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: significa que cada hilo (thread) que ejecuta esa sección tiene su propia copia de esa variable. Al inicio de la sección paralela, las variables privadas no están inicializadas y al final de la sección, las copias privadas desaparecen. Por lo tanto, cualquier cambio hecho a estas variables dentro de los hilos no afecta a la variable original ni a las copias de otros hilos.
   b) Shared: todas las copias de la variable son compartidas entre los hilos, es decir, todos los hilos ven y modifican la misma copia de la variable. Esto puede llevar a problemas de sincronización si varios hilos intentan modificar la variable al mismo tiempo.
   c) Firstprivate: similar a private, pero con una diferencia importante: las variables que se declaran como firstprivate se inicializan con el valor de la variable original cuando se entra en la sección paralela. Esto significa que cada hilo tiene su propia copia privada, pero estas copias empiezan con el mismo valor que la variable origina.
   d) Barrier: es un punto de sincronización en el código donde todos los hilos deben detenerse hasta que todos hayan llegado a ese punto. Ningún hilo puede pasar de la barrera hasta que todos los demás hilos la hayan alcanzado.
   e) Critical: es una región del código que debe ser ejecutada por un solo hilo a la vez. Cuando un hilo está ejecutando una sección critical, los demás hilos deben esperar hasta que esa sección esté disponible nuevamente.
   f) Atomic: asegura que una operación específica en una variable compartida se realice de manera atómica, es decir, sin ser interrumpida por otros hilos. A diferencia de las secciones critical, las operaciones atómicas están optimizadas para ser más rápidas 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.

Las diferencias consisten en lo siguiente. Durante la ejecución del programa, la salida que se imprime desde cada hilo en cada iteración del bucle podría parecer desordenada debido a que los hilos ejecutan en paralelo y no hay un control sobre el orden en que se imprimen las líneas. Además, los valores de variable1 serán incrementados por diferentes hilos en paralelo, lo que puede resultar en un valor final inesperado debido a las condiciones de carrera que se generan a causa de ello.

La salida de variable2 mostrará el valor de i correspondiente a cada iteración para ese hilo en particular, ya que cada hilo maneja su propia copia de variable. En este caso no se generan condiciones de carrera.

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.

   El valor key aparece un total de 3 veces en el vector a.

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