## ARQUITECTURA DE COMPUTADORES II

AUTORES:
David Expósito Singh
Florin Isaila
Daniel Higuero Alonso-Mardones
Javier García Blas
Borja Bergua Guerra

*Área de Arquitectura y Tecnología de Computadores*
*Departamento de Informática*
*Universidad Carlos III de Madrid*

Julio de 2012

# TEMA 3:
# *MP DE MEMORIA COMPARTIDA (II)*

# Índice (II)

4. Modelo de coherencia de memoria
   1. Introducción
   2. Consistencia secuencial
   3. Consistencia relajada
      1. Processor consistency
      2. Weak consistency
      3. Release consistency

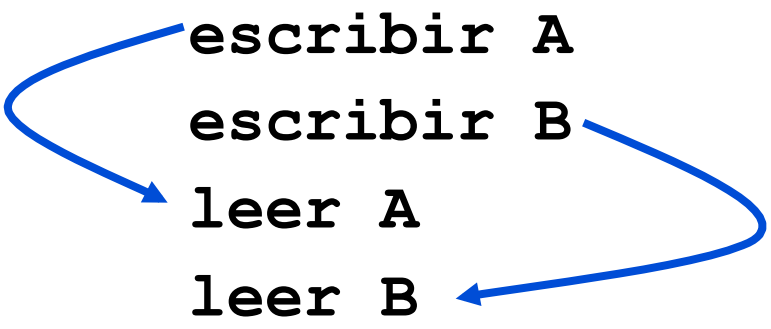
# MCM: Introducción

- Idea:
  - Nuestra "mentalidad": modelo 'uniprocesador': accesos atómicos a M y en el orden del programa.

    Sin embargo, el compilador o la UC (p.ej., **superescalares**) no necesariamente fuerzan el orden secuencial: **buffer de escritura**, **terminación fuera de orden**, etc.
  - ¿Cuál es la "semántica" de las operaciones de acceso a memoria (compartida)?
  - ¿Cómo se ordenan las **escrituras** y **lecturas** entre los diferentes procesadores?

# MCM: Introducción (II)

- Nuestra visión:

