Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3086 Programación de microprocesadores

Ciclo 2 de 2,024
Temario A

Laboratorio 05

Competencias para desarrollar
Distribuir la carga de trabajo entre hilos utilizando programación en C y OpenMP.

Instrucciones
Esta actividad se realizará individualmente. Al finalizar los períodos de laboratorio o clase, deberá entregar este archivo en formato PDF y los archivos .c en la actividad correspondiente en Canvas.

1. **(18 pts.)** Explica con tus propias palabras los siguientes términos:
   a) Private: cada hilo obtiene su propia copia de esa variable. Esto significa que cada hilo puede modificar su copia sin afectar a las copias de otros hilos. Al final de la región paralela, el valor de las variables privadas no se conserva, y el hilo maestro no ve los cambios hechos por los hilos.

   b) Shared: Es accesible y modificable por todos los hilos dentro de la región paralela. Todos los hilos comparten la misma instancia de la variable, lo que significa que los cambios realizados por un hilo pueden ser vistos por los demás hilos. Es importante gestionar adecuadamente el acceso concurrente para evitar condiciones de carrera.

   c) Firstprivate: Con una diferencia importante: la variable privada se inicializa con el valor de la variable correspondiente en el hilo maestro (el hilo que ejecuta el código antes de entrar en la región paralela). Es como tener una copia privada para cada hilo, pero todas las copias se inicializan con el mismo valor antes de que empiece la región paralela.

   d) Barrier: Es un punto de sincronización en un programa OpenMP donde todos los hilos deben detenerse y esperar hasta que todos los hilos alcancen esa barrera. Solo cuando todos los hilos han llegado a la barrera, pueden continuar ejecutando el código que sigue. Es útil para asegurar que ciertas secciones del código no se ejecuten hasta que todos los hilos hayan completado la ejecución de una parte previa del programa.

   e) Critical: Es una región de código que debe ser ejecutada por un solo hilo a la vez. Es una forma de asegurar que una porción crítica del código no sea ejecutada simultáneamente por múltiples hilos, lo cual podría causar problemas como condiciones de carrera. Si un hilo está ejecutando una región crítica, los demás hilos deben esperar hasta que la región quede libre.

   f) Atomic: Es una forma más ligera que critical para garantizar que una operación específica se realice de manera atómica, es decir, sin interrupciones por otros hilos. Es útil cuando se necesita realizar una simple operación (como incrementar un contador) de forma segura y eficiente sin incurrir en el mayor costo de rendimiento que podría tener una región critical. El atomic asegura que la operación sobre una variable sea ejecutada de manera indivisible, evitando condiciones de carrera en operaciones simples.

2. **(12 pts.)** Escribe un programa en C que calcule la suma de los primeros N números naturales utilizando un ciclo ***for*** **paralelo**. Utiliza la cláusula **reduction con +** para acumular la suma en una variable compartida.
   a) Define N como una constante grande, por ejemplo, N = 1000000.

b) Usa omp_get_wtime() para medir los tiempos de ejecución.

3. **(15 pts.)** Escribe un programa en C que ejecute tres funciones diferentes en paralelo usando la **directiva #pragma omp sections**. Cada sección debe ejecutar una función distinta, por ejemplo, una que calcule el factorial de un número, otra que genere la serie de Fibonacci, y otra que encuentre el máximo en un arreglo, operaciones matemáticas no simples. Asegúrate de que cada función sea independiente y no tenga dependencias con las otras.

4. **(15 pts.)** Escribe un programa en C que tenga un ciclo for donde se modifiquen dos variables de manera paralela usando #pragma omp parallel for.
   a. Usa la cláusula shared para gestionar el acceso a la variable1 dentro del ciclo.
   b. Usa la cláusula private para gestionar el acceso a la variable2 dentro del ciclo.
   c. Prueba con ambas cláusulas y explica las diferencias observadas en los resultados.

5. **(30 pts.)** Analiza el código en el programa Ejercicio_5A.c, que contiene un programa secuencial. Indica cuántas veces aparece un valor key en el vector a. Escribe una versión paralela en OpenMP utilizando una descomposición de tareas **recursiva**, en la cual se generen tantas tareas como hilos.

6. **REFLEXIÓN DE LABORATORIO: se habilitará en una actividad independiente.**