Final Year Project

Foreshadow

Cache Simulator

FORESHADOW

Muhammad Faisal i14-0154

Hammad Nadeem i15-0162

Hamza Talat i15-0328

Supervisor

Dr Mirza Omer Beg

Department of Computer Science

FAST NUCES

2018

Anti-Plagiarism Declaration

This is to declare that the above publication produced under the:

Title: Foreshadow Cache Simulator

is the sole contribution of the authors and no part hereof has been reproduced on as it is basis (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. The work presented in the report is our own and it has not been submitted/presented previously to any other institution or organization. We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: _____

Student 1

Name: Hamza Talat

Signature: _____

Student 2

Name: Hammad Nadeem

Signature:_____

Student 3

Name: Muhammad Faisal

Signature: _____

Supervisor

Name: Dr Mirza Omer Beg

Signature: _____

1. Introduction

Foreshadow is a cache simulator that is designed to analyze the performance difference between prefetching algorithms and cache configurations. In this project we will be implementing three software based prefetchers. We will also be implementing a web based cache simulator that will show the evaluated results in graphical representation. This simulator will give a functionality to the users to add their own prefetching algorithms and compare their performances with already available prefetcher .The output is based on the performance factors like hit rate, latency, miss rate and trashing.

1.1. Motivation

Modern day processors are much more faster than secondary memory and in order to cope with these fast processors, caches were introduced in the hardware. The methods used in silicon for prefetching is simple due to design constraints. A standard stream buffer prefetcher is implemented on almost all hardware so we want to show the potential improvement in performance of the cache by implementing machine learning prefetchers. Advanced Micro Devices (AMD) has implemented a branch predictor in their latest Zen micro architecture and they claim significant improvement.

2. Project Vision

This project is intended to develop three prefetching algorithms, six eviction policies, a cache simulator and a web based graphical user interface. This project will log memory access sequence from a live program using Intel pin tool. These traces will be used to train a neural network and a preceptron learner prefetcher. These models will be compare with a stream buffer prefetcher in a simulation while monitoring several factors like hit rate, miss rate, trashing, latency etc. The data gathered from the simulation will be shown to the user in the form of graphical representation, charts and graphs. It will also allow the users to use their own prefetcher implementation. Cache size, associativity, mapping and eviction policy can be configured by the user so that

software developers can examine how their code behaves with different types of cache.

2.1 Problem Statement

Over the course of time, the difference in speed of central processing unit and random access memory is increasing. This phenomena is called the memory wall. With a monopoly on the world dram manufacturing, the situation is not going to improve any time soon. The cache can only speed up a process if it is utilized effectively. The use of modern prefetching algorithms can increase the performance of the cache and save a large number of cycles accessing memory. This would save electric power and reduce time of execution. The visualization of caches will also help students understand architecture. And the ability to use custom implementations will save researchers time evaluating new techniques.

2.2 Business opportunity

This is an open source project which is available on Github and was created for computer scientists. Link to Github project is: https://github.com/hammadnadeemx/Foreshadow

2.3 Objectives

- A web based cache simulator
- Reading memory access sequences
- Visualization of results
- Three prefetching algorithms
- Six eviction policies

2.4 Scope of the Project

Our project mainly consists of two parts. First part consists of different prefetching algorithms with several eviction policies and the second part consist of a web based cache simulator with result visualization. Mostly students studying Computer architecture find it difficult to understand cache concepts and this project will help them understand easily. Researchers can use this to evaluate new prefetching techniques.

2.5 Constraints of project

This software is just a simulation so we are not going into any architecture specific details. Any memory intensive benchmark is executed on the host system and all the memory access are recorded using Intel pin tool. This is repeated several times to gather data for training machine learning

models. These models will be used as prefetchers in the simulator along with a stream buffer prefetcher. When the simulator runs, it will first gather data from a new execution of the same benchmark and pass it to each of the prefetchers. These prefetchers will then utilize their virtual cache and hit rate, thrashing and miss rate will be monitored. To add the effect of latency a short time delay will be added to produce more realistic results. After all memory sequence has been used, the software will display output in a graphical form.

2.6 Stakeholders Description

A stakeholder in the design of the system is an individual, team, organization, or classes thereof, having an interest in the realization of the system/project. Usually they are

- Students
- Researchers
- Developers

2.6.1. Stakeholders Summary

The stakeholders according to their roles and concerns are :

Students	See the working of cache prefetchers in graphical form.
Researchers	Researchers will evaluate the performance of different prefetching techniques on different cache configurations.
Developers	Examine the behavior of their application with various cache configurations.

2.6.2. Key High Level Goals and Problems of Stakeholders

The main problem is with the researchers that don’t want to create a whole new simulator from scratch in order to test their prefetcher so they will use this simulator to check the results on different cache configurations. Students can use this software to

better understand computer architecture. Software developers can use it to examine cache behavior of their code.

3. Software Requirement Specifications

3.1. List of Features

The features of Foreshadow are :

3.1.1. Prefetcher

- Three complete prefetchers.
- Custom prefetcher implementation.

3.1.2. Simulator

- A web based cache simulator.
- Six eviction policies.
- Visualization of results.

3.2. Functional Requirements

The functional requirements of foreshadow are :

- The ability to log memory access sequences of any executable.
- The cache configuration's and number of caches under simulation is specified by user input.
- Simulate the specified caches and visually display their performance difference.

3.3. Quality Attributes

Some the quality attributes are :

3.3.1. Functional Stability

Forshadow should be able to run multiple simulations on end without any crashes.

3.3.2. Performance Efficiency

This project is efficient in performance as it returns results in a minimal amount of time.

3.3.3. Usability

Usability and re-usability are the key aspects as the simulator allows the user to write his own prefetcher and is user friendly.

3.3.4. Maintainability

As all the structure is defined in classes so it is easy to maintain. Open source software has a reputation for being easy to maintain.

3.4. Non-Functional Requirement

The software will be stable enough to run for several hours at a time and if it crashes it will be able to recover any collected data until that point.

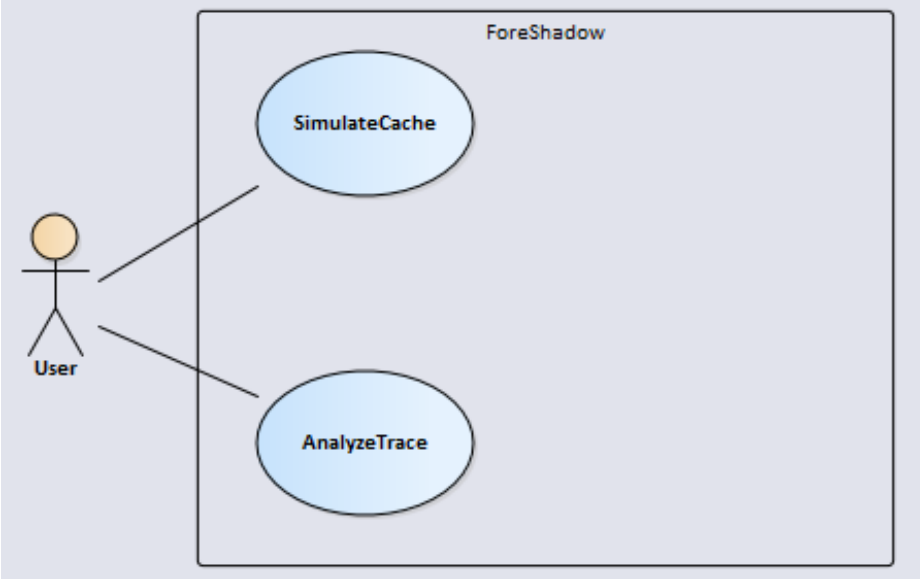
4. High level Use cases

Some of the use cases are as follows:

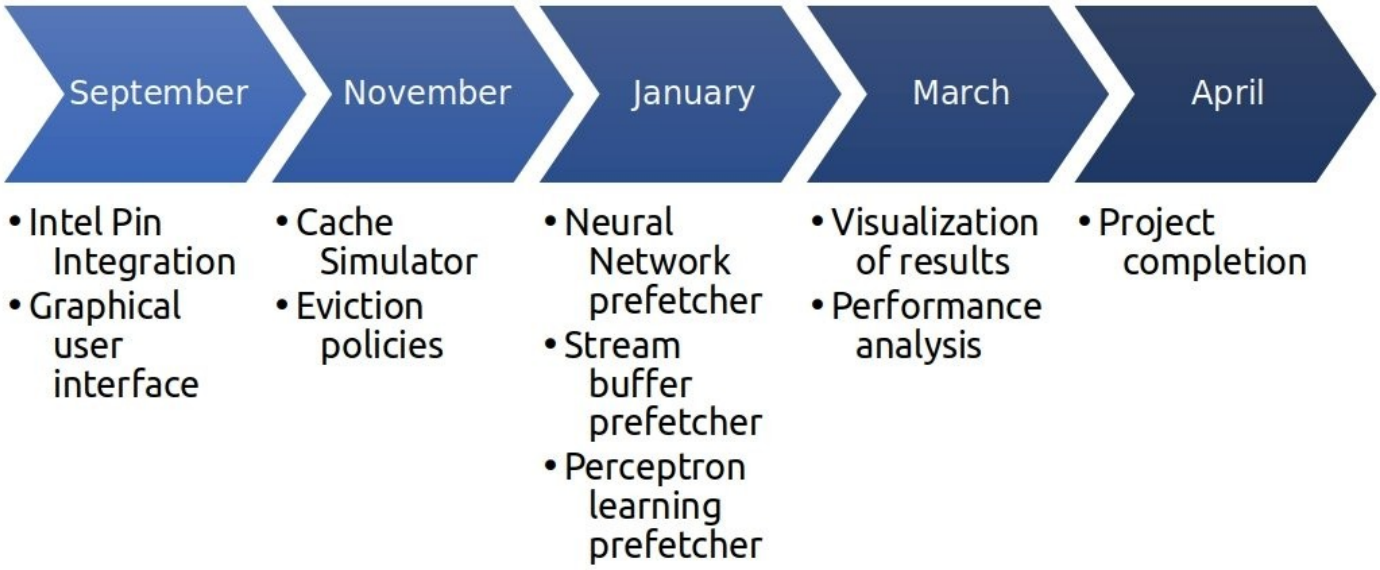
Use Case:	SimulateCache
Actors:	User
Type:	Primary
Description:	The user selects 2 or more caches and their configuration which includes, number of sets, lines per set, eviction policy and prefetching method. The simulation then runs the benchmark and compares their performance. Finally the results are shown graphically.

Use Case:	AnalyzeTraces
Actors:	User
Type:	Primary
Description:	The user selects a benchmark to gather memory access patterns and specifies the iterations. Foreshadow will run this multiple times and save the memory patterns. These patterns will be used for model training.

4.1 Use case diagram



5. Iteration Plan



6. Iteration 2

6.1. Expanded Use Cases

Use Case Name	SimulateCache.
Scope	Allows the simulation to begin.
Level	User Goal.
Primary Actor	Researchers, students and developers.
Stakeholders and Interests	▪ User.
Preconditions	▪ The user must have any memory intensive software to run as a benchmark. ▪ The user must specify the number, type of cache, size and eviction policies. ▪ The user needs to have his own implementation of prefetcher or chose any of the default prefetchers.
Post conditions	▪ The user is shown the results of the run which include graphs, charts and table. ▪ The user has the option of viewing the contents of each cache one step at a time as it is being utilized. ▪ Scores are shown to help compare each configuration.
Main Success Scenario	Success Scenario with all possible actor and system actions:
Actor Actions	**System Response**

1. User will select New Run.	2. System will navigate to the page that ask the user to enter the configuration of each cache.
3. User enters all required details.	4. System will prompt user to verify his information.
5. User will verify his information and select start.	6. System will run the benchmark and display the results of the comparison.
Extensions	6a. User entered a invalid program path. 1. System will generate error message of file not found. 1. The user then enters information for another program and system will check again.
Special Requirements	▪ The user must have a compatible operating system. ▪ The users prefetcher must have correct functionality otherwise it may not work with Foreshadow. ▪ The user must have a compatible web browser.

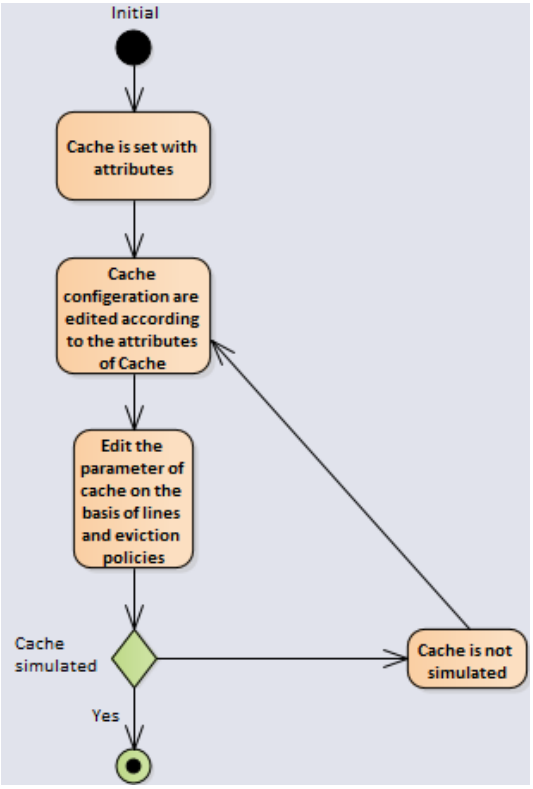
Technology and Data Variation List	1. D3.js is used for the front end.
Frequency of Occurrence	This operation can occur frequently.

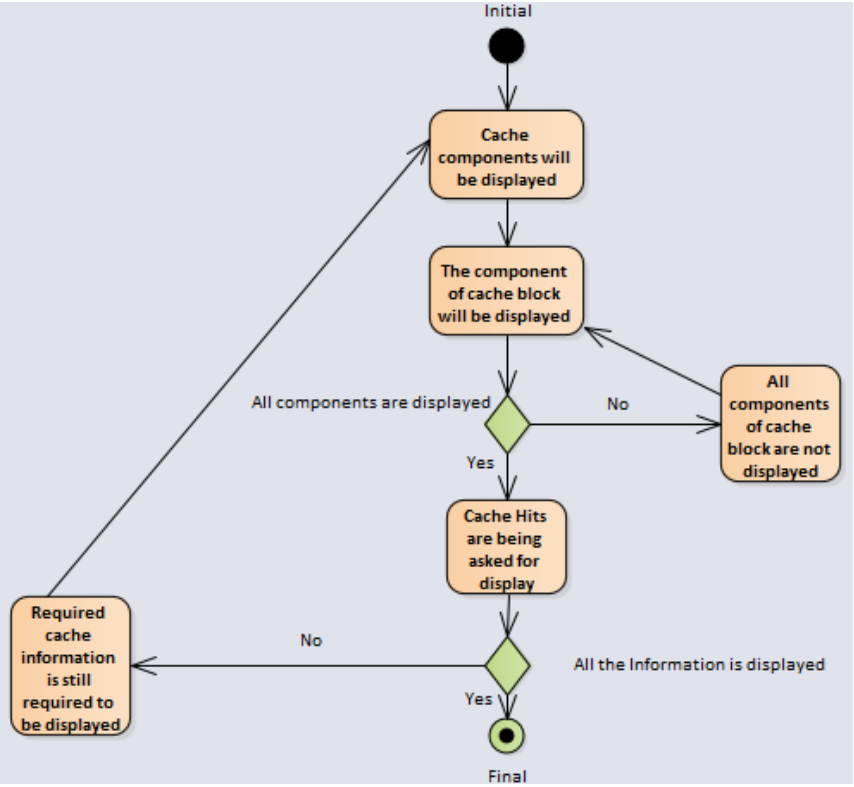
Use Case Name	AnalyzeTraces.
Scope	Allow the users to gather data for training models.
Level	User Goal.
Primary Actor	Researchers, students and developers
Stakeholders and Interests	▪ User wants to collect memory traces so he can train either the neural network or the perceptron learner prefetcher.
Preconditions	▪ User must have a valid program that runs on its own like a benchmark. ▪ The user must specify the number of iterations to run the software.
Post conditions	▪ The user has enough data for training.
Main Success Scenario	Success Scenario with all possible actor and system actions:
Actor Actions	**System Response**
1. User will select Collect Traces.	2. System will navigate to the page that ask user to enter the path to the executable file and number

	of runs.
3. User will enter these details.	4. System will prompt user to verify his information.
5. User will verify his information and select Start.	6. System will run the program and collect addresses. It will display a progress bar during this run. 7. After successfully completing the runs, the addresses are stored in a file.
Extensions	6a. The program crashes. 2. System will generate error message of incomplete run. 2. The user has the option to try again or select another program.
Special Requirements	▪ The user must have a compatible operating system. ▪ The user must have a stable memory intensive benchmark. ▪ The user must have a compatible web browser.
Technology and Data Variation List	1. Foreshadow will use Intel PIN tool to gather memory accesses. 2. D3.js is used for the front end.

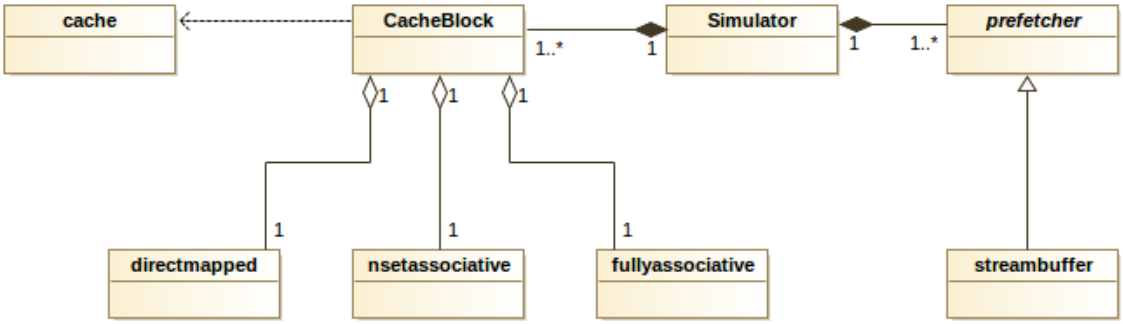
Frequency of Occurrence	This operation can occur frequently but only for model training purposes.

6.2. Activity Diagram



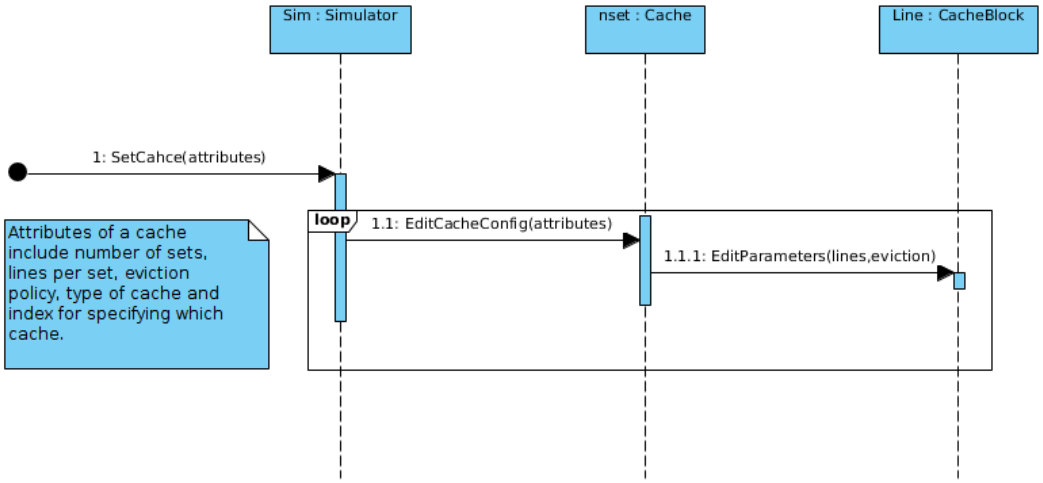


6.3. Domain Model

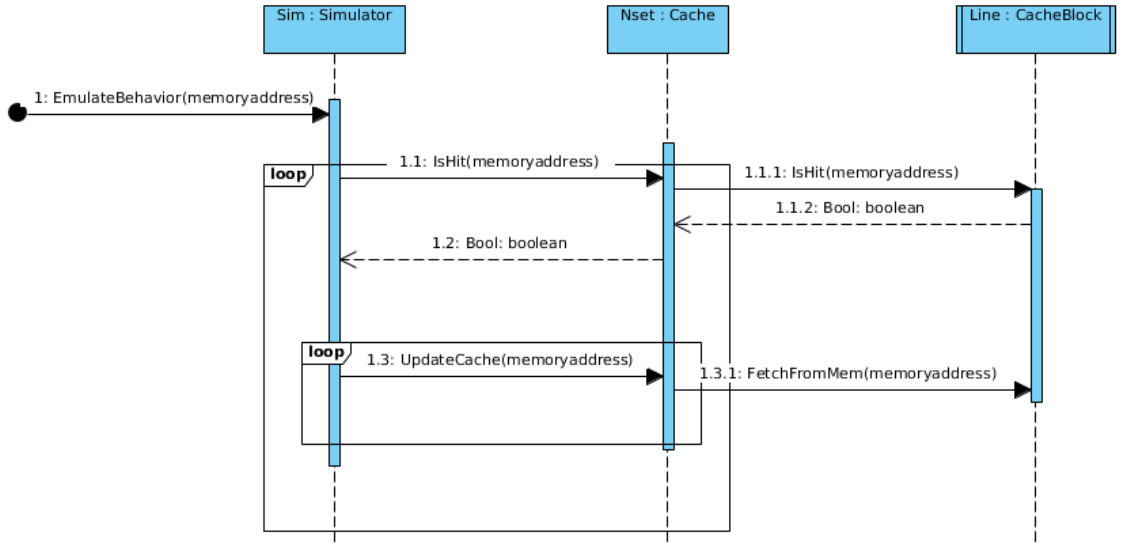


6.4 System Sequence diagram

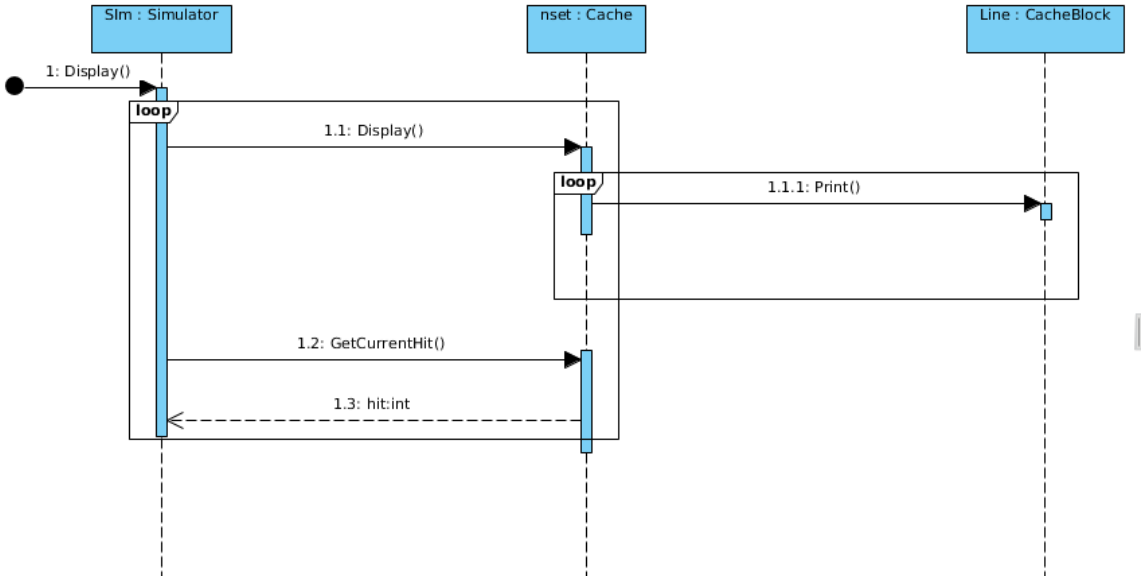
- Sub usecase to set configuration of simulators caches:



- Sub usecase to show internal state of all caches under simulation:



SimulateCache use case:

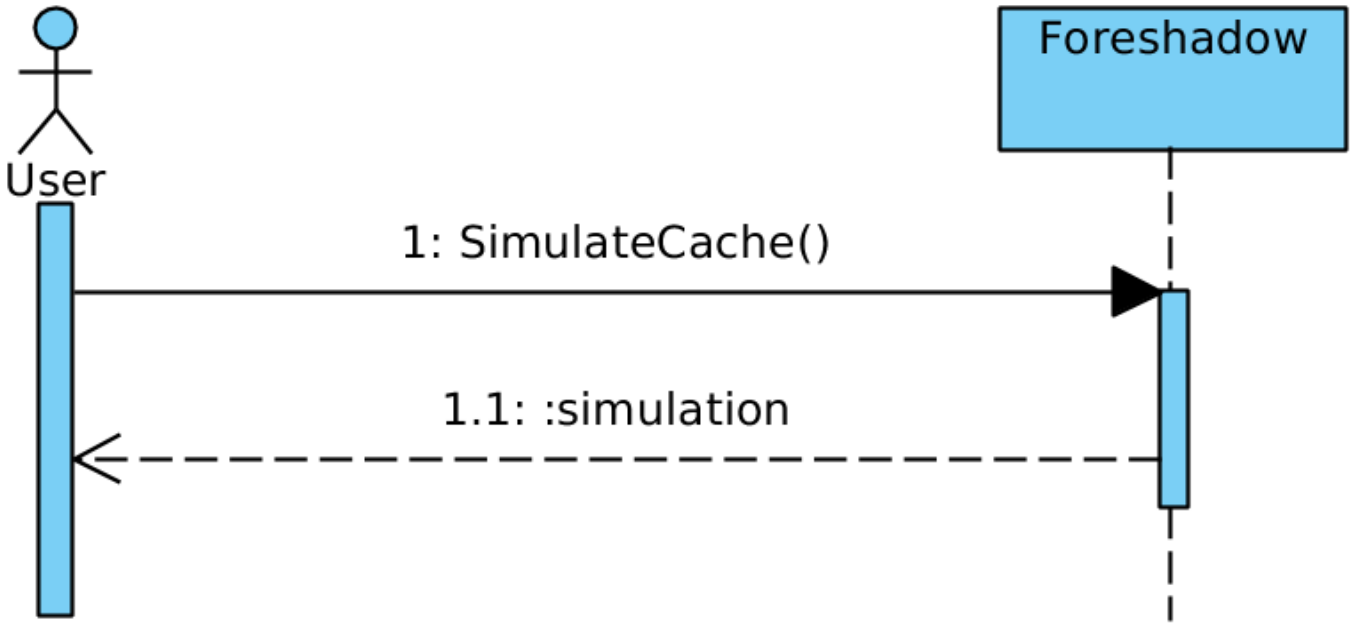


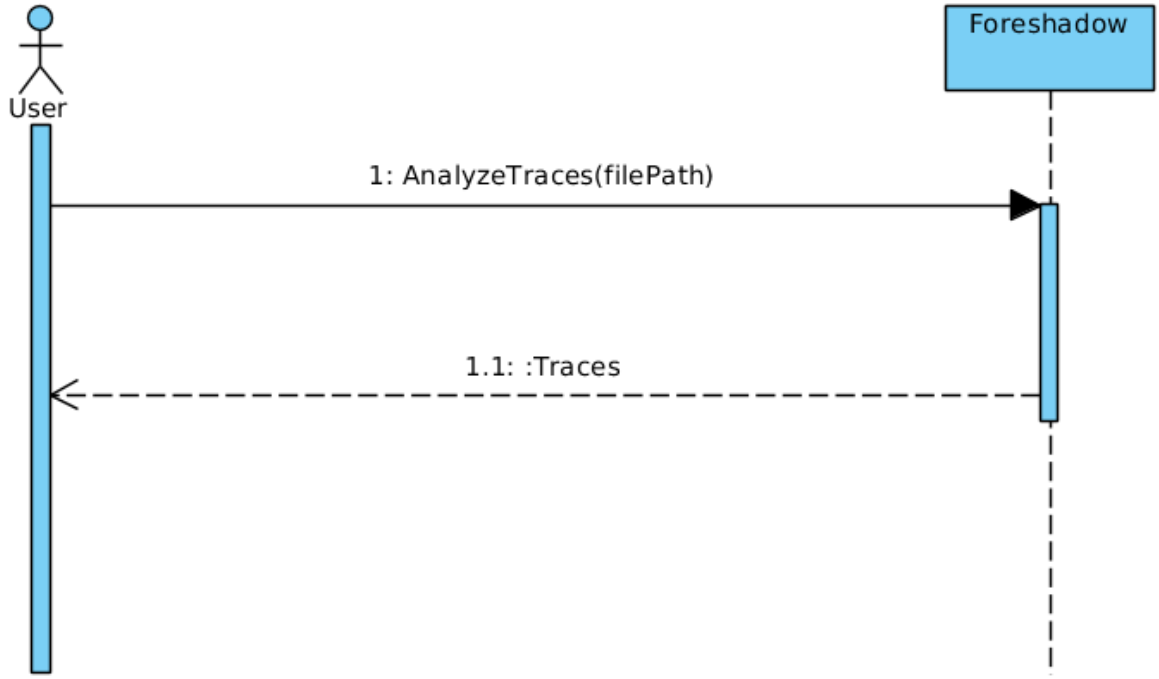
6.5. Operation Contracts

Name	Build_Cache()
Responsibility	Creates virtual caches according to the specifications entered by the user.
Type	System
Cross Reference	Use Case: Simulate Cache
Pre-conditions	A user wants to run a comparison.
Post-conditions	A cache instance c was created. C was associated with simulator.

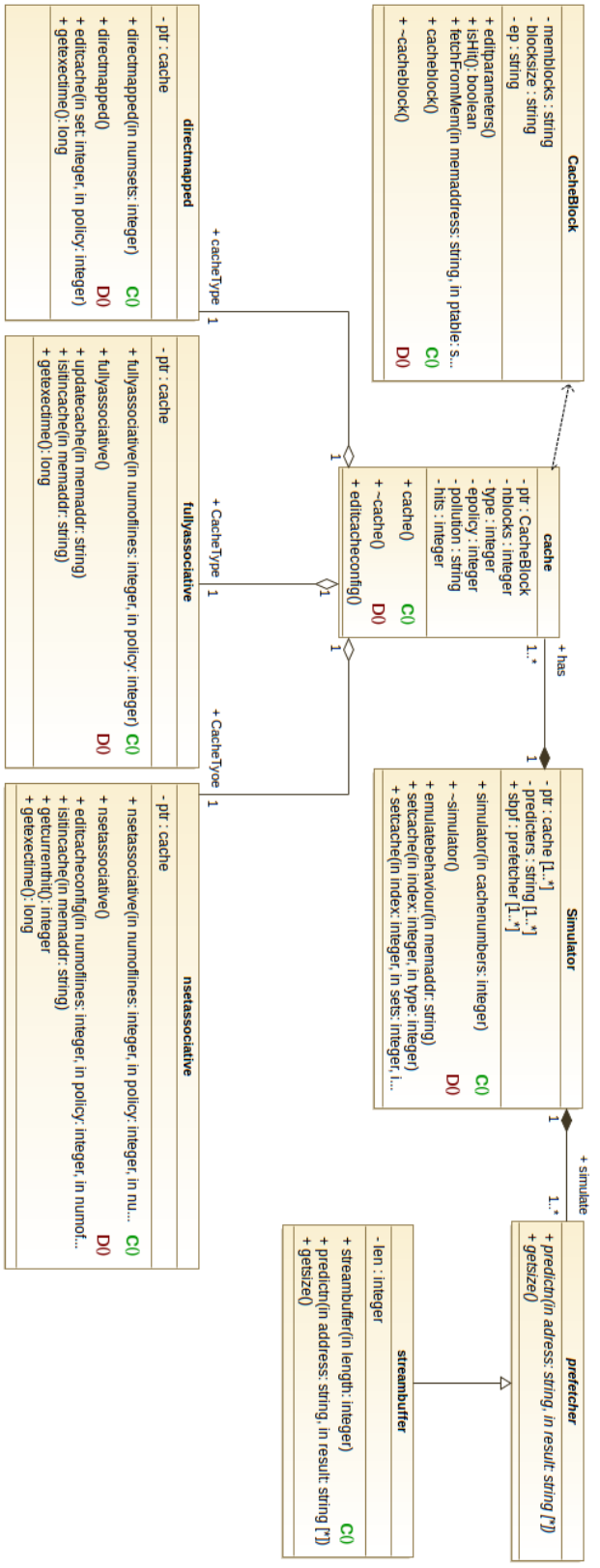
Name	SelectProgram()
Responsibility	Runs a program several times under intel pin to gather memory access sequences for training.
Type	System
Cross Reference	Use Case : Analyze Traces
Pre-conditions	A valid program is present.
Post-conditions	A memory sequence file was created.

6.6. Sequence Diagrams





6.7. Class Diagram



7. Implementation Details

Implementation completed:

- LIFO
- LRU
- LFU
- MRU
- RR
- FIFO
- GUI
- Simulator
- Stream buffer prefetcher