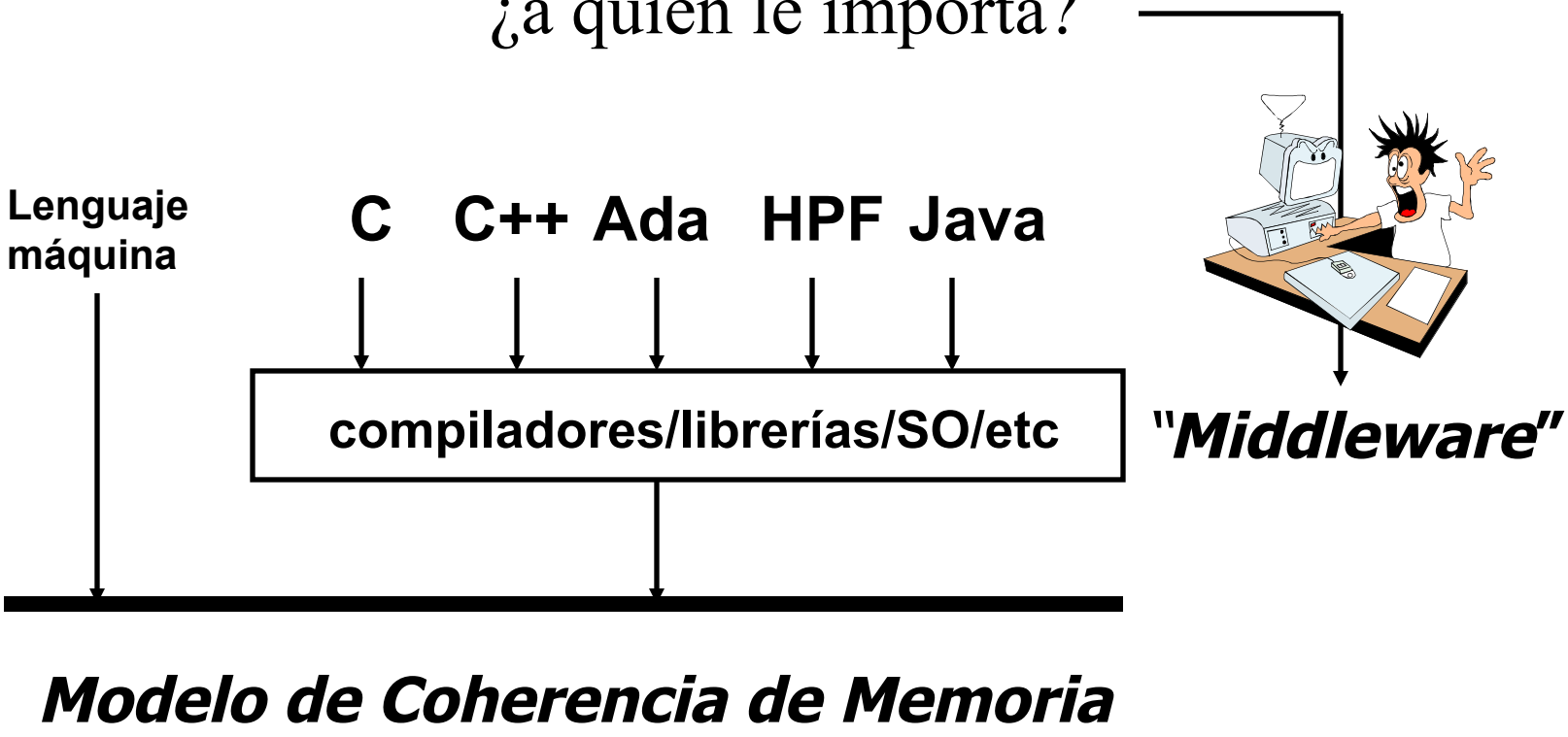



- Ejecución en MP donde los procesadores comparten información: el orden relativo en que se realizan los accesos es importante.

# Modelo de consistencia de memoria (MCM)

- *"Memory Consistency Models for Shared Memory MP"*

¿a quién le importa?



# MCM: Introducción (III)

- Ejemplo:

**P1**

```
A:= 66;
flag:= 1;
```

**P2**

```
while (flag==0)
do
        nothing
print A;
```

- ¿Qué resultado esperamos? ¿Por qué pudiera ser que no lo obtuviésemos?:
  - caches privadas: ¿?
  - buffer de escritura: ¿?
  - red de interconexión/buses: ¿?

# MCM: Consistencia Secuencial

- Consiscencia Secuencial (*Sequential Consistency*, **SC**) de Lamport (1979):

