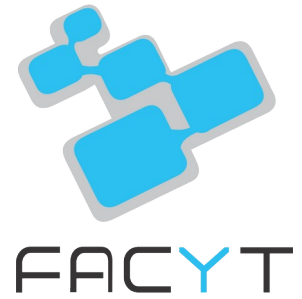




Universidad de Carabobo  
Facultad Experimental de Ciencias y Tecnología  
Departamento de Computación  
Arquitectura del Computador



# Proyecto Simulador de Memoria Caché

Juan Yaguaro  
27.536.231

Valencia, enero de 2021

# Introducción

El presente proyecto tiene como principal intención simular el comportamiento de la memoria caché dependiente de una configuración dada y un conjunto de direcciones de memoria a almacenar.

La memoria caché es un tipo de memoria volátil (a menudo de pequeña capacidad), la cual provee de altas velocidades de acceso al procesador y almacena los datos atómicos más frecuentemente utilizados por la unidad de procesamiento.

Debido a su característica de memoria temporal, en la cual los datos menos frecuentemente accedidos son eliminados con el tiempo, le permite garantizar el acceso a datos al procesador de manera rápida y eficiente.

# Simulador de Memoria Caché

El programa principal “cachesim” realiza la simulación de memoria caché. Este programa fue realizado bajo el estándar ISO C++17, aprovechándose de las distintas clases y funcionalidades que ofrece la librería estándar de C++.

El programa recibe distintos argumentos como modificadores a su funcionalidad. Principalmente recibe 3 argumentos necesarios los cuales son:

- i. Nombre de archivo de configuración, en donde se encuentra la configuración solicitada para el simulador. La estructura de este archivo es la siguiente:

Tamaño de caché en bytes (Entero potencia de 2).

Tipo de caché (0 para cache directa, 1 para cache asociativa por conjuntos).

Tamaño de línea de caché en bytes (Entero potencia de 2, no mayor al tamaño de caché).

Política de reemplazo (0 para algoritmo LRU, 1 para algoritmo MRU).

- ii. Nombre de archivo de datos, en donde se encuentran las direcciones que se desean almacenar en el simulador. La estructura de este archivo es la siguiente:

*Dirección 0 (Numero natural).*

*Dirección 1 “”.*

.

.

.

*Dirección n.*

- iii. Nombre de archivo de salida (opcional) en donde se especifica en donde se desea mostrar la informacion de salida. La estructura de este archivo es la siguiente:

*Leyenda*

*{Dirección 0} {Acierto/Fallo} {ID de conjunto} {Contenido en cache}  
{Nuevo contenido en cache}*

.

.

.

*{Dirección n} ""*

Adicionalmente, existen distintas opciones que se pueden activar para conocer información del programa o modificar el formato de salida:

- i. Version del programa, donde se imprime por salida estándar la versión actual del simulador de caché.
- ii. Ayuda del programa, donde se imprime por salida estándar la forma de utilizar el programa y enlaces hacia documentación.
- iii. Opción Hexadecimal, donde los valores de las direcciones se muestran en su valor hexadecimal correspondiente.

La dirección límite a representar se establece como  $2^{16}$  para así simular el comportamiento de la caché basado en una memoria principal de 16 bits.

## Diseño del programa

El simulador de memoria caché “cachesim” se compone de 3 clases dentro del namespace cachesim, las cuales son:

- i. *cachesim::cache*: clase abstracta que almacena los valores de configuración obtenidos. Se encuentra en el header `<cachesim/cache_h>`.
- ii. *cachesim::direct\_cache*: clase que hereda de *cachesim::cache*, la cual simula el comportamiento de una memoria caché de tipo directa. Se encuentra en el header `<cachesim/direct_cache.h>`
- iii. *cachesim::set\_associative\_cache*: clase que hereda de *cachesim::cache*, la cual simula el comportamiento de una memoria caché de tipo asociativa por conjuntos. Se encuentra en el header `<cachesim/set_associative_cache.h>`.

