**UNIVERSIDAD DE SANTIAGO DE CHILE**
**FACULTAD DE INGENIERÍA**
**DEPARTAMENTO DE INGENIERÍA INFORMÁTICA**


# Laboratorio N 3

Integrantes: Benjamín Muñoz Tapia

Curso: Organización de Computadores

Sección: 0-L-1

Profesor(a): Leonel Medina Daza

24 de Agosto de 2018

# Tabla de contenidos

# 1. Introducción

## 1.1. Problema

Para este trabajo el problema principal erradica en la aplicación del conocimiento teórico sobre memoria caché según entradas determinadas. El estudiante debe ser capaz de aplicar las políticas de reemplazo y estructurar bien los distintos caché posibles, para entregar el mejor, es decir, el que tenga mayor tasa de hit, con su respectiva política de reemplazo, junto con el estado actual de esta después de la última consulta.

## 1.2. Motivación

A los estudiantes del Departamento de Ingeniería Informática de la Universidad de Santiago de Chile, se les pide realizar este proyecto con el fin de lograr aplicar los conocimientos de la asignatura Organización de Computadores. La motivación principal de dicho trabajo puede entenderse como la buena realización del trabajo, con el fin de asemejar los conocimientos de la asignatura, y obtener los aprendizajes esperados, para así poder avanzar tanto en la carrera como en el curso, más específicamente con memoria caché, políticas de reemplazo y tipos de mapeo.

## 1.3. Objetivos

### 1.3.1. Objetivo general

Como objetivo general se tiene la resolución del problema presentado en el enunciado del laboratorio mediante lenguaje C y entregar el archivo de salida correspondiente con la solución.

### 1.3.2. Objetivos específicos

1. Leer y manejar bien los datos para su uso, mediante el ingreso de estos por consola
2. Dominar los conceptos de memoria caché para lograr implementarlos bien mediante estructuras o herramientas propias del lenguaje C

3. Realizar las distintas políticas de reemplazo para poder comparar las distintas tasas de hit y miss.

4. Junto con lo anterior, obtener la solución óptima y entregarla mediante un archivo de texto

## 1.4. Propuesta de solución

Se propone recibir los datos mediante consola, ya sea con la librería unistd, o con el argumento argv del método main. Por otra parte se pretende simular la caché por partes, es decir, vías, bloques y palabras; lo cual se hará mediante estructuras para crear distintos tipos de datos y ayudar al manejo de estos. Una vez creada la caché, se simularán los distintos casos con las políticas de reemplazo y se almacenarán los distintos resultados de hit y miss para entregar los mejores, según lo pedido.

## 1.5. Herramientas

1. Estructuras Una estructura es una herramienta propia de C que permite moldear distintos tipos de datos y adaptarse así a distintos tipos de problemas, encontrando sus respectivas soluciones. Su principal característica es poder contener dentro de sí misma cualquier tipo de dato conocido, incluyendo a la misma estructura.

2. Getopt Es una herramienta de la librería unistd.h que permite obtener los parámetros por consola junto con banderas, lo cual es requerimiento explícito del enunciado.

## 1.6. Estructura del informe

El presente informe pretende describir el desarrollo de este trabajo explicando la metodología de este a través de un análisis breve del código, sin entrar en detalles de programación, sino que en algoritmos. Luego se harán pruebas para ver la ejecución del programa y examinar el resultado de estas respecto a lo propuesto y el conocimiento ya existente de caché. Finalmente se hará una conclusión del trabajo, donde se verá el cumplimiento de objetivos y la importancia de este.

# 2. Marco teórico

1. Memoria Caché

   Es un tipo de memoria volátil, que está entre el procesador y la memoria principal, donde esta es de rápido acceso y sirve para almacenar datos que requerirá usar la CPU evitando acceder a la memoria RAM, ahorrando ciclos de reloj.

2. Política de Reemplazo

   Son técnicas que permiten mejorar el rendimiento de la caché a medida que se van realizando consultas por los datos. Para este laboratorio se aplicarán las ténicas FIFO, LIFO, LRU Y MRU. Estas técnicas buscan obtener la mejor tasa de hits, el cual es el objetivo principal de este trabajo para las distintas combinaciones de caché.