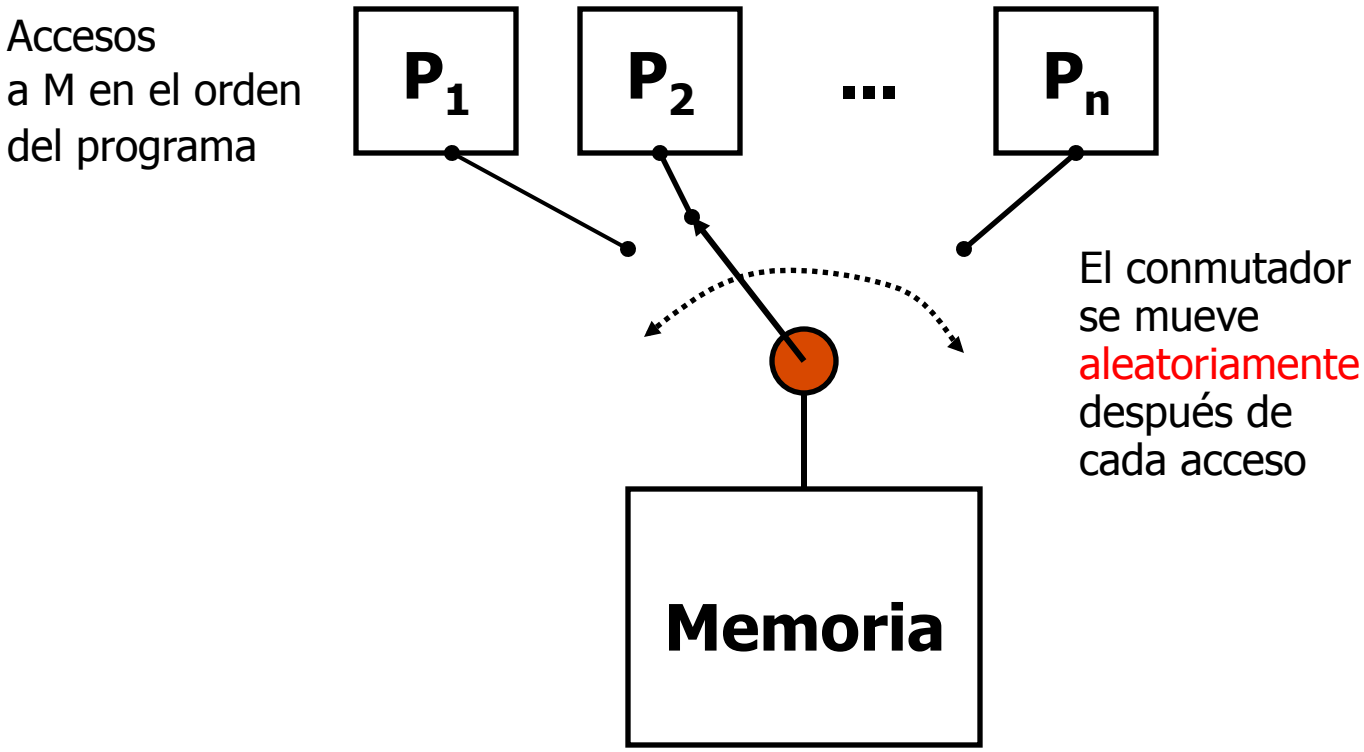
> *"A system is sequentially consistent if the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program."*

# MCM: Consistencia Secuencial

- Es decir, dos requisitos:
  1. Se mantiene el orden del programa para las operaciones de cada procesador.
  2. Se mantiene un orden secuencial único entre *todas* las operaciones.
- ¿"Semántica" cercana a la usada en programación secuencial en un monoprocesador?:
  - Se mantiene cualquier orden *implícitamente* asumido por el programador:
    - ¿asumimos de manera natural el orden *como lo ve* el programador?:

# MCM: Consistencia Secuencial

- Visualización: *serialización* de los accesos:



# MCM: Consistencia Secuencial

- Condiciones suficientes para consistencia secuencial (Dubois et al., 1987):

1. Supongamos que existe un mecanismo de **coherencia de caches**:
   - Las **escrituras** en una misma dirección se observan en el mismo orden desde cualquier procesador.

2. En cada procesador espera a "arrancar" (*issues*) cualquier acceso hasta que se completa el anterior:
   - Una **lectura** se completa cuando se obtiene su valor.
   - Una **escritura** se completa cuando el nuevo valor está visible para todos los procesadores.
   - Por simplicidad, supondremos que las escrituras son atómicas.