Estas clases estan diseñadas en base a la herencia y el polimorfismo, con la finalidad de promover una única interfaz de métodos públicos para todas las definiciones particulares de clase.

Esto es posible gracias a las distintas técnicas de programación orientada a objetos disponibles dentro del lenguaje de programación C++, en el cual se permite crear instancias de clases derivadas con apuntadores hacia la clase base. Dicha técnica se conoce como resolución de tipo en tiempo de ejecución (RTTI por sus siglas en inglés).

Se promueve el uso de funciones virtuales en la clase base (*cachesim::cache*), para así sobrescribir el comportamiento dependiendo de la necesidad de cada clase. También se definen aquellas funciones virtuales como abstractas (conjunto a identificador =0), para dejar explícitamente señalado al compilador que la clase base no puede tener instancias directas.

En el caso del contenedor de elementos, este se define en cada una de las clases derivadas para promover una correcta especialización sin necesidad de crear clases de tipo plantilla. Sus especializaciones son las siguientes:

- i. *cachesim::direct\_cache* utiliza un contenedor `std::vector<int>` con un alias identificado como *cache\_set*.
- ii. *cachesim::set\_associative\_cache* utiliza un `std::vector<cache_set>`, sacando provecho al alias utilizado en *cachesim::direct\_cache*.

La visualización gráfica de la estructura de las clases sería la siguiente:

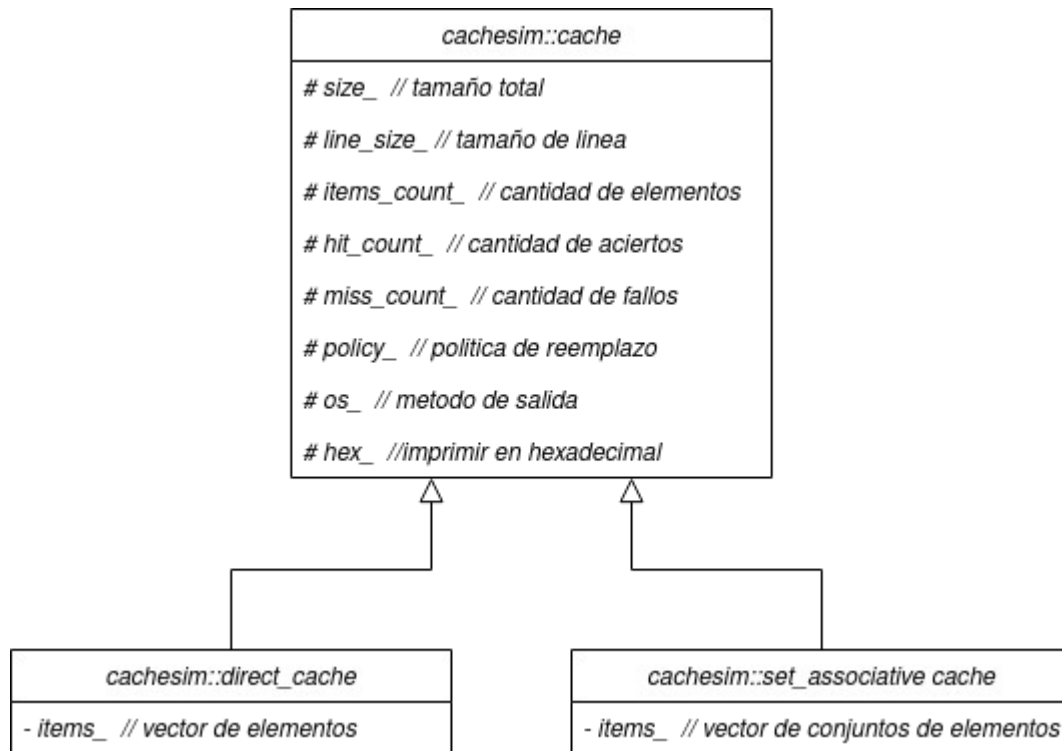


Imagen 1. Diagrama de clases de cachesim.

