Universidad de Costa Rica
Sede Rodrigo Facio

Escuela de Ciencias de la Computación e Informática

CI1323 - Arquitectura de Computadoras

Prof. Ileana Alpízar

Primera parte del proyecto programado

Juan Carlos Porras Quirós, carné B25201
Dagoberto Quirós Vargas, carné B05051
Pablo Sauma Chacón, carné B16195

San José, Costa Rica
Fecha de entrega: Martes 13 de octubre 2015

Descripción del proyecto

El proyecto completo corresponde al diseño y programación de la simulación de un procesador MIPS para enteros doble núcleo, con la capacidad de ejecutar hilos de un mismo proceso utilizando memoria compartida centralizada. Se pretende diseñar y programar la simulación de un procesador MIPS de doble núcleo (un solo hilo cada uno), el cual debe ser capaz de ejecutar hilos (que por facilidad serán todos del mismo proceso) y que utilizan para su sincronización semáforos también simulados. Memoria compartida centralizada, coherencia de caché con invalidación para escrituras.

Este documento acompaña a la primera parte del proyecto. En esta primera parte se tiene lista la corrida de hilos de un mismo proceso en los dos núcleos, pero aún sin tomar en cuenta los accesos a memoria de datos (i.e., sin el LW, el SW, el LL ni el SC). Esto implica, entre otras cosas: que se tiene funcional el manejo de la caché de instrucciones y la lectura de una instrucción para su ejecución, ejecución de cada una de las instrucciones (excepto las de memoria), manejo de un único reloj (ambos núcleos "están siempre en el mismo ciclo de reloj"), manejo del quantum y cambio de contexto respectivo. Incluye la prevista para dejar de contabilizar quantum si hay acceso a memoria que implique retraso.

Este manual se encuentra adjunto con el código fuente y un ejecutable (con cambio de extensión para que no de error en plataformas de correo electrónico). A continuación se describen las dos maneras de correr el código del proyecto.

Método 1: Ejecución local en Windows

Para la correcta ejecución del proyecto en este método, se debe contar con el JDK 1.7 o superior, y la biblioteca ***pthreads*** de C++ (adjunta con este documento).

Con este documento, se adjunta un archivo de Interfaz.jar. Este archivo ejecutable inicializa la interfaz. Tras haber ingresado los archivos que desea ejecutar en la interfaz (dando click en ***Agregar archivo***, navegando al archivo que quiere, seleccionando, y repitiendo para todos los archivos que irá a “correr”), y haber indicado los parámetros que desea utilizar (si es **modo lento** o **modo rápido**, el **quantum** de procesador, la latencia de memoria **m**, y la duración de transferencia de bus **b**), al darle click en ***Ejecutar hilos***, se iniciará el proceso de ejecución de los hilos que se hayan adjunto.

De haberse seleccionado modo lento, cada vez que se presiona la tecla Enter, se ejecuta un ciclo de reloj, donde se despliega la instrucción ejecutada en ese ciclo, con datos del procesador donde está corriendo, detalles del hilo, y valores de registro.

Método 2: Compilación del cluster Arenal

Para la correcta ejecución de este método, se debe tener acceso al cluster Arenal de la Escuela de Ciencias de la Computación e Informática de la Universidad de Costa Rica, alcanzable en la dirección arenal.ecci.ucr.ac.cr. Se recomienda el acceso mediante WinSCP. Además, la computadora desde donde se ejecute WinSCP, debe tener instalado el complemento de terminar PuTTY.

Tras haber entrado a arenal.ecci.ucr.ac.cr con permisos y cuenta propia, se debe hacer una carpeta, y poner ahí todo el código fuente. Una vez ahí, se puede ejecutar el Main.exe, o compilar un nuevo ejecutable con PuTTY y la instrucción ***g++ -fpermissive -pthread Main.cpp -o Main***. Tras haber generado un nuevo ejecutable (o bien, si se usará el ya compilado), se debe correr con PuTTY con una instrucción de formato ***./nombre modoLento quantum m b archivos***, donde ***nombre*** es el nombre que se le dio al ejecutable, ***quantum***, ***m*** y ***b*** son los valores correspondientes, y ***archivos*** es la dirección de los archivos que se desean correr, separados por espacio.

Por ejemplo, la instrucción ***./Main false 30 1 1 1.txt 2.txt 3.txt 4.txt 5.txt 6.txt*** corre en modo rápido los seis hilos adjuntos (disponibles en la carpeta) en el programa ***Main***, con un quantum de 30, un m de 1, y un b de 1. Al cambiar ***false*** por ***true***, se corre en modo lento.

Problemas sin resolver

Como único problema sin resolver (por motivos de tiempo), queda la implementación del retraso o *wait* entre los fallos de caché. Esto es implementable al indicar en el código que el procesador debe esperar (o gastar) x ciclos mientras para simular el fallo.