3. Hit y Miss

   Hit es cuando se encuentra un dato al realizar un mapeo en la caché, mientras que miss es cuando no se encuentra el dato, y según la política de reemplazo se inserta en la caché. También es posible calcular la tasa de hits y de miss, donde se divide cada uno de estos por la cantidad de mapeos totales.

# 3. Desarollo

Para comenzar el desarrollo del programa se necesitan los parámetros iniciales, que se obtienen con el comando getopt de la libreria unistd.h, con la cual mediante banderas se indica el parámetro ingresado, la primera que es -n para indicar el nombre del archivo con los datos, la segunda -m para indicar el tamaño de los datos en bytes, y la tercera -p para indicar cuantas palabras por bloque hay. Una vez obtenidos estos parámetros lo primero es leer el archivo y obtener los datos en un arreglo de enteros junto con la cantidad de estos mediante un contador. Luego se hace un arreglo de caché según la cantidad de bloques que son calculados con la función cantidadBloques, y esto permite tener el tamaño total de datos y se le asignan vías en orden de potencias de 2 para así tener todos los casos de vías posibles.

Una vez obtenidas las distintas caché con sus respectivos números de vías, se crea una matriz de enteros con los hit, donde cada fila posee cuatro columnas, una para cada política por caché. Luego se comienza a recorrer cada cache, y para cada una se comienzan a aplicar las políticas de reemplazo, y después de aplicar cada una se limpia la caché para poder usar la siguiente política desde cero con la misma caché.

Finalmente se recorren los hits buscando el máximo de estos, y se imprime en el primer archivo de salida cada caché que posea esa cantidad de hits, junto con el grado de asociatividad y el numero de miss. Por otro lado en el segundo archivo de prueba se escriben todas las mejores caché especificando los datos que posee.

Con respecto a las políticas de reemplazo se tienen:

1. FIFO

   First in, First Out. En esta política se busca si el dato está en la caché o no. Si este dato está se suma 1 a los hits, y sino, se inserta después de aumentar en 1 los miss.

2. LIFO

   Last in, Last Out. Esta política inserta los elementos al principio igual que FIFO, pero una vez llena la caché solo cambia el último insertado.

3. MRU

   Most Recently Used. Para esta política se cuenta con una variable auxiliar llamada

MRU en la estructura vía. Al comienzo se llena igual que en FIFO, pero una vez llena, se busca si el dato está o no en la caché, pero para ambos casos, se actualiza el índice del bloque que es el último que se usó, cosa de que en caso de haber miss, se inserte en dicho bloque.

4. LRU

   Least Recently Used. En esta política se verifica primero si el caché esta lleno o no.

   - Si no está lleno: se ve si el dato está o no en la caché, si es así se suma un hit y se suma a todos los bloques en uno su variable LRU, excepto al que fue insertado que se asigna en 0. Si el dato no se encuentra, se procede a aumentar en uno los miss e insertar el dato aumentando en uno los LRU de cada bloque excepto el actual.
   - Si esta lleno: se procede a buscar el valor en la caché. Si este se encuentra se suma uno en hit y se realiza el aumento de la variable LRU para cada bloque, haciendo 0 el bloque actual. Por otro lado si no está lleno se inserta el dato en el bloque que tenga la mayor variable LRU, es decir, el que menos se ha ocupado.

# 4. Experimentos a realizar

## 4.1. Resultados obtenidos

Se utilizarán dos archivos de prueba, y por términos de espacio, no se mostrarán todas las combinaciones de caché, ni el estado completo de cada caché del segundo archivo de salida. En ambos casos el tamaño de datos es 4096, y se usan 2 palabras por bloque.

1. Prueba 1

```
25
12
35
36
21
58
112
20
6168
4611
4120
6168
```

Figura 1: Prueba 1 de datos de caché.

```
-----------
Grado de asociatividad:  2
Numero de Hits:  2
Numero de Miss: 10
Tecnica de Reemplazo:  MRU
-----------
Grado de asociatividad:  2
Numero de Hits:  2
Numero de Miss: 10
Tecnica de Reemplazo:  LRU
-----------
Grado de asociatividad:  4
Numero de Hits:  2
Numero de Miss: 10
Tecnica de Reemplazo:  FIFO
-----------
Grado de asociatividad:  4
Numero de Hits:  2
Numero de Miss: 10
Tecnica de Reemplazo:  LIFO
-----------
```

Figura 2: Mejores caché para el ejemplo 1.

2. Prueba 2

```
via[0] bloque[29] palabra[1]: 35
via[0] bloque[30] palabra[0]: 36
via[0] bloque[30] palabra[1]: 37
via[0] bloque[31] palabra[0]: 20
via[0] bloque[31] palabra[1]: 21
via[0] bloque[32] palabra[0]: 58
via[0] bloque[32] palabra[1]: 59
via[0] bloque[33] palabra[0]: 112
via[0] bloque[33] palabra[1]: 113
via[0] bloque[34] palabra[0]: 6168
via[0] bloque[34] palabra[1]: 6169
via[0] bloque[35] palabra[0]: 4610
via[0] bloque[35] palabra[1]: 4611
via[0] bloque[36] palabra[0]: 4120
via[0] bloque[36] palabra[1]: 4121
```

Figura 3: Estado final de caché para el ejemplo 1.

```
15
32
25
24
9
12
10
11
30
32
16
19
21
24
18
20
2
9
7
1
10
```

Figura 4: Prueba 2 de código MIPS.

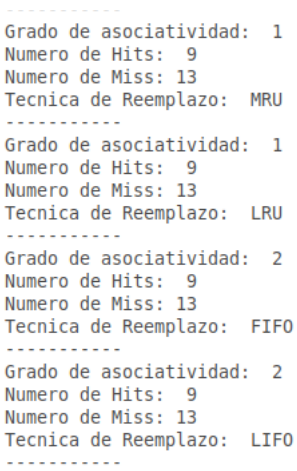


Figura 5: Mejores caché para el ejemplo 2.

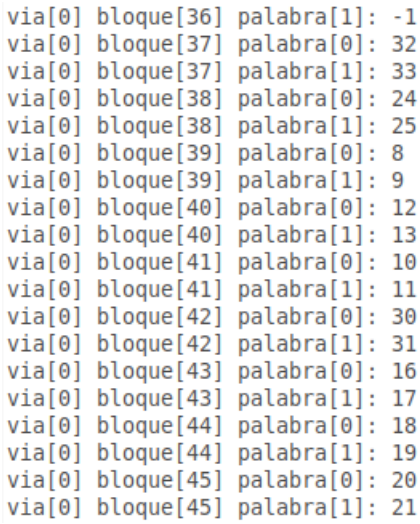


Figura 6: Estado de caché para el ejemplo 2.

## 4.2. Análisis de resultados

Puede apreciarse con esto la exitosa implementación de la caché, obteniendo resultados concretos en ambos archivos de texto. El resultado obtenido más que de la política de reemplazo, se ve que depende de la cantidad de palabras por bloque y el tamaño de los datos ingresados, ya que, al aumentar la cantidad de palabras por bloque, se observa un mayor numero de hits que en los otros casos debido a que disminuyen los bloques, y por lo tanto, hay menos espacio donde direccionar, produciendo mayor competencia.

# 5. Conclusiones

Viendo el desarrollo de este trabajo y correctos resultados de este, se puede observar un cumplimiento de los objetivos específicos dando así un cumplimiento del objetivo general de este trabajo. También se logra ver lo dinámica que es la caché al momento de manipular datos, tal como se vio en las clases de teoría y se afirmaba lo volátil que era para almacenar y compartir información. Puede decirse que este trabajo ayudó a entender de mejor forma el funcionamiento de la caché, que es la memoria de más rápido acceso de las que tienen los computadores, y sin esta, no se podrían realizar los procesos que hacen las máquinas actuales a la velocidad de nanosegundos, porque la caché es la responsable del rápido manejo y acceso a la información de la memoria, sin restarle claro, importancia a la memoria principal.