Los principales métodos de la clase son aquellos de acceso público, como *allocate(const int& value)*, el cual almacena un valor en el simulador memoria cache. Se encuentran presentes distintos metodos de acceso como *size()*, *line\_size()*, entre otros para obtener el estado del simulador.

Existe un enumerador *emplace\_policy* el cual registra el tipo de política de reemplazo a utilizar (algoritmos LRU y MRU). Se encuentra en el header *<cachesim/cache.h>*.

Se crearon los headers *<cachesim/version.h>* y *<cachesim/prefix.h>* para almacenar información referente a la versión del programa y las opciones disponibles, incluyendo tambien *<cachesim/eror.h>* para mensajes de error.

Tambien se encuentra el header *<cachesim/cache.h>* como header público para la inclusión de todas las clases del simulador en una sola línea.

El namespace *cachesim* se compone a su vez de dos sub namespaces llamados *error* (donde se encuentran los mensajes de error) y *limits* (donde se encuentran los valores límites del simulador).

Los headers de librería estándar utilizados son `<vector>`, `<algorithm>`, `<stdexcept>`, `<fstream>`, `<iomanip>`, `<iostream>`, `<memory>`, `<string>`, `<string_view>`, `<cmath>`, `<random>`, `<cstdint>`.

Las excepciones son utilizadas para determinar si los tamaños de caché y línea de caché leídos son válidos, es decir, que ambos sean potencias de 2 y que el tamaño de línea de caché sea menor o igual al tamaño de caché.

Se implementan distintas técnicas modernas de programación en C++ como inicialización por lista en constructores, RTTI, RAII, conversión estática, apuntadores inteligentes, evaluación de constantes en tiempo de compilación, entre otras.

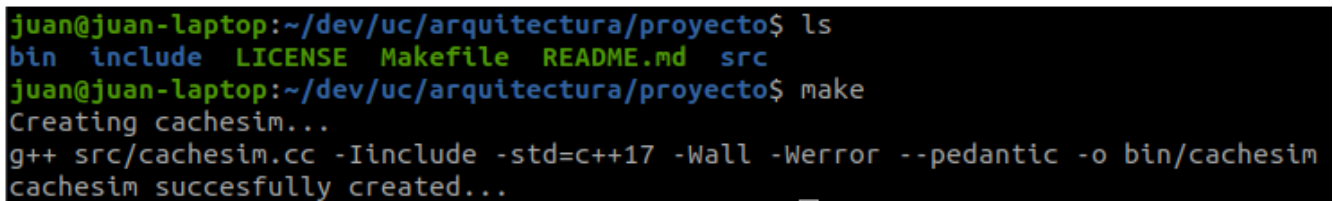
Entre las técnicas clásicas de orientación a objetos, se promueve el encapsulamiento, la herencia y el polimorfismo, permitiendo adoptar el tipo de clase a utilizar en tiempo de ejecución dependiendo de la configuración recibida.

# Instalación

Los requisitos para completar la instalación del simulador cachesim incluyen GNU Make (versión 4.0.0 o superior) y GNU Compiler Collection (versión 8.0.0 o superior).

Para su instalación se debe ejecutar el comando *make* desde la carpeta raíz del proyecto, donde se encuentra el archivo *Makefile*.

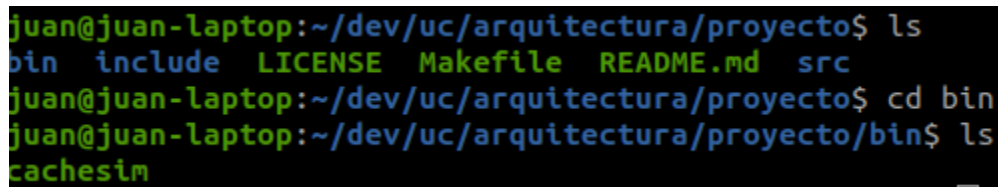
Una compilación e instalación satisfactoria de cachesim imprimiría por consola los siguientes mensajes:

A terminal window with a black background and green text. The prompt is 'juan@juan-laptop:~/dev/uc/arquitectura/proyecto\$'. The first command is 'ls', showing 'bin include LICENSE Makefile README.md src'. The second command is 'make', which outputs 'Creating cachesim...' followed by the compilation command 'g++ src/cachesim.cc -Iinclude -std=c++17 -Wall -Werror --pedantic -o bin/cachesim' and finally 'cachesim succesfully created...'.