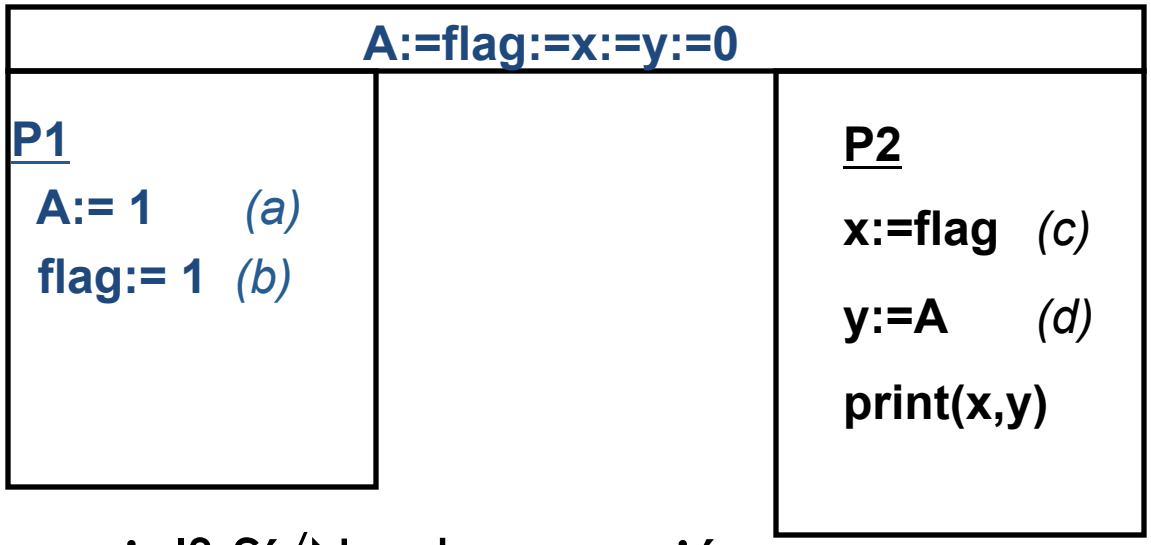
# MCM: Consistencia Secuencial

- Interpretación de Culler et al.: *condiciones suficientes* para consistencia secuencial:

1. Cada proceso "lanza" (issues) las operaciones de memoria en el orden del programa.
2. Después de lanzar una operación de escritura, el proceso espera que se complete la escritura antes de lanzar la siguiente operación.
3. Después de lanzar una operación de lectura, el proceso espera a que se complete la lectura y a que se realice la escritura cuyo valor va a ser devuelto por la lectura, antes de arrancar la siguiente operación: escritura atómica.

# MCM: Consistencia Secuencial

- Ejemplo 1:




¿Coherencia Secuencial? Sí/No: demostración

**0 0**

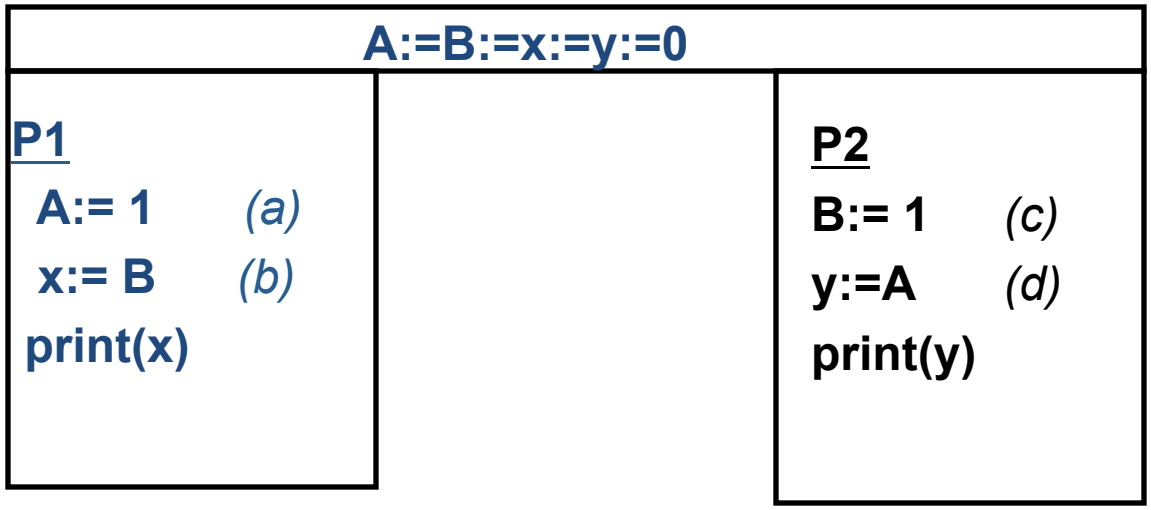
**1 1**

**0 1**

**1 0**

# MCM: Consistencia Secuencial

- Ejemplo 2:




