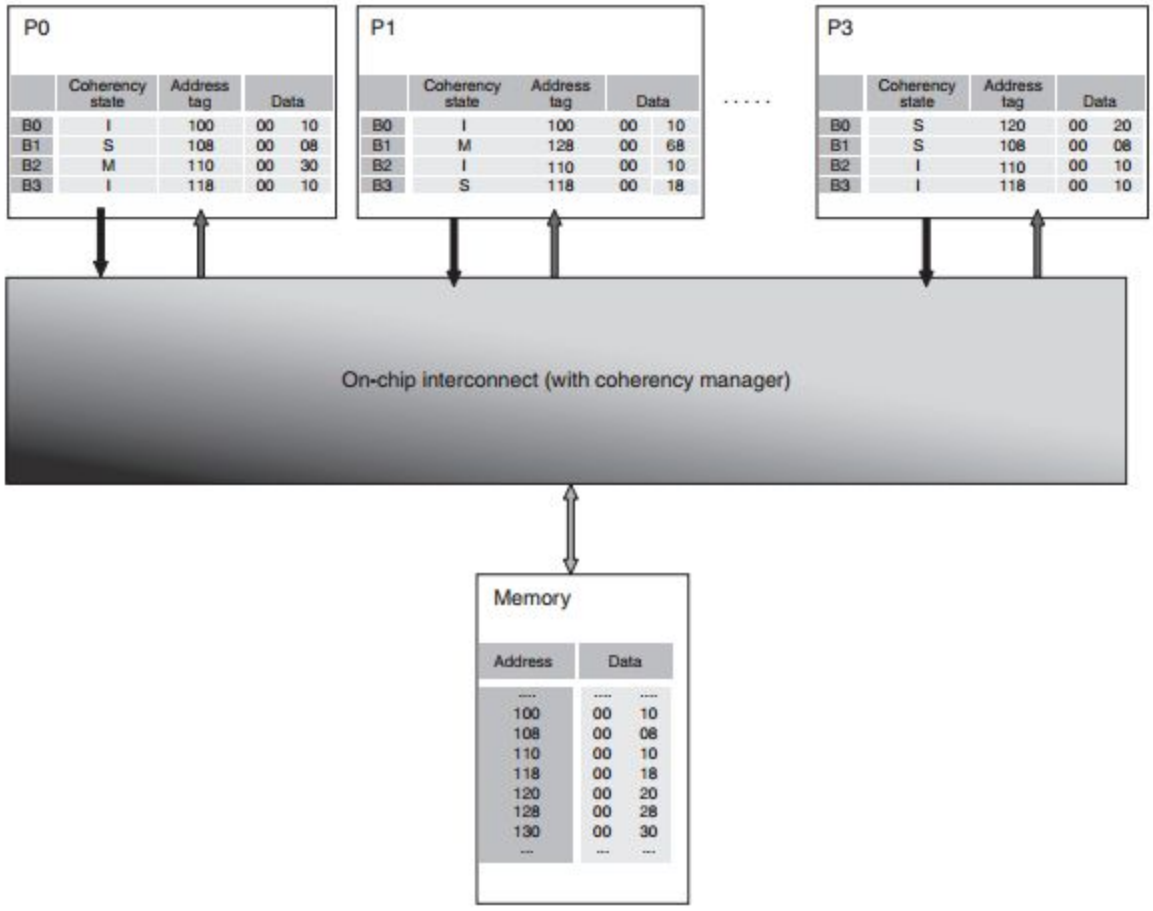
Práctico - Multiprocesadores

Ejercicio 1

Considere un multiprocesador con cuatro CPUs (P0, P1, P2, P3) como en la figura. Suponga que se utiliza el protocolo de coherencia de caché MSI.



Dado el estado de la memoria, indicar, para cada uno de los siguientes accesos, cuál es el valor retornado por el caché, cuáles son las acciones realizadas por el sistema de memoria y cuál es el estado del sistema luego del acceso (indicar solo aquellos bloques que cambien)

P0: read 120
P0: write 120 ← 80
P3: write 120 ← 80
P1: read 110
P1: read 100
P0: write 108 ← 48
P0: write 130 ← 78
P3: write 130 ← 78
P1: write 100 ← 90

Ejercicio 2

Repita el ejercicio 1 si se utiliza el protocolo MESI en lugar de MSI. Indique para cada caso qué mensajes se envían por el bus de interconexión y compárelos con los enviados en el ejercicio 1.

Ejercicio 3

Suponga el estado inicial de la memoria del ejercicio 1, pero esta vez operando con un protocolo basado en directorios.

a. Indique el estado inicial del directorio, a fin de que sea coherente con el resto del sistema de memoria.
b. Indique, para cada uno de los accesos, cuál es el estado del directorio y cachés luego de realizado el mismo, y qué mensajes son intercambiados entre los cachés y el directorio.

Ejercicio 4

Suponga un multiprocesador de memoria distribuida de 16 gb, con 8 procesadores y bloques de memoria de 256 bytes. Suponga que se utiliza un protocolo de coherencia de caché basado en directorios. Calcule cuántos bits son necesarios para mantener el directorio. Suponga que se utilizan dos bits para mantener el estado de cada bloque.

Ejercicio 5

Una solución propuesta al problema de *false sharing* es agregar un bit de 'válido' por palabra. Esto permitiría invalidar una palabra sin remover el bloque entero, permitiendo a un procesador acceder a la porción del bloque que no ha sido invalidada por otro procesador. Explique qué modificaciones deberían realizarse en el protocolo MSI para implementar este agregado.

Ejercicio 6

Explique qué acciones podría tomar el compilador de una aplicación *multihilo* que use memoria compartida para evitar el *false sharing*. **Sugerencia:** Piense en cómo se posicionan tradicionalmente las variables en memoria.

Ejercicio 7

Varios protocolos de coherencia por *husmeo* tienen estados adicionales, transiciones adicionales o mensajes diferentes para reducir el *overhead* de mantener la coherencia de caché. Una optimización usual es la introducción de un estado *Owned* (denotado con la letra O). El estado Owned se comporta como el Compartido (*Shared*) si el caché realiza lecturas sobre el bloque, pero se comporta como el estado Modificado (*Modified*) en el sentido de que el bloque debe proveer el contenido del bloque si otro caché lo precisa (si realiza un read/write miss a dicho bloque). Un *read miss* a un bloque en estado Owned o Modificado es provisto por el caché con dicho bloque y el caché que provee el bloque transiciona al estado Owned (si estaba en estado Owned, permanece en este estado) Este protocolo es útil para evitar que se realicen muchos pedidos a memoria, y tiene sentido para aquellos procesadores donde brindar un dato desde un caché es más rápido que realizar la consulta a memoria principal. Dibuje un diagrama de estados para este nuevo protocolo.