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto$ ls
bin include LICENSE Makefile README.md src
juan@juan-laptop:~/dev/uc/arquitectura/proyecto$ make
Creating cachesim...
g++ src/cachesim.cc -Iinclude -std=c++17 -Wall -Werror --pedantic -o bin/cachesim
cachesim succesfully created...
```

Imagen 2. Mensajes de compilación exitosa de cachesim. Importante identificar inicialmente que se encuentra en la carpeta raíz del proyecto.

Luego de esto, el archivo ejecutable se encontrará dentro de la carpeta *bin*, por lo que debe de accederse a ella antes de ejecutar el programa.

A terminal window with a black background and green text. The prompt is 'juan@juan-laptop:~/dev/uc/arquitectura/proyecto\$'. The first command is 'ls', showing 'bin include LICENSE Makefile README.md src'. The second command is 'cd bin', changing the directory. The third command is 'ls', showing 'cachesim'.

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto$ ls
bin include LICENSE Makefile README.md src
juan@juan-laptop:~/dev/uc/arquitectura/proyecto$ cd bin
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ls
cachesim
```

Imagen 3. Desplazamiento a carpeta bin desde la carpeta raíz.

Al realizar estos pasos con éxito, ya se encuentra listo para ejecutar el simulador cachesim desde su computadora.



## Ejemplos de uso

Existen distintas interacciones posibles dentro de cachesim. Inicialmente, si se ejecuta el programa sin argumentos de entrada, este mostrará el mensaje por defecto como se muestra a continuación:

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim
cachesim - simulates cache behavior with given configuration and values to allocate.
Try [cachesim -h] or [cachesim --help] for help.
```

Imagen 4. cachesim siendo ejecutado sin argumentos de entrada.

En este caso, el programa nos recomienda utilizarlo el argumento `-h` o `-help` para conocer más al respecto. Ambos muestran el mismo mensaje, el cual es el siguiente:

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim -h
Usage: cachesim -c=[FILENAME] -d=[FILENAME] -o=[FILENAME] -[OPTION]
    -c=[FILENAME]      filename for config file.
    -d=[FILENAME]      filename for data file.
    -o=[FILENAME]      filename for output file (default value is std::cout).
    -x                output hex values of directions.
    -h, --help         display all available commands.
    -v, --version      display version of test_generator.
Full documentation at: <https://github.com/juanyaguaro/cachesim>.
Or available locally in <docs> folder.
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim --help
Usage: cachesim -c=[FILENAME] -d=[FILENAME] -o=[FILENAME] -[OPTION]
    -c=[FILENAME]      filename for config file.
    -d=[FILENAME]      filename for data file.
    -o=[FILENAME]      filename for output file (default value is std::cout).
    -x                output hex values of directions.
    -h, --help         display all available commands.
    -v, --version      display version of test_generator.
Full documentation at: <https://github.com/juanyaguaro/cachesim>.
Or available locally in <docs> folder.
```

Imagen 5. cachesim siendo ejecutado con el argumento de entrada `-h` o `--help`.

Este argumento de entrada muestra un mensaje más descriptivo acerca de las opciones disponibles en cachesim.

Principalmente nos indica que requiere de los argumentos `-c` y `-d`, los argumentos `-o` y `-x` son opcionales. A continuación se define cada uno de los argumentos de entrada:

- i. `-c=[NOMBRE_DE_ARCHIVO]`: indica el nombre del archivo de configuración del simulador. Es un argumento requerido.

