UNIVERSIDAD DE COSTA RICA
ESCUELA DE INGENIERÍA ELÉCTRICA

TAREA 2

IE-0521 Estructuras de Computadoras Digitales II

I Ciclo de 2016

Fecha de entrega:

Lunes 20 de junio de 2016 antes de las 7:00 am en la página del curso. Después de esta hora se recibirá por correo electrónico aplicando las normas que la carta al estudiante del curso menciona.

Equipos:

La tarea se podrá desarrollar en equipos de máximo 2 personas.

Valor:

10% del total de la nota del curso.

Objetivo:

Desarrollar conceptos relacionados a procesadores de arquitecturas de memoria compartida: paralelismo y coherencia.

Descripción:

1. Usando un lenguaje de programación de alto nivel (C++, Python, Go recomendados) desarrolle un programa que simule el comportamiento de un par de CPUs usando el protocolo MESI (https://en.wikipedia.org/wiki/MESI_protocol) para mantener la coherencia de los datos en el cache.
   a) Se debe simular un sistema con dos CPUs y dos niveles de cache:
      - Cache L1 local, 8 KB.
      - Cache L2 compartido, 64 KB.
      - Bloques de 16 B.
      - Asociatividad: Mapeo directo.
      - Los dos CPUs no necesariamente deben correr en paralelo para la simulación: la simulación puede alternar la ejecución de ambos CPUs.
   b) El programa deberá poder procesar un archivo que contiene direcciones de memoria y datos:
      - El archivo a utilizar está disponible en: http://ie0521.jsdanielh.org/memory-trace-gcc.trace.gz
      - El archivo contiene varias líneas donde cada una contiene el siguiente formato:
        - `<dirección en hexadecimal> {L,S}`

- Cada línea representa la dirección de memoria que se desea acceder y si se está haciendo una lectura (L) o una escritura (S).
- Se debe asumir que las líneas impares son instrucciones del CPU0 y las pares son instrucciones del CPU1.

c) En el documento incluir:
- Breve descripción del protocolo MESI y cómo se implementó.
- ¿Cuáles son los beneficios de tener un protocolo de coherencia? ¿El protocolo MESI provee estos beneficios? En caso afirmativo explique cómo.
- Incluya para las últimas 10 instrucciones ejecutadas por cada procesador el estado final de los datos (estado del protocolo MESI).
- ¿Qué debilidades o problemas podría presentar el protocolo de coherencia MESI?

2. Escriba un programa que calcule los primeros 5000 números primos. Se debe medir el tiempo de ejecución del programa.
   a) El programa debe ser escrito en C/C++.
   b) Se debe realizar una versión que utilizando la Criba de Eratóstenes encuentre los números sin utilizar ningún tipo de paralelismo.
      - Mida cuánto dura la ejecución del programa.
   c) Se debe realizar una versión que utilizando la Criba de Eratóstenes encuentre los números utilizando paralelismo. Para esto, se utilizará la librería OpenMPI.
      - Mida cuánto dura la ejecución del programa (el programa debe al menos ser 25% más rápido que ejecutándose serialmente).
      - Conteste las siguientes preguntas:
        - ¿Cuántos procesos diferentes fueron lanzados? ¿Por qué?
        - ¿Cuál estrategia fue usada para paralelizar el programa? ¿Por qué?
        - ¿Es lo máximo que se pudo haber paralelizado? ¿Qué mejoras se le podrían hacer para sacar más provecho del paralelismo?
        - ¿Cuáles dificultades existen para hacer el programa más rápido aún? ¿Cómo se podría lograr?
      - Documente los resultados con capturas de pantalla y las características del sistema donde se corrió (número de CPUs y frecuencia máxima, sistema operativo, cantidad de memoria).
      - Para instalar la librería en Ubuntu:
        - `sudo apt-get install libopenmpi-dev`
3. El código debe ser entregado mediante un repositorio público en GitHub (http://github.com)

a) Se revisará la fecha de los commits de forma tal que cumplan con la fecha de entrega. En caso de que haya un commit posterior a la fecha de entrega, se calificará como si fuese un entregable tardío de la tarea completa.
b) El repositorio deberá tener al menos un commit por cada uno de los miembros del equipo.
c) El repositorio deberá contener un archivo con el nombre README.md que indique cómo configurar (que incluya los requisitos para compilar o ejecutar), compilar (si se necesitara) y ejecutar el código.
d) En caso de que se necesite compilar el código se debe desarrollar un Makefile que lo permita realizar.
e) El repositorio deberá contener un archivo con el nombre LICENSE indicando la licencia del código (GPLv2 recomendado).
f) El código debe estar debidamente documentado.
g) El repositorio NO debe contener archivos binarios.

4. En el sitio del curso se deberá entregar un reporte en formato PDF que incluya:
   a) Portada
   b) Resumen
   c) Introducción
   d) Explicación de alto nivel (sin detalles profundos) de cómo se realizó la implementación.
      - Se debe incluir el enlace del repositorio en GitHub del código.
   e) Tablas, gráficos o ilustraciones reflejen los resultados obtenidos.
   f) Análisis y discusión de los datos.
      - Incluir respuestas a las preguntas planteadas.
   g) Conclusiones
5. Se asume que a este nivel, el estudiante tiene conocimientos de programación, por lo que cualquier refrescamiento del lenguaje y herramientas es responsabilidad del estudiante.
6. Se utilizará como sistema operativo base Ubuntu 14.04 en adelante, por lo que el código y sus instrucciones de construcción deben poder ser ejecutadas en este sistema operativo.