Informe:

Desarrollamos un modelo para el sistema de un procesador con dos núcleos, utilizando un monitor para garantizar la exclusión mutua y así resolver los problemas de concurrencia. El monitor hace uso de una Red de Petri para administrar así la generación y posterior procesamiento de las actividades. Se cuenta además con un objeto Política que resuelve el conflicto entre las transiciones sensibilizadas que alimentan los buffers de los núcleos, manteniendo la carga equitativa.

Entre los *criterios adoptados*, podemos mencionar que:

* El objeto Política asigna las actividades generadas a cada buffer según una asignación de probabilidades según el caso:

1. Ambos núcleos con el mismo tiempo de “service\_rate”: 50% para cada núcleo.
2. Un núcleo con el doble de tiempo de “service\_rate” que el otro: 66% para el núcleo con menor “service\_rate”.
3. Un núcleo con el triple de tiempo de “service\_rate” que el otro: 75% para el núcleo con menor “service\_rate”.

* El objeto Política posee un vector en el cual se ordenan las transiciones de la red según el orden de prioridad elegido, permitiendo así decidir en caso de conflicto, que transición es preferible disparar primero. El orden de prioridad elegido es:

1. T5
2. T8
3. T4
4. T10
5. Arrival\_rate
6. Service\_rate\_1
7. Service\_rate\_2
8. T0
9. T12
10. Power\_Up\_Delay\_1
11. Power\_Up\_Delay\_2
12. T2
13. T11
14. Power\_Down\_Threshold\_1
15. Power\_Down\_Threshold\_2

* Para asegurar exclusión mutua en el monitor, se implementaron semáforos en el acceso al monitor.
* Si un hilo intenta disparar una transición temporizada, primero se corrobora que este sensibilizada por tiempo, en caso de no estarlo, el hilo se duerme el tiempo que le haga falta. Antes de dormirse, libera el mutex. Transcurrido el tiempo, el hilo retorna del monitor para luego volver a intentar disparar la transición.
* Al disparar una transición en el monitor antes de permitirle el acceso a otro hilo, primero nos fijamos si hay algún otro hilo bloqueado dentro de la cola del monitor que pueda disparar una transición. En ese caso se elige qué hilo despertar en base a las prioridades que se encuentran en Política.
* Se considera terminada la ejecución del main al finalizar el procesamiento de las 1000 actividades. Los hilos de los núcleos que sigan en ejecución, luego de procesar dichas actividades, quedarán en las colas del monitor tras no poder salir del modo de “Stand\_by” al no generarse nuevas actividades.

*Resultados Obtenidos*:

* Debido a que los Núcleos son independientes entre sí, es posible trabajen en simultáneo si ambos tienen actividades disponibles en sus respectivos buffers.
* Al trabajar en simultáneo, siendo más los consumidores que los productores y, sabiendo que demoran la misma cantidad de tiempo para realizar sus actividades los buffers nunca llegan a su capacidad máxima, tendiendo a estar en cero la mayor parte del tiempo.
* Dado el paralelismo del desarrollo de las actividades desarrolladas por los P y C, el criterio de producción y consumo en cada buffer adoptado, y lo explicado en el punto anterior, podemos considerar que el tiempo de ejecución del programa depende solo del tiempo que demora un productor en cumplir su ciclo y su cantidad de iteraciones. En este caso en particular, el tiempo mínimo de ejecución que calculamos es de 8 minutos y 20 segundos (1000 iteraciones multiplicado por 50ms). Por consiguiente, de querer reducir este tiempo a 3 minutos, o menos, sería necesario reducir la cantidad de iteraciones y/o el tiempo de demora de actividad requerido por cada P y C.
* Tiempo de ejecución obtenidos:
  + Prueba 1:

alfa=5; beta=10; gamma=10; corridas=10

tiempo promedio de ejecución=5,494 seg

* + Prueba 2:

alfa=10; beta=10; gamma=10; corridas=10

tiempo promedio de ejecución=10,463 seg

* + Prueba 3:

alfa=5; beta=10; gamma=20; corridas=10

tiempo promedio de ejecución=6,520 seg

* + Prueba 4:

alfa=10; beta=10; gamma=20; corridas=10

tiempo promedio de ejecución=10,579 seg

* + Prueba 5:

alfa=5; beta=10; gamma=30; corridas=10

tiempo promedio de ejecución=7,794 seg

* + Prueba 6:

alfa=10; beta=10; gamma=30; corridas=10

tiempo promedio de ejecución=10,653 seg