- ii. `-d=[NOMBRE_DE_ARCHIVO]`: indica el nombre del archivo de datos a introducir en el simulador. Es un argumento requerido.
- iii. `-o=[NOMBRE_DE_ARCHIVO]`: indica el nombre del archivo donde se guardarán los resultados del programa. Es un argumento opcional, de no encontrarse presente se imprimirán los resultados por consola.
- iv. `-x`: indica que los datos se imprimirán en su representación hexadecimal. Es un argumento opcional, de no encontrarse presente se imprimirán los valores en su representación decimal.

Adicionalmente se encuentra el argumento de entrada `-v` o `-version`, el cual imprime el siguiente mensaje a consola indicando la versión del simulador `cachesim`:



```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim -v
cachesim 0.2.0
Copyright 2020 Juan Yaguaró.
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim --version
cachesim 0.2.0
Copyright 2020 Juan Yaguaró.
```

Imagen 6. `cachesim` siendo ejecutado con el argumento de entrada `-v` o `--version`.

Realizando una demostración, si se tienen dos archivos llamados `c.in` y `d.in` para los archivos de entrada de configuración y datos respectivamente, también deseando que se guarden los resultados en un archivo de salida `o.out`, sin mostrar los valores en hexadecimal, el siguiente comando realizaría dicha tarea:

```
./cachesim -c=c.in -d=d.in -o=o.out
```

Es importante señalar que, en `cachesim` el orden de los argumentos de entrada es irrelevante, por lo tanto, la siguiente expresión sería equivalente a la expresión anterior:

```
./cachesim -d=d.in -c=c.in -o=o.out
```

La única limitación sería que los argumentos de entrada `-h`, `--help`, `-v` y `-version` solo podrán ser agregados como argumentos únicos en el comando realizado, de lo contrario serán ignorados. Por lo tanto, la siguiente entrada se consideraría inválida:

```
./cachesim -v -h -c=c.in -d=d.in
```

# Generación de casos de pruebas

Para la evaluación de pruebas, se optó por realizar pruebas de tipo manual y automáticas.

Dentro de cachesim se encuentra un pequeño programa adicional llamado `test_generator`, el cual se encarga de generar casos de pruebas en archivos de configuración y datos.

Los requerimientos de instalación para `test_generator` son los mismos que para `cachesim`, y para instalarlo se debe ejecutar el comando `make test_generator` desde la carpeta raíz como se muestra a continuación:

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto$ ls
bin include LICENSE Makefile README.md src
juan@juan-laptop:~/dev/uc/arquitectura/proyecto$ make test_generator
Creating test_generator...
g++ src/test_generator.cc -Iinclude -std=c++17 -Wall -Werror --pedantic -o bin/test_generator
test_generator succesfully created...
juan@juan-laptop:~/dev/uc/arquitectura/proyecto$ cd bin
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ls
cachesim test_generator
```

Imagen 7. Compilación, instalación y ubicación de `test_generator`.

Sus argumentos de entrada son similares a los de `cachesim`, posee un mensaje por defecto:

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./test_generator
test_generator - generates input files for cachesim.
Try [test_generator -h] or [test_generator --help] for help.
```

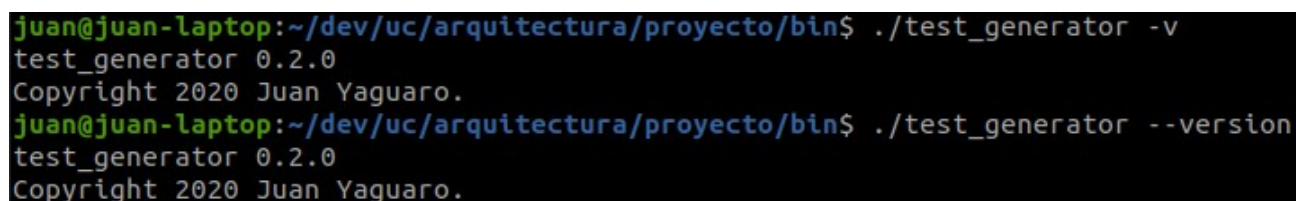
Imagen 8. `test_generator` siendo ejecutado sin argumentos de entrada.

