de memoria central por parte de decenas de núcleos, donde evitar cuellos de botella en la latencia uniforme es crítico.
- **Servidor B**: Un supercomputador de alta densidad para computación científica masiva, estructurado en múltiples nodos físicos interconectados, donde cada nodo agrupa su propia cantidad masiva de procesadores y bloques de memoria dedicados.

¿Qué esquema de memoria compartida (UMA o NUMA) asignarías a cada servidor y por qué?

Solución:

- **Servidor A**: Se le asigna UMA, pues al requerir tiempos de acceso uniformes y simétricos para todos los procesadores a través de un bus único o un crossbar switch, se evitan variaciones imprevistas de latencia.
- **Servidor B**: Se le asigna NUMA, ya que permite escalar la cantidad de procesadores organizándolos en nodos con memoria local propia, optimizando el rendimiento al acceder primero al segmento de memoria local asignado.

(b) Explique la principal diferencia entre los mecanismos de *Bus Snooping* y *Bus Snarfing* en la gestión de la coherencia de caché.

Solución: En *Bus Snooping* los controladores de caché simplemente "escuchan" o revisan las solicitudes en el bus compartido (como lecturas o invalidaciones) para determinar si deben invalidar o actualizar el estado de sus propias líneas. En cambio, en *Bus Snarfing*

además de escuchar, cuando el controlador detecta una escritura dirigida a una dirección que posee, actualiza inmediatamente el contenido de su propia caché con los datos de dicha solicitud.

(c) Imagina que un bloque de memoria se encuentra cargado simultáneamente en las cachés de la CPU_0 y la CPU_1 bajo el protocolo MESI. Ambas líneas se encuentran en estado Shared (S). Si la CPU_0 decide realizar una operación de escritura (PrWr) sobre dicho bloque:

- ¿Qué señal debe emitir el controlador de caché de la CPU_0 hacia el bus compartido?
- ¿Cómo afecta esto al estado de la línea en la CPU_1?
- ¿Qué estado adopta finalmente la línea en la CPU_0?

Solución: La CPU_0 debe emitir la señal BusRdX a través del bus compartido. Al recibir esta señal, el controlador de la CPU_1 detecta que su copia dejará de ser válida, por lo que actualiza inmediatamente su línea al estado Invalid (I). Finalmente, la línea en la CPU0 pasa al estado Modified (M).

Pregunta 2: Coherencia caché (P2, I3-2026-1)

Suponga que posee una arquitectura multiprocesador 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`	127
0x01	`var2`	126
...	...	...
0xFE		0
0xFF		0

```
// CPU0              // CPU1              // CPU2
POP A                MOV A, (var1)     MOV A, 1
                     ADD A, 1          MOV B, 2
POP B                MOV B, (var2)     PUSH A
                     ADD B, A          PUSH B
ADD A, B             MOV (B), A        POP B
MOV (var1), A        MOV (var2), 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):

	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												
2												
3												
4												
5												
6												

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

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

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

- 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.
- 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 hacer `POP` sin haber hecho un `PUSH` antes, la operación generará un *overflow* sobre el contador `SP`, causando que su valor cambie de 255 a 0. Esto implica que, en la CPU0, al ejecutar `POP` realizará una lectura de memoria sobre la dirección 0 en el segundo ciclo y luego sobre la dirección 1 en el cuarto ciclo, *i.e.* lee las variables `var1` y `var2`.
- Si una CPU hace una lectura sobre una línea ya modificada por ella misma, esta sigue estando en dicho estado y solo cambia en dos escenarios: (1) si otra CPU realiza una lectura sobre ella, cambiando a estado *Shared*; o (2) si otra CPU realiza una escritura sobre ella, cambiando a estado *Invalid*.

Pregunta 3: Coherencia caché (P4, I3-2024-1)

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	`arr`	2
0x01		0
0x02		1
...	...	...
0xFE		0
0xFF		0

```
// CPU0          // CPU1          // CPU2
MOV B,arr        MOV A,2          MOV A,5
INC B            PUSH A           MOV B,1
MOV B,(B)        POP A            PUSH A
MOV (B),B                         PUSH 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):

	CPU0					CPU1					CPU2				
Ciclo	**0x00**	**0x01**	**0x02**	**0xFE**	**0xFF**	**0x00**	**0x01**	**0x02**	**0xFE**	**0xFF**	**0x00**	**0x01**	**0x02**	**0xFE**	**0xFF**
0	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
1															
2															
3															
4															

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

	CPU0					CPU1					CPU2				
Ciclo	**0x00**	**0x01**	**0x02**	**0xFE**	**0xFF**	**0x00**	**0x01**	**0x02**	**0xFE**	**0xFF**	**0x00**	**0x01**	**0x02**	**0xFE**	**0xFF**
0	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
1	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
2	I	I	I	I	I	I	I	I	I	M	I	I	I	I	I
3	I	E	I	I	I	I	I	I	I	I	I	I	I	I	M
4	M	E	I	I	I	I	I	I	I	S	I	I	I	M	S

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

- `MOV B,arr` **no es una instrucción de lectura de memoria**. Solo guarda la dirección de `arr` en `B`.
- La ejecución de `POP A` 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.

Feedback ayudantía

Escanee el QR para entregar feedback sobre la ayudantía.

