Universidad de Costa Rica
Escuela de Ingeniería Eléctrica

IE0521 Estructuras de computadoras digitales II

Proyecto Programado II Parte

Objetivos

Analizar el redimiendo de distintas configuraciones de memoria caché por medio de simulaciones en un lenguaje de alto nivel.

Descripción del Proyecto

Cada grupo de estudiantes deberá realizar un simulador de memoria caché, en C\C++, que permita obtener métricas de rendimiento utilizando un *trace* real (lista de accesos a memoria). Además de obtener estos resultados, los estudiantes deberán realizar pruebas a nivel de unidad utilizando el framework gtest de google, para garantizar el correcto funcionamiento del sistema.

Como punto se partida, se cuenta con el siguiente repositorio de git (cache-git). Dicho repositorio cuenta con las descripciones de las funciones, pruebas, estructuras, cmakes entre otros. La solución propuesta deberá ajustarte a este marco de trabajo y cumplir con el estándar de código establecido en clase.

Características del caché

Se simularan dos optimizaciones de caché: un caché multinivel y un *vitimcache*. El usuario podrá elegir cual de las optimizaciones utilizar para la simulación.

Caché multinivel

Se deberá simular dos niveles de caché: primer nivel (L1) y segundo nivel (L2). El usuario podrá definir el tamaño y la asociatividad de L1, mientras que, el tamaño y asociatividad de L2 serán definidos a partir de las características del caché de primer nivel. El tamaño del caché de segundo nivel deberá ser cuatro veces el tamaño de L1 y su asociatividad se definirá como el doble de la del primer nivel. Además, los niveles L1 y L2 utilizan una política de remplazo LRU, son inclusivos y write-throuth, mientras que entre L2 y memoria se utiliza una política writeback.

VictimCache

El *victimCache* será de 16 entradas totalmente asociativo.

Simulación del trace

El diseño deberá ser parametrizable, es decir, por medio de argumentos se indicará las características del caché a simular.

Los parámetros de entrada serán los siguientes:

1. Tamaño del caché en KB (-t)

2. Tamaño de la línea en bytes (-l)

3. Asociatividad (-a)

4. optimización adicional (-opt)

Como entrada a la simulación se utilizará los *traces* mcf.trace.gz y art.trace.gz que son parte del benchmark SPEC CPU 2006. Estos pueden ser descargados del siguiente enlace: (TraceLink).

El formato del *trace* es el siguiente:

LS	Dirección	IC
# 0	7fffed80	1
# 0	10010000	10
# 0	10010060	3
# 0	10010030	4
# 0	10010004	6
# 0	10010064	3
# 0	10010034	4

- LS : un valor de cero indica un *load* y un uno un *store*.
- Dirección: La direcciones son valores de 8 caracteres en hexadecimal.
- IC: es el número de instrucciones que se ejecutaron entre la referencia a memoria anterior y la actual (contando la instrucción actual).

El programa deberá poder ejecutarse de la siguiente forma:

$$\text{gunzip -c mcf.trace.gz | cache -t} < \# > \text{ -a} < \# > \text{ -l} < \# > \text{ -opt} < vc|l2|none >$$

Para los nuevos parámetros se tiene que:

- opt: indica cual optimización de caché utilizar. En este caso hay tres opciones válidas: *vc* (*victimcache*), *l2* para un caché multinivel y *none* para realizar la simulación sin ninguna optimización.

La salida de la simulación deberá imprimirse en consola, su formato dependerá de la optimización elegida. El formato para los resultados de la simulación del caché multinivel se muestran en la tabla 1. La tabla 2 muestra los resultados esperados para la simulación del *victimcache* y la tabla 3 muestra la salida esperada para una simulación donde la el parámetro de optimización se especifica como *none*.

Pruebas

Además de poder correr sobre un trace real, los bloques funcionales deberán ser probados utilizando el framework gtest de google. La descripción del test plan se encuentra en el siguiente link: (test-plan). Todas las pruebas deberán realizar lo solicitado y retornar los valores esperados.

Tabla 1: Formato para los resultados de simulación de un caché multinivel

Cache parameters:	
L1 Cache Size (KB):	16
L2 Cache Size (KB):	64
Cache L1 Associativity:	2
Cache L2 Associativity:	4
Cache Block Size (bytes):	32
Simulation results:	
Overall miss rate:	0.00
L1 miss rate:	0.00
L2 miss rate:	0.00
Global miss rate	0.00
Misses (L1):	00000
Hits (L1):	00000
Misses (L2):	00000
Hits (L2):	00000
Dirty evictions (L2):	00000

Tabla 2: Formato para los resultados de simulación de un victimcache

Cache parameters:	
L1 Cache Size (KB):	16
Cache L1 Associativity:	2
Cache Block Size (bytes):	32
Simulation results:	
Miss rate (L1+VC):	0.00
Misses (L1+VC):	00000
Hits (L1+VC):	00000
Victim cache hits:	00000
Dirty evictions:	00000

Tabla 3: Formato para los resultados de simulación de un caché básico

Cache parameters:	
L1 Cache Size (KB):	16
Cache L1 Associativity:	2
Cache Block Size (bytes):	32
Simulation results:	
Miss rate(L1):	0.00
Misses (L1):	00000
Hits (L1):	00000
Dirty evictions:	00000

Reporte

Se deberá entregar un reporte en formato pdf discutiendo los beneficios de contar con cachés multinivel y un *victimcache*, no solo a nivel teórico, sino también comparando los resultados de las

simulaciones realizadas.

Evaluación y entrégales

Esta segunda parte del proyecto tiene un valor de un 13 % y se realizará en grupos de 2 personas, como máximo.

Se calificará que el diseño sea parametrizable, que los resultados de la simulación sean correctos y que el programa sea eficiente. Las pruebas deberán estar completas, apegarse al test plan y evidenciar el correcto funcionamiento del programa, un proyecto sin las pruebas tendrá una nota máxima de 60 %.

La entrega se realizará por medio de un repositorio de git, este deberá incluir una sección *Readme* con la descripción del programa, así como las instrucciones o dependencias para ejecutarlo. En mediación virtual, se habilitará un espacio donde los estudiantes deberán adjuntar el reporte y proveer la dirección a su repositorio, antes del día **31 de mayo** a medio día.

Otras consideraciones

- Se asume que el estudiante tiene conocimientos de programación, por lo que cualquier refrescamiento del lenguaje y herramientas es responsabilidad del estudiante.
- Se utilizará como sistema operativo base Ubuntu 16.04 en adelante, por lo que el código y sus instrucciones de construcción deben poder ser ejecutadas en este sistema operativo.
- El código debe ser legible y estar comentado apropiadamente. Elija un formato para los comentarios y un formato para el nombre de las variables. Sea consistente con los formatos elegidos a lo largo de su programa.
- Cada función del programa debe contener un encabezado indicando una descripción general, los parámetros de entrada y los de salida.
- Tal como lo indica la carta al estudiante, la nota máxima para un programa que no compila sera de 30 %.
- Cada día de retraso en la entrega reducirá la nota máxima en 5 %.