Así como argumentos de ayuda `-h` y `--help`:

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./test_generator -h
Usage: test_generator -c=[FILENAME] -d=[FILENAME]
-c=[VALUE]          filename for config file.
-d=[VALUE]          filename for data file.
-h, --help          display all available commands.
-v, --version       display version of test_generator.
Full documentation at: <https://github.com/juanyaguaro/cachesim>.
Or available locally in <docs> folder.
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./test_generator --help
Usage: test_generator -c=[FILENAME] -d=[FILENAME]
-c=[VALUE]          filename for config file.
-d=[VALUE]          filename for data file.
-h, --help          display all available commands.
-v, --version       display version of test_generator.
Full documentation at: <https://github.com/juanyaguaro/cachesim>.
Or available locally in <docs> folder.
```

Imagen 9. `test_generator` siendo ejecutado con el argumento de entrada `-h` o `--help`.

Como también argumentos de version `-v` y `--version`:

A terminal window with a black background and green text. The prompt is 'juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin\$'. The first command is './test\_generator -v', which outputs 'test\_generator 0.2.0' and 'Copyright 2020 Juan Yaguaró.'. The second command is './test\_generator --version', which also outputs 'test\_generator 0.2.0' and 'Copyright 2020 Juan Yaguaró.'.

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./test_generator -v
test_generator 0.2.0
Copyright 2020 Juan Yaguaró.
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./test_generator --version
test_generator 0.2.0
Copyright 2020 Juan Yaguaró.
```

*Imagen 10. test\_generator siendo ejecutado con el argumento de entrada -v o --version.*

Para generar los casos de prueba (archivos de entrada válidos dentro de cachesim), test\_generator requiere únicamente de los argumentos `-c` y `-d`. A continuación se define cada uno de los argumentos de entrada:

- i. `-c=[NOMBRE_DE_ARCHIVO]`: indica el nombre del archivo de configuración del simulador que se desea generar. Es un argumento requerido.
- ii. `-d=[NOMBRE_DE_ARCHIVO]`: indica el nombre del archivo de datos a introducir en el simulador que se desea generar. Es un argumento requerido.

Por ejemplo, para generar los casos de prueba prueba\_config.in y prueba\_data.in con test\_generator, se deberá escribir el siguiente comando:

```
./test_generator -c=prueba_config.in -d=prueba_data.in
```

Nuevamente, el orden en el que se añaden los argumentos de entrada es irrelevante (manteniendo la misma regla de no mezclar los argumentos de ayuda y versión con los de funcionalidad).

La salida del programa serán números generados en los archivos de configuracion tomando en cuenta que los tamaños de cache y tamaños de linea siempre serán potencias de 2 (siendo tamaño de línea menor o igual que el tamaño total), así como el método de reemplazo y el tipo de caché siendo los números 0 o 1 únicamente (escogiéndolos al azar).

Para el caso del archivo generado de datos, se encuentran  $n$  números aleatorios (máximo 100) siempre positivos y en el rango de 0 y  $2^{16}$ . La distribución de estos números favorece la reaparición de los mismos, como si fuera una muestra con reemplazo en términos estadísticos (esto aumenta sustancialmente la probabilidad de un alto número de aciertos de caché, tal como sucedería en la realidad).

Este programa se ha realizado con la finalidad de ayudar al usuario a probar distintas opciones aleatorias que producirán resultados variados en el simulador cachesim, ofreciendo una alternativa valiosa con la automatización de un proceso tedioso como lo es el diseño de casos de pruebas.

