Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА

## Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Програмний симулятор кеш-пам'яті комп'ютера

(тема)

Виконав:
здобувач 3 року навчання,
групи КІУКІу-22-1
Максим АФАНКОВ
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: ст. викл. Дмитро РОСІНСЬКИЙ
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

# ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Афанкову Максиму Валерійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмний симулятор кеш-пам'яті комп'ютера

затверджена наказом по університету від “ 26 ” травня 2025 р. № 425 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 2025 р.

3. Вхідні дані до роботи 1. Цільова платформа – Windows, Linux.
2. Тип кеш-пам’яті, що моделюється, – довільний.
3. Можливість розширених налаштувань симулятора.

4. Перелік питань, що потрібно опрацювати у роботі

1. Аналіз проблеми
2. Реалізація симулятора ECS
3. Методичні матеріали та демонстраційні приклади
4. Модуль симулятора протоколу MESI
5. Експериментальне дослідження
6. Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 15 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Строк / терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз проблеми та огляд існуючих рішень | 10.06.25-13.06.25 | |
| 2 | Реалізація симулятора ECS | 14.06.25-21.06.25 | |
| 3 | Розробка методичних матеріалів та | 18.06.25-21.06.25 | |
| | демонстраційних прикладів | | |
| 4 | Розробка модуля протоколу MESI | 23.06.25-28.06.25 | |
| 5 | Експериментальне дослідження | 30.06.25-02.07.25 | |
| 6 | Оформлення матеріалів кваліфікаційної роботи | 03.07.25-05.07.25 | |
| 7 | Подання кваліфікаційної роботи керівникові та | 07.07.25-09.07.25 | |
| | її попередній захист | | |
| 8 | Подання кваліфікаційної роботи на рецензу- | 10.07.25-11.07.25 | |
| | вання | | |
| | | | |

Дата видачі завдання " 09 " червня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Дмитро РОСІНСЬКИЙ
(підпис) (посада, власне ім'я, прізвище)

# РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 66 с., 13 рис., 2 табл., 1 дод., 13 джерел.

БАГАТОЯДЕРНИЙ ПРОЦЕСОР, КЕШ-ПАМ'ЯТЬ, КЕШ-ПРОМАХ, ПОЛІТИКА ЗАМІЩЕННЯ РЯДКА, ПОПАДАННЯ У КЕШ, ПРОТОКОЛ КОГЕРЕНТНОСТІ, РІВНІ КЕШУ, СИМУЛЯТОР.

Метою кваліфікаційної роботи є розробка програмного симулятора, який візуалізує функціонування кеш-пам'яті комп'ютера з метою навчання.

Симулятор орієнтований на вивчення таких фундаментальних аспектів кешування, як ємність кеш-пам’яті, розмір рядка, ступінь асоціативності, політика заміщення та інші параметри. Особливу увагу в симуляторі приділено демонстрації роботи протоколу когерентності кеш-пам’яті MESI. Завдяки поетапному візуальному представленню кожного з можливих станів і переходів у протоколі, користувачі мають змогу глибоко зрозуміти логіку когерентного управління даними у багатоядерних системах.

Результати експериментального впровадження симулятора у навчальний процес підтвердили його доцільність та високу ефективність у реальному аудиторному середовищі.

# ABSTRACT

Bachelor’s thesis: 66 pages, 13 figures, 2 tables, 1 appendix, 17 sources.

CACHE HIT, CACHE LEVELS, CACHE LINE REPLACEMENT POLICY, CACHE MEMORY, CACHE MISS, COHERENCE PROTOCOL, MULTICORE PROCESSOR, SIMULATOR.

The aim of the qualification work is to develop a software simulator that visualizes the functioning of computer cache memory for educational purposes.

The simulator is designed to facilitate the study of fundamental aspects of caching, such as cache capacity, line size, associativity, replacement policy, and other parameters. Particular attention in the simulator is devoted to demonstrating the operation of the MESI cache coherence protocol. Through step-by-step visual representation of each possible state and transition within the protocol, users are able to gain a deep understanding of the logic behind coherent data management in multicore systems.

The results of experimental implementation of the simulator in the educational process have confirmed its feasibility and high effectiveness in real classroom environments.

ЗМІСТ

# СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЦП – центральний процесор

ACM – Асоціація обчислювальної техніки (англ., Association for Computing Machinery)

ATF – файл трасування адреси (англ., Address Trace File