¿Consistencia Secuencial? Sí/No: demostración: x-y

**1 1**

**0 1**

**1 0**

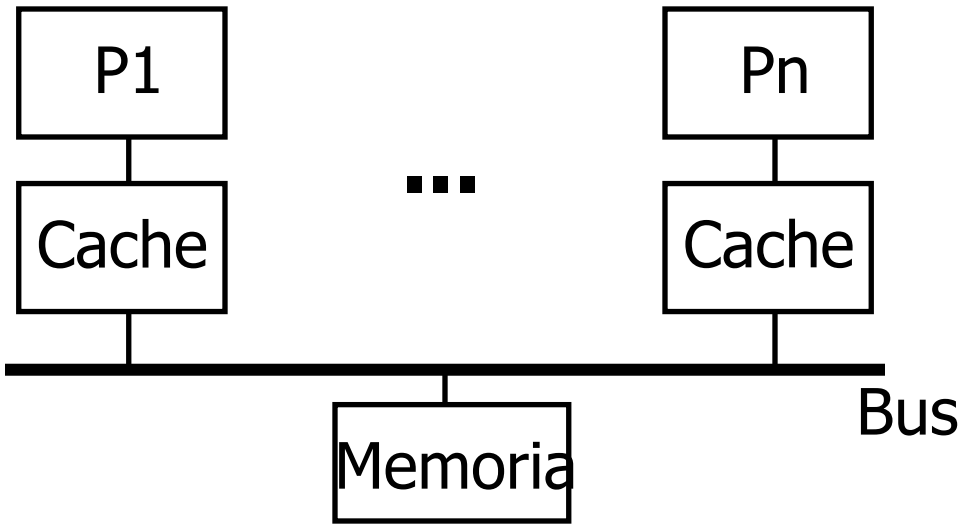
**0 0**

# MCM: Consistencia Secuencial

- Influencia sobre el modelo de consistencia secuencial en el empleo de memoria cache.
  - Memoria cache con escritura directa a memoria (*write-through*) y actualización de caches.
    - Garantiza la consistencia secuencial.
  - Memoria cache con post-escritura a memoria (*write-back*).
    - En función de la implementación, puede garantizar o no la consistencia secuencial.

# MCM: Consistencia Secuencial

- Escenario 1:

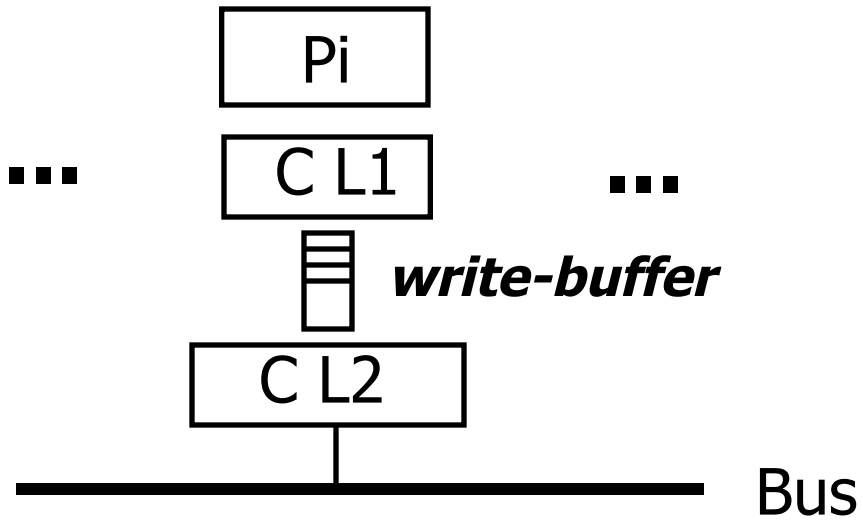



- Supongamos "write-back" con **política de actualización**.
- Hay un mecanismo de coherencia de caches: 'serialización' por el bus: escrituras 'serializadas'.
- Una escritura observada al mismo tiempo por todos los procesadores.
- Los accesos desde cada procesador se completan en el orden de programa.

*Se garantiza la SC sin ningún mecanismo adicional*

# MCM: Consistencia Secuencial

- Escenario 2:




  - L1: *write-through*; L2: *write-back*.
  - Las lecturas en L2 se retrasan hasta que se vacía el WB.

    Si siempre se accede al L2 con actualización: garantiza SC (caso anterior).

    Si permitimos acceder al L1: SC: necesario retrasar los accesos a este nivel hasta que no haya escrituras pendientes en el WB.

    *EL MP puede no mantener SC en pro de rendimiento*

# MCM: Consistencia Secuencial

- Escenario 3: NUMA
  - Ahora no existe bus que serialice los accesos: el único orden que garantiza la red es *punto-a-punto*.
  - Coherencia de caches:
    - » serializa una dirección de memoria.
  - Los accesos iniciados en orden no se completan necesariamente en orden: distribución de la memoria + caminos de diferente longitud a través de la red.
  - Las **escrituras** son inherentemente **no atómicas.**
  - Necesidad de conocer cuándo se completa una escritura:
    - » Para conseguir atomicidad.
    - » Para retrasar un acceso hasta que uno previo finalice.
  - Necesita mensajes de confirmación: reconocimiento de todas las invalidaciones en una escritura: contador.
  - Manera de asegurar atomicidad en escrituras: retrasar el acceso a un valor hasta que se obtengan todos las confirmaciones.
  - Como consecuencia, latencias muy largas: decenas a centenas de ciclos

# MCM: Consistencia Relajada

- Modelos de **Consistencia Relajada** (*Relaxed Consistency, RC*):
  - Menos exigente que la SC, pero más eficiente: permite optimizaciones jugando con el orden de los accesos a memoria.
  - Todos los modelos de RC suministran algún tipo de mecanismo que permite forzar *explícitamente* la coherencia secuencial.
  - Modelos:
    - *Processor Consistency* (PC): Goodman, 1989
    - *Weak Consistency* (WC): Dubois *et al.*, 1986
    - *Release Consistency* (RC): Garachorloo *et al.*, 1990

# MCM: Consistencia Relajada (II)

- *Processor Consistency (PC):*
  - El procesador que realiza el acceso a memoria ve secuencias de *loads* y *stores*, o *loads* seguidos por un *store*, en el orden del programa.
  - Sin embargo, varios stores no se realizan necesariamente en el orden del programa.
  - Este funcionamiento implica que el procesador utiliza un *buffer* de escritura.

# MCM: Consistencia Relajada (III)

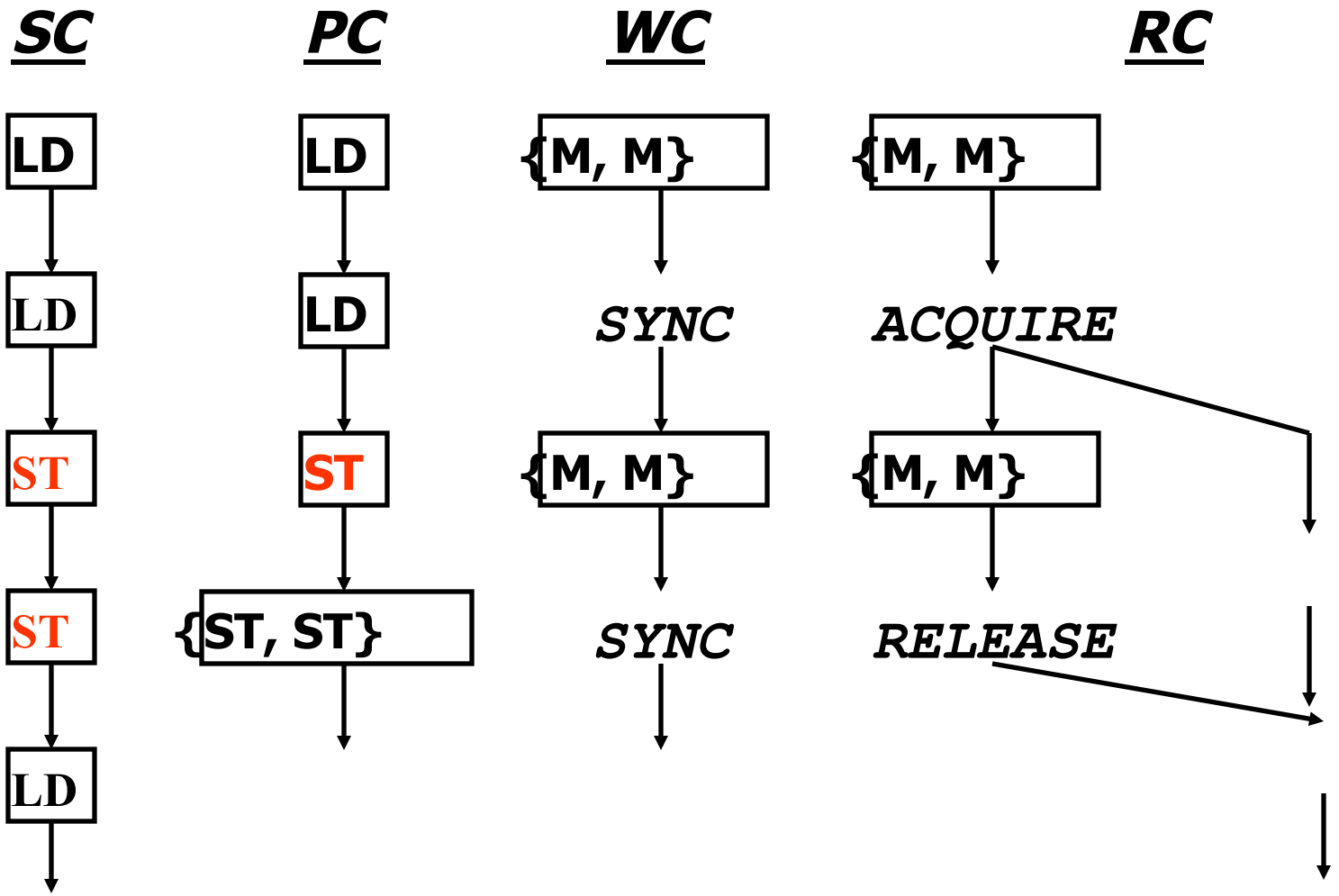
- ***Weak Consistency (WC):***
  - Sólo se garantiza la coherencia en los puntos de sincronización: por ejemplo mediante la primitiva SYNC: todos los restantes accesos a memoria pueden ocurrir en cualquier orden.

- ***Release Consistency (RC):***
  - Es un modelo aún menos exigente: primitivas ACQUIRE y RELEASE.

    ACQUIRE es una operación de lectura y permite "ganar" el acceso a una variable compartida (p.ej., un cerrojo o un flag).

    RELEASE es operación de escritura para "ceder" el acceso a una variable compartida.

# MCM: Esquema comparativo



# MCM: Referencias

- Libros: el Culler y, especialmente, el Sima *et al*.
- Específicas:
  - S.V. Adve and K. Gharachorloo: *Shared Memory Consistency Models: A Tutorial,* IEEE Computer, Dec. 1996: 66-76
  - M.D. Hill: *Multiprocessors Should Support Simple Memory-Consistency Models,* IEEE Computer, Aug, 1998: 28-34 **[Disponible en Aula Global]**
  - Lugar donde se pueden encontrar una serie de pruebas que permiten determinar el tipo de modelo que sigue cada arquitectura:

    http://www.mpdiag.com/archtest.html