## Análisis de resultados

Se procedió a generar un caso de prueba con test\_generator, el cual se llamará c1.in para el archivo de configuración, y d1.in para el archivo de datos. La salida del programa se muestra a continuación:

```
Copyright 2020 Juan Aguado  
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./test_generator -d=d1.in -c=c1.in  
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$
```

Imagen 11. Generación de archivos de prueba con test\_generator.

Los datos generados en ambos archivos fueron los siguientes:

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ cat c1.in  
4  
0  
1  
1  
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ cat d1.in  
53514  
60101  
53514  
60101  
53514  
37553  
33527  
48297  
37553  
63174  
13754  
37553  
60101  
36999  
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$
```

Imagen 12. Visualización de datos dentro de archivos generados con test\_generator.

Se puede notar la repetición de ciertos valores a lo largo del archivo, el cual en este caso resulta ser corto en cuanto a cantidad de elementos.

Luego se procede a utilizar el simulador cachesim con los siguientes argumentos para guardar el resultado dentro de un archivo result1.out:

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim -d=d1.in -c=c1.in -o=result1.out
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$
```

Imagen 13. Ejecución de comando para activar la simulación de memoria caché en cachesim.

La ejecución de dicho comando genera el siguiente resultado en el archivo de salida result1.out:

```
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ cat result1.out
-----
Address to allocate    Hit(1)/Miss(0)    Set ID    Old cache line content    New cache line content
-----
          53514             0           2             -1             -1
          60101             0           1             -1             -1
          53514             1           2             53514           53514
          60101             1           1             60101           60101
          53514             1           2             53514           53514
          37553             0           1             60101           60101
          33527             0           3             -1             -1
          48297             0           1             37553           37553
          37553             0           1             48297           48297
          63174             0           2             53514           53514
          13754             0           2             63174           63174
          37553             1           1             37553           37553
          60101             0           1             37553           37553
          36999             0           3             33527           33527
Total cache allocations:      14
Total cache hits:             4
Total cache misses:          10
Cache hit frequency:         28.5714%
Cache miss frequency:        71.4286%
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$
```

Imagen 14. Datos presentes en archivo generado por cachesim.

En dicho archivo se pueden visualizar los distintos datos como lo sería cada uno de los números a introducir en el simulador y la reacción del simulador a dicho intento de guardar el número.

Al final del archivo se encuentran las estadísticas referentes a la cantidad de aciertos, fallos y la frecuencia de los mismos.

Con esto se puede visualizar que se realiza la simulación de memoria caché desde cachesim de manera precisa y efectiva, haciendo uso de los métodos de salida para actualizar al usuario del estado del simulador en cada momento.

Para realizar una verificación referente a seguridad y pruebas contra errores, se decide utilizar el programa valgrind para ejecutar cachesim sobre él, con los mismos datos y visualizar si existen distintos riesgos de fallo de memoria. A continuación se muestra la salida que otorga valgrind:



```

juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ valgrind ./cachesim -d=d1.in -c=c1.in -o=result1.out
==34025== Memcheck, a memory error detector
==34025== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==34025== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==34025== Command: ./cachesim -d=d1.in -c=c1.in -o=result1.out
==34025==
==34025== HEAP SUMMARY:
==34025==      in use at exit: 0 bytes in 0 blocks
==34025==    total heap usage: 10 allocs, 10 frees, 98,928 bytes allocated
==34025==
==34025== All heap blocks were freed -- no leaks are possible
==34025==
==34025== For lists of detected and suppressed errors, rerun with: -s
==34025== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$

```

Imagen 15. Análisis de riesgos de cachesim desde valgrind.

Como puede visualizarse, valgrind garantiza que el programa se encuentra a prueba de errores y pérdidas de memoria, liberando todos aquellos bloques de memoria que requería para ejecutarse, sin dar espacio a pérdidas de la misma.

