Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (II/2018)

Tarea 8

Fecha de entrega: miércoles 7 de noviembre de 2018 a las 11:59 AM

Parte programación

En esta tarea, tendrán la oportunidad de experimentar con el diseño de un sistema de multiprogramación y jerarquía de memoria, desde el punto de vista de la memoria y de la CPU. Más específicamente, deberán programar la simulación de un acceso completo a una dirección de memoria virtual en un sistema que incluye una `TLB` y una *caché* y que se ve enfrentado a la ejecución de diversos programas que realizan peticiones a memoria. Este sistema, además, hace uso de distintas políticas de reemplazo y funciones de correspondencia, en particular:

- **Política de reemplazo páginas:** `LRU`.
- **Función de correspondencia** `TLB`**:** *fully associative*.
- **Política de reemplazo** `TLB`**:** `LRU`
- **Función de correspondencia *caché*:** *N-way associative* parametrizada[1].
- **Política de reemplazo *caché*:** `LFU` (con desempate mediante `FIFO`).

Input

La tarea desarrollada debe recibir como *input* las especificación completa de la memoria virtual junto con las instrucciones realizadas por cada programa.

Los *inputs* vendrán en el siguiente orden:

- **MEM_FIS_SIZE:** Indica el tamaño en bytes de la memoria física de datos.
- **MEM_VIR_SIZE:** Indica el tamaño en bytes de la memoria virtual de datos.
- **PAGE_SIZE:** Indica el tamaño en bytes de cada página.
- **TLB_LINE:** Indica el número de líneas de la `TLB`.

[1] Es decir, se recibe como parámetro el valor de *N*.

- **CACHE_SIZE:** Indica el tamaño en bytes de la memoria *caché*.
- **CACHE_LINE:** Indica el número de líneas de la memoria *caché*.
- **N_WAY_CACHE:** Indica el valor de N para la función de correspondencia de la *caché*.
- **NUM_PROG:** Indica el número de programas a ejecutar.
- **PROG:** Una lista de lista, donde cada lista interior representa un programa, el que indica una secuencia ordenada de largo arbitrario para acceder a datos de memoria. El primer elemento de la lista representa el programa 1, el segundo elemento al programa 2 y así sucesivamente.

A continuación, se muestra el código necesario para leer las especificaciones de la *caché* y memoria virtual.

```
# Este código debe estar al inicio de su programa
import json

MEM_FIS_SIZE = int(input())
MEM_VIR_SIZE = int(input())
PAGE_SIZE = int(input())
TLB_LINE = int(input())
CACHE_SIZE = int(input())
CACHE_LINE = int(input())
N_WAY_CACHE = int(input())
NUM_PROG = int(input())
PROG = json.loads(input())
```

Un ejemplo de los *inputs* usando el código anterior es el siguiente:

```
512
1024
32
4
128
16
4
4
"[[1,2,4,6,2,-1,4,7,3,5,-1,2,1],[6,4,2,5,7,8,5,4,5,-1,5],[1,4,5,6,7],[1,2,3,4,5,6,3]]"
```

Debe tomar las siguientes consideraciones:

- Existe una palabra por línea en la `TLB`.
- Las tablas de páginas no utilizan espacio en memoria.
- El cambio de contexto (*scheduling*) será cooperativo, en donde cada ***yield*** que será representado con los accesos a memoria "-1". Por ejemplo, PROG 1 tiene dos ***yield*** mientras que PROG 2 tiene un ***yield*** y los otros dos programas no tienen ninguno.
- Al ocurrir un cambio de contexto, todas las líneas de la `TLB` y la *caché* se deben invalidar.
- Cada palabra de la `TLB` es una entrada de página.
- Los procesos no tienen páginas asignadas cuando empiezan a ejecutarse (todas son inválidas).
- Solo se considera `HIT` en la `TLB` cuando la página a la que se accede es válida.

- La capacidad de almacenamiento del *swap file* es infinita.
- El orden de ejecución de los programas está dado por el número del programa.
- El *input* solo serán valores definidos anteriormente. En ningún momento se ingresarán números o datos que no corresponden a la propiedad definida.
- Todos los tamaños, número de palabras o bloques, e incluso el parámetro N serán potencias de 2 mayores a 1. Además, estos son **coherentes**, por ejemplo, no se entregará un *input* donde el tamaño de página sea mayor a la memoria física.

Output

Como salida, la tarea debe imprimir **exactamente** los siguientes valores **por programa**:

- Número de *hits* a la `TLB`. El formato es: "`hit TLB: XX`" donde `XX` es el número de *hits*.
- Número de *hits* a la *caché*. El formato es: "`hit cache:  XX`" donde `XX` es el número de *hits*.
- Número de *page faults*. El formato es: "`page fault:  XX`" donde `XX` es el número de *page faults*.
- Número de *swap in*. El formato es: "`swap in:  XX`" donde `XX` es el número de *swap ins*.
- Número de *swap out*. El formato es: "`swap in:  XX`" donde `XX` es el número de *swap outs*.
- Número de páginas válidas. El formato es: "`page valid:  XX`" donde `XX` es el número de páginas válidas.
- Número de páginas en disco. El formato es: "`page disk:  XX`" donde `XX` es el número de páginas en disco.

Un ejemplo del formato de *output* para 2 programas:

```
PROGRAMA 1
hit TLB: 5
hit cache: 3
page fault: 4
swap in: 4
swap out: 5
page valid: 3
page disk: 1

PROGRAMA 2
hit TLB: 0
hit cache: 0
page fault: 2
swap in: 1
swap out: 2
page valid: 3
page disk: 1
```

Debe tomar las siguientes consideraciones al momento de entregar el *output*.

- Debe imprimir los datos en el mismo orden del PDF y con el formato solicitado (incluir un salto de línea entre un programa y otro).
- No debe imprimir nada durante la ejecución del programa.

Supuestos

Puede hacer uso de **supuestos** en casos límite o elementos que no hayan sido explicitados en el enunciado. No obstante, para que estos sean válidos durante la corrección, **debe** explicitarlo en su `README`.

No se aceptarán supuestos sobre criterios que hayan sido establecidos en el enunciado.

Entrega y evaluación

La tarea se debe realizar de **manera individual** y se entregará a través de GitHub. El formato de entrega corresponde a un *script* en Python 3.5 ó 3.6 llamado `memory_simulator.py`, que se debe encontrar en **la raíz del repositorio** junto con todos los archivos que sean necesarios para que su tarea funcione.

El repositorio subido debe contar con un archivo `README.md` escrito en *Markdown* que identifique sus datos y consideraciones que el corrector deba tomar en cuenta, tales como el sistema operativo usado para realizar la tarea, versión en Python en la que fue programado el *script*, entre otras. El `README` se puede subir hasta 24 horas después de la entrega. Si lo sube posterior al plazo de entrega establecido, debe crear una *issue* en el repositorio de su tarea indicándolo para que sea considerado en la corrección. Los archivos que no ejecuten o que no cumplan el formato de entrega establecido implicarán nota **1.0** en la tarea. En caso de atraso, se aplicará un descuento de **1.0** punto por cada 6 horas o fracción.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno (grupo) copia un trabajo, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir reprobación del curso y un procedimiento sumario. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.