Profesores: Carlos González, Leo Medina
Ayudantes: Alexander Palma, Florencia Corvalán, Marco Hernández, Maximiliano Orellana, Ricardo Ruz

Laboratorio 4. Simulador de Caché en un Jupyter Notebook

Objetivos de aprendizaje

- Simular el funcionamiento de una jerarquía memoria para comparar el desempeño de distintas organizaciones de caché utilizando una secuencia determinada de accesos a memoria.
- Comprender mejor el funcionamiento de una jerarquía de memoria compuesta de una memoria principal (*e.g.*, RAM) y varios niveles de caché, poniendo en práctica los conceptos discutidos en clases de mapeo de caché, políticas de reemplazo, métricas de rendimiento, entre otros.
- Familiarizarse con la plataforma Colaboratory de Google, que ofrece espacio de libre acceso para desarrollar "Jupyter Notebooks" con núcleo Python. Esta herramienta de trabajo permite crear y compartir documentos que contienen código ejecutable en un servidor, ecuaciones, gráficos u otro tipo de visualización, resultados numéricos y texto narrativo (*e.g.*, reporte y/o documentación).

Diseño e implementación de un simulador de una jerarquía de memoria

Implemente un simulador de una jerarquía de memoria con 1, 2 o 3 niveles de caché. El simulador debe ser escrito en Python (2 o 3) en la plataforma Colaboratory de Google, a la que puede acceder con sus credenciales Usach.

El simulador debe realizar lo siguiente:

- Leer un archivo de configuración YAML especificando la arquitectura de la memoria caché. Este archivo debe tener los siguientes campos: "arquitectura", "cache_1" y "mem_principal", y en caso de que la caché tenga más de un nivel, debe también incluir los campos "cache_2", para una caché de dos niveles, y "cache_2" y "cache_3", para una caché de tres niveles. La arquitectura debe ser especificada mediante los tamaños de palabra y de bloque en bytes. Además, cada nivel de caché se debe configurar con su número de bloques, su grado de asociatividad (1: mapeo directo) y su tiempo de acceso en ciclos de reloj; y la memoria principal, con su tiempo de acceso en ciclos de reloj. Un ejemplo de archivo de configuración es el siguiente:

Profesores: Carlos González, Leo Medina
Ayudantes: Alexander Palma, Florencia Corvalán, Marco Hernández, Maximiliano Orellana, Ricardo Ruz

```
# Archivo de configuración YAML para el simulador de caché
arquitectura:
  tamano_palabra: 4 #bytes
  tamano_bloque: 16 #bytes

cache_1: #requerido
  bloques: 16
  asociatividad: 2
  tiempo_acceso: 1 #ciclos

cache_2:
  bloques: 64
  asociatividad: 4
  tiempo_acceso: 10 #ciclos

cache_3:
  bloques: 256
  asociatividad: 8
  tiempo_acceso: 100 #ciclos


mem_principal: #requerido
  tiempo_acceso: 1000 #ciclos
```

- Leer un archivo conteniendo una traza de accesos a memoria del tipo “dir, tipo_acceso”, donde “dir” es la dirección de memoria principal en byte expresada en hexadecimal, y “tipo_acceso” indica si el acceso corresponde a una lectura (R) o escritura (W). Un ejemplo de traza es el siguiente:

```
26c1ff40, R
2003fff0, R
26c1ff38, R
308b23f0, R
26c1ff38, R
308b23f0, W
308b23f8, R
26c1ff48, R
308b6ff8, R
308b2400, R
```

- Simular el traspaso de bloques entre los distintos niveles de la jerarquía de memoria de acuerdo con los accesos a memoria indicados en la traza.
- Calcular los tiempos de acceso promedio (AMAT) para cada uno de los niveles de caché y para la memoria principal. HINT: la memoria principal tiene un tiempo de acceso constante indicado en el archivo de configuración.
- Ofrecer la opción de cargar un archivo desde la máquina local para la lectura de archivos.

Para su simulador, asuma lo siguiente:

- Todas las instrucciones son de acceso a memoria, *i.e.*, la cantidad de instrucciones está dada por el número de accesos registrados en la traza.
- El procesador tiene una tasa de reloj de 1 GHz.
- Inicialmente la(s) memoria(s) caché está(n) completamente vacía(s) y la memoria principal está completamente llena con 0's.
- Para un acceso a memoria de tipo escritura, siempre se escribirá un 1 y se utilizará la técnica de "write-through".

Primera prueba del simulador

Como resultado de una primera simulación, en el mismo Jupyter Notebook, Ud. debe mostrar lo siguiente:

- Una representación gráfica del estado final de la(s) memoria(s) caché, mostrando para cada conjunto y vía el dato que contiene, *i.e.,* "0", "1" o "vacío", y el número de bloque, en decimal, de la memoria principal que quedó almacenado en la caché. Para una caché con más de 8 conjuntos o más de 8 vías, muestre solamente los primeros 4 y los últimos 4 conjuntos o vías, respectivamente. Sugerencia: utilice colores y, por ejemplo, pinte los espacios que contienen un "0" de verde, los que contienen un "1", de azul, y los que están vacíos, de rojo (en tal caso, habría que incluir una leyenda que indique qué significa cada color).
- Un gráfico con los AMATs de cada nivel de la jerarquía de memoria.

Medición de desempeño

Utilice el simulador para comparar el desempeño de distintas jerarquías de memoria. Para esto, Ud. debe desarrollar en el mismo Jupyter Notebook lo siguiente:

- Para una jerarquía de memoria con sólo un nivel de caché, determine el efecto del tamaño del bloque y del grado de asociatividad en el rendimiento del sistema. Para ello, en su Jupyter Notebook se debe leer un archivo conteniendo una traza de accesos a memoria y graficar:
  i) La tasa de desaciertos de caché en función del tamaño del bloque para 4 tipos de mapeos de caché: directo, asociativo de 2 vías, asociativo de 4 vías y completamente asociativo.
  ii) La tasa de desaciertos de caché en función del grado de asociatividad desde 1 (mapeo directo) hasta completamente asociativo para 4 tamaños de bloque (en palabras): 1, 2, 4 y 8.

- Determinar la aceleración lograda al incorporar un segundo y un tercer nivel de caché. Para ello, en su Jupyter Notebook se debe leer un archivo conteniendo una traza de accesos a memoria, calcular los CPI para el caso con un nivel de caché, con dos niveles de caché y con tres niveles de caché, y graficar la aceleración lograda en los dos últimos casos para tres configuraciones de caché de su elección que queden claramente identificadas en su documentación.

Profesores: Carlos González, Leo Medina
Ayudantes: Alexander Palma, Florencia Corvalán, Marco Hernández, Maximiliano Orellana, Ricardo Ruz

Exigencias

- El programa debe ser entregado a través del Classroom en dos copias: 1) link al Jupyter Notebook de Colaboratory y 2) archivo Jupyter Notebook .ipynb.
- El Jupyter Notebook debe permitir al usuario cargar los archivos necesarios (configuración y traza) desde su máquina local.
- El programa debe cumplir con un mínimo de calidad de software, esto es, funciones separadas, y nombres, tanto de funciones como de variables, de fácil comprensión.
- La documentación debe estar incluida en el mismo Jupyter Notebook e incluir un muy breve informe del trabajo realizado, incluyendo i) explicación de cómo se resolvió el problema, y ii) conclusiones.

Evaluación

- La nota del laboratorio será el promedio simple del código fuente, los gráficos y la documentación presentada en el mismo Jupyter Notebook.

Entrega

- Este laboratorio debe ser entregado a más tardar el día viernes 20 de agosto de 2021, hasta las 23:59 hrs.