Esto señala que cachesim se figura como un programa robusto, capaz de identificar errores y señalarlos al usuario, como podrían ser errores de archivos no encontrados, argumentos de entrada inválidos, tamaños de cache inválidos, archivo de configuración incompleta, o incluso tipo de caché inválido:

```

juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim -d=archivo_inexistente.in -c=c1.in -x
Error: Failed to open archivo_inexistente.in
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim -d=d1.in -c=c1.in -x -opcion_invalida
Error: Invalid argument received.
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ cat tamano_invalido.in
3
1
9
0
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim -x -c=tamano_invalido.in -d=d1.in -o=result2.out
Error: Invalid cache size.
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ cat configuracion_incompleta.in
4
8
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim -c=configuracion_incompleta.in -d=d1.in -o=result3.out
Error: Invalid input read in config file.
Error: Invalid cache size.
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ cat tipo_invalido.in
1
8
3
7
juan@juan-laptop:~/dev/uc/arquitectura/proyecto/bin$ ./cachesim -c=tipo_invalido.in -o=result4.out -d=d1.in
Error: Invalid cache type read in config file.

```

Imagen 16. Distintos mensajes de cachesim a errores en datos de configuración recibidos.

Con esto se garantiza que el programa no aborte su ejecución inesperadamente, dando así un comportamiento determinístico y definido con el tipo de información recibida.



# Conclusión

El simulador de memoria caché cachesim, en conjunto con su herramienta de generación de pruebas test\_generator han resultado exitosos en la realización de sus tareas solicitadas, las cuales se reducen a simular el comportamiento de la memoria cache bajo distintas configuraciones de tamaño, tipo y funcionamiento.

Se recomienda con mucho agrado a los usuarios de cachesim, a utilizar el generador de pruebas test\_generator para probar distintas configuraciones aleatorias, promoviendo así el análisis de resultados de cachesim sobre múltiples casos de pruebas.

La definición de la API de cachesim se ha hecho con la finalidad de promover la fácil identificación de fallas y errores, modularizando casi en su totalidad las distintas funcionalidades que provee cachesim. Haciendo uso de buenas practicas de los nuevos estándares de C++ como ISO C++17, se implementa cachesim y test\_generator bajo la guía de estilo de código C++ de Google.

Para finalizar, la experiencia y el aprendizaje que ha brindado este proyecto se considera muy importante para comprender el funcionamiento a nivel abstracto de las tareas que realiza la memoria caché.

# Bibliografía

- C++ Standard Committee. (s. f.). *ISO C++ Standard*. <https://isocpp.org>.  
<https://isocpp.org/std/the-standard>
- *Cache Memory - an overview | ScienceDirect Topics*. (s. f.).  
<https://www.sciencedirect.com>.  
<https://www.sciencedirect.com/topics/computer-science/cache-memory>
- Google. (s. f.). *Google C++ Style Guide*. <https://google.github.io>.  
Recuperado 11 de enero de 2021, de  
<https://google.github.io/styleguide/cppguide.html>
- Hennessy, J. L., & Patterson, D. A. (2011). *Computer Architecture: A Quantitative Approach* (5.<sup>a</sup> ed.). Morgan Kaufmann Publishers.
- Microsoft. (2016, 4 noviembre). *Run-Time Type Information*.  
<https://microsoft.com>. <https://docs.microsoft.com/en-us/cpp/cpp/run-time-type-information?view=msvc-160>
- Patterson, D. A., & Hennessy, J. L. (2012). *Computer Organization and Design* (4.<sup>a</sup> ed.). Elsevier Gezondheidszorg.
- Patterson, P. A. J. D. L. (2021). *Estructura y Diseño de Computadores* (2.<sup>a</sup> ed.). Reverte.
- *RAII - cppreference.com*. (s. f.). <https://en.cppreference.com>. Recuperado 5 de enero de 2021, de <https://en.cppreference.com/w/cpp/language/raii>
- Yaguarro, J. (s. f.). *juanyaguarro/cachesim*. <https://github.com/juanyaguarro/>.  
Recuperado 10 de enero de 2021, de  
<https://github.com/juanyaguarro/cachesim>