Tenemos la siguiente arquitectura de memoria el system multiprocesador donde se va a ejecutar tiene 32 GB ($32x2^{30}$ bytes) de memoria principal, distribuidos en los 4 nodos NUMA, y cada procesador tiene una memoria cache de 4MB ($4x2^{20}$ bytes). Dentro de cada nodo NUMA se utiliza el protocolo snoopy MESI, y entre nodos NUMA el protocolo basado en directorio; además el sistema operativo utiliza la política "first touch" para colocar en memoria los datos. El tamaño de un entero son 4 bytes y el tamaño de una línea de cache es de 32 bytes (3p).

a) Calcula el número de bits necesarios en cada directorio para mantener la coherencia entre nodos NUMA. (0,25p)

32GB * 1linea/32bytes= $32*2^{30}/2^5=32*2^{25}$ líneas en cada NUMA node

$2^{25}*2+4=32*2^{25}*6$ bits por NUMA node.

b) Calcula el número de bits necesarios para mantener la coherencia mediante MESI en cada cache. (0,25p)

4Mbytes($4*2^{20}$)/$2^5=4*2^{15}$ líneas por cache

$4*2^{15}*2$(bits de estado)=$4*2^{16}$bits por cada cache

c) Tenemos el siguiente código paralelizado donde N=16384:

```
void vectoradd(int *A, int *B, int *C, int N) {
#pragma omp parallel num_threads(8)
{
int id=omp_get_thread_num();
int howmany=omp_get_num_threads();
  for (int i=id; i <N; i+=howmany) {
     C[i]=A[i]+B[i];
                                    }
}
}
```

¿Cómo se reparten las iteraciones los threads? (0,25p)

En este código, los threads se reparten las iteraciones de la siguiente manera:

Cada thread se asigna un id, el cual es único entre todos los threads.

Se obtiene el número total de threads, el cual se almacena en la variable howmany.

El loop for se itera desde el id del thread hasta N, con un incremento de howmany.

Esto significa que cada thread se encarga de calcular las sumas de los elementos del vector C que se encuentran en posiciones que son múltiplos de su id de thread y múltiplos de howmany.

Por ejemplo, si N=16384, y hay 8 threads, entonces cada thread se encargará de calcular las sumas de los siguientes elementos del vector C:

Thread 0: elementos 0, 8, 16, 24, ..., 16384

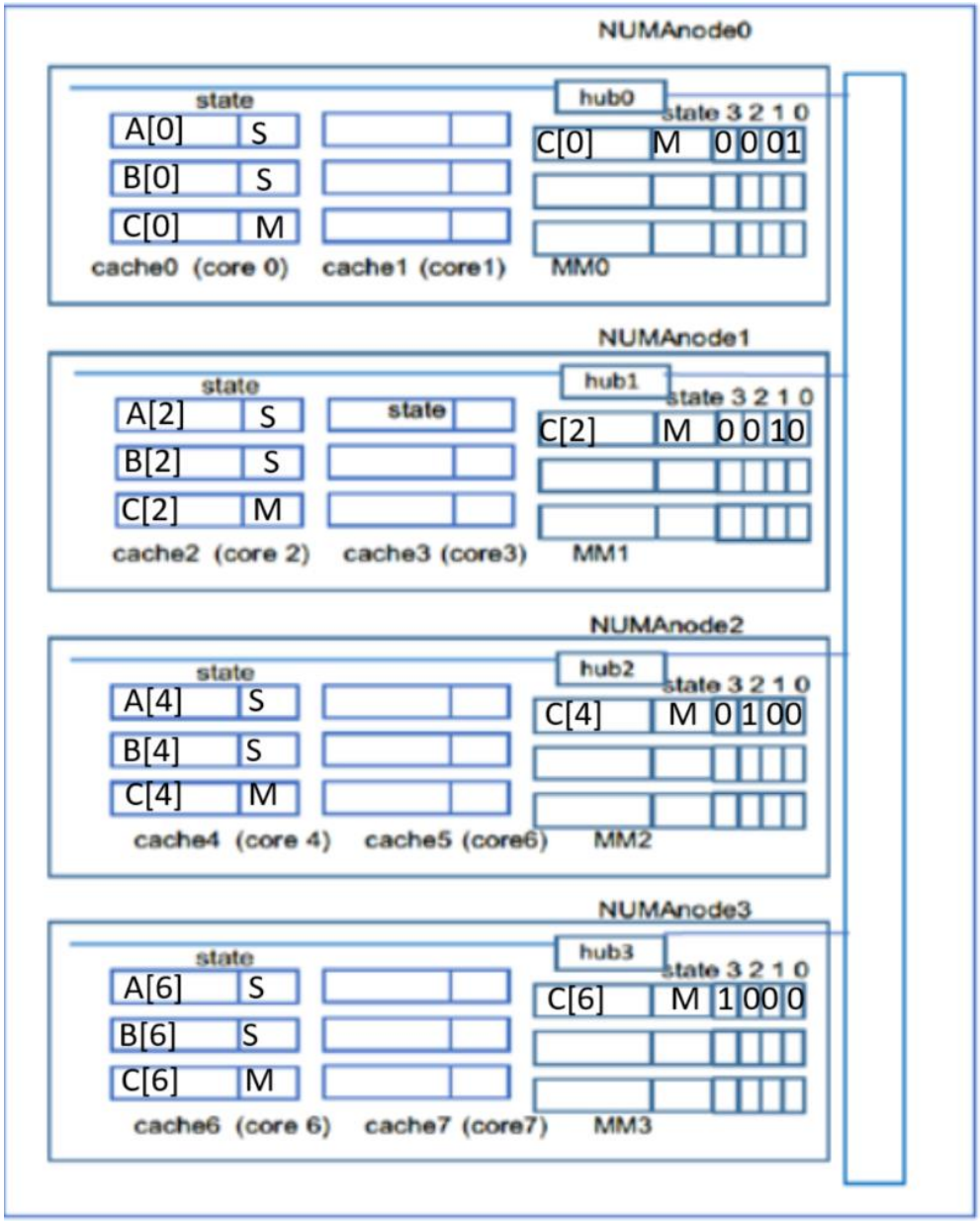
Thread 1: elementos 1, 9, 17, 25, ..., 16385

...

Thread 7: elementos 7, 15, 23, 31, ..., 16383

De esta manera, el código se ejecuta de manera eficiente, utilizando todos los threads disponibles para realizar las sumas.

d) Si todos los threads 0,1,2,3,4,5 6 y 7, hacen la primera iteración que les corresponde en orden (es decir el primero ejecuta el th0, después el th1, etc) como quedarían las caches y los NUMAnodes. (1p) --> Llena el dibujo anterior, o haz uno similar con los datos llenos.



e) ¿Y después de todas las iteraciones cómo quedarían las caches y los NUMA nodes? (0,5p, si no te cabe obviamente en el dibujo explícalo con tus palabras. Asume que los threads siguen ejecutando en el mismo orden del del apartado d).

NUMAnode0:

- **cache0 (core 0)**: Contiene los últimos elementos procesados por el thread 0, es decir, A[N-8], B[N-8] y C[N-8]. El estado es Modified (M) ya que el thread 0 ha escrito en C[N-8].
- **cache1 (core 1)**: Contiene los últimos elementos procesados por el thread 1, es decir, A[N-7], B[N-7] y C[N-7]. El estado es Modified (M) ya que el thread 1 ha escrito en C[N-7].

NUMAnode1:

- **cache2 (core 2)**: Contiene los últimos elementos procesados por el thread 2, es decir, A[N-6], B[N-6] y C[N-6]. El estado es Modified (M) ya que el thread 2 ha escrito en C[N-6].
- **cache3 (core 3)**: Contiene los últimos elementos procesados por el thread 3, es decir, A[N-5], B[N-5] y C[N-5]. El estado es Modified (M) ya que el thread 3 ha escrito en C[N-5].

NUMAnode2:

- **cache4 (core 4)**: Contiene los últimos elementos procesados por el thread 4, es decir, A[N-4], B[N-4] y C[N-4]. El estado es Modified (M) ya que el thread 4 ha escrito en C[N-4].
- **cache5 (core 5)**: Contiene los últimos elementos procesados por el thread 5, es decir, A[N-3], B[N-3] y C[N-3]. El estado es Modified (M) ya que el thread 5 ha escrito en C[N-3].

NUMAnode3:

- **cache6 (core 6)**: Contiene los últimos elementos procesados por el thread 6, es decir, A[N-2], B[N-2] y C[N-2]. El estado es Modified (M) ya que el thread 6 ha escrito en C[N-2].
- **cache7 (core 7)**: Contiene los últimos elementos procesados por el thread 7, es decir, A[N-1], B[N-1] y C[N-1]. El estado es Modified (M) ya que el thread 7 ha escrito en C[N-1].

Los directorios (hubs) en cada NUMAnode tendrán bits de presencia establecidos para las cachés que contienen los últimos elementos procesados por cada thread. Por ejemplo, el directorio en NUMAnode0 tendrá bits de presencia establecidos para las cachés 0 y 1, indicando que estas cachés contienen los elementos A[N-8], B[N-8], C[N-8], A[N-7], B[N-7] y C[N-7].

f) Si después de ejecutarse todo el código y quedar las caches y NUMA nodes como han quedado después del apartado e, el core 0 quiere escribir en C[9], cuáles serían los paso (paso a paso) y como quedarían las caches y los NUMA nodes.(0,75p)

Si el core 0 quiere escribir en C[9] después de que todo el código se haya ejecutado, los pasos serían los siguientes:

1. **Lectura de C[9]**: El core 0 necesita leer el valor actual de C[9]. Como C[9] no está en la caché 0, se produce un fallo de caché. El sistema entonces busca C[9] en las otras cachés y en la memoria principal.

2. **Invalidación de C[9] en otras cachés**: Si C[9] se encuentra en alguna otra caché (por ejemplo, la caché 1 si el thread 1 ha procesado la iteración 9), entonces esa caché debe invalidar su copia de C[9] porque el core 0 está a punto de modificarlo. Esto se hace para mantener la coherencia de la caché.
3. **Escritura de C[9]**: Una vez que C[9] ha sido leído en la caché 0 y se ha invalidado en todas las demás cachés, el core 0 puede proceder a escribir el nuevo valor en C[9].

Después de estos pasos, las cachés y los NUMAnodes quedarían de la siguiente manera:

- **NUMAnode0**:
  - **cache0 (core 0)**: Ahora contiene C[9] con el nuevo valor escrito por el core 0. El estado es Modified (M).
  - **cache1 (core 1)**: Si antes contenía C[9], ahora ese valor ha sido invalidado.
- **NUMAnode1**:
  - **cache2 (core 2)** y **cache3 (core 3)**: Si antes contenían C[9], ahora ese valor ha sido invalidado.
- **NUMAnode2**:
  - **cache4 (core 4)** y **cache5 (core 5)**: Si antes contenían C[9], ahora ese valor ha sido invalidado.
- **NUMAnode3**:
  - **cache6 (core 6)** y **cache7 (core 7)**: Si antes contenían C[9], ahora ese valor ha sido invalidado.