Programación Concurrente 2019
Trabajo Práctico Nº 3

Condiciones

- El trabajo es grupal, de 3 a 4 alumnos (5 es la excepción).
- La defensa del trabajo es presencial
- La evaluación es individual (hay una calificación particular para cada integrante)
- Solo se corrigen los trabajos que hayan sido subidos al aula virtual (LEV)
- Los problemas de concurrencia deben estar correctamente resueltos y explicados.
- El trabajo debe implementarse en lenguaje Java.
- Se evaluará la utilización de objetos y colecciones, como así también la explicación de los conceptos relacionados a la programación concurrente.

Enunciado

Se debe implementar un simulador de un procesador con dos núcleos. A partir de la red de Petri de la **figura 1**, la cual representa a un procesador mono núcleo, se deberá extender la misma a una red que modele un procesador con dos núcleos. Además, se debe implementar una Política que resuelva los conflictos que se generan con las transiciones que alimentan los buffers de los núcleos (CPU_Buffer).

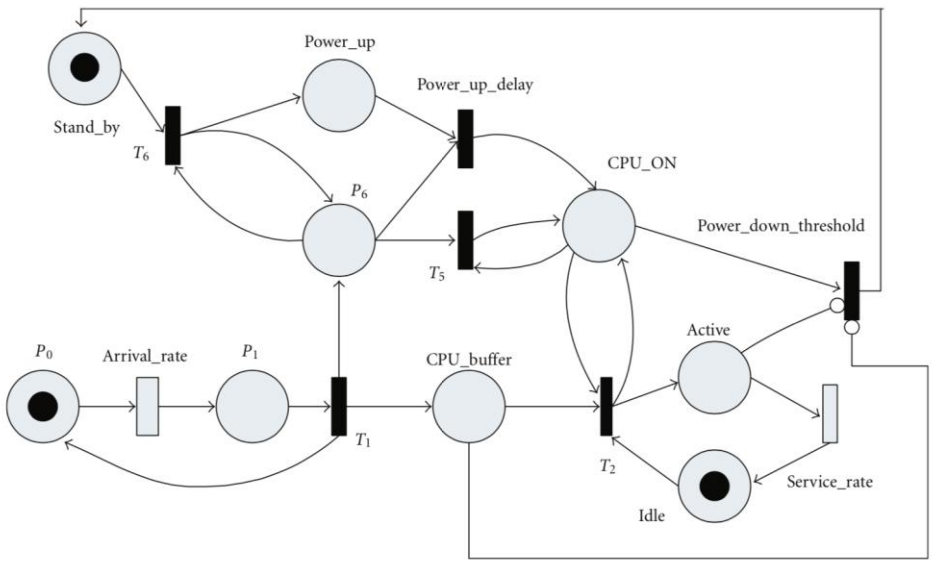

Figura 1

Nota: Las transiciones “Arrival_rate” y “Service_rate” son temporizadas y representan el tiempo de arribo de una tarea y el tiempo de servicio para concluir una tarea.

Ejercicios

1) Modelar el sistema de un procesador con dos núcleos, extendiendo la red de Petri de la ***figura 1***. Verificar todas sus propiedades haciendo uso de la herramienta **PIPE** o **Petrinator**.
2) Hacer el diagrama de clases que modele el sistema.
3) Implementar un objeto Política que resuelva el conflicto entre las transiciones sensibilizadas que alimentan los buffers de los núcleos, manteniendo la carga de la CPU equitativa.
4) Hacer el diagrama de secuencia que muestre el disparo exitoso de una de las transiciones que alimenta a uno de los buffers de la CPU, mostrando el uso de la política.
5) Indicar la cantidad de hilos necesarios para la ejecución y justificar.
6) Modelar el sistema con objetos en Java.
7) Realizar las siguientes ejecuciones con 1000 tareas completadas (para cada caso):
    a) Ambos núcleos con el mismo tiempo de *"service_rate".*
    b) Un núcleo con el doble de tiempo de *"service_rate"* que el otro.
    c) Un núcleo con el triple de tiempo de *"service_rate"* que el otro.
8) Verificar los resultados haciendo uso de un archivo de log, verificando el cumplimiento de los invariantes.
9) Debe hacer una clase Main que al correrla, inicie el programa.

Entregar:

a) Un archivo de imagen con el diagrama de clases, en buena calidad.
b) Un archivo de imagen con el diagrama de secuencias, en buena calidad.
c) Un archivo de imagen con la red de Petri, en buena calidad.
d) El código fuente Java (proyecto) de la resolución del ejercicio.
e) Un informe (obligatorio) que documente lo realizado, los criterios adoptados y que explique los resultados obtenidos.

Subir al LEV el trabajo **TODOS** los participantes del grupo.

Fecha de entrega

24 de Junio de 2019

Fecha de defensa

25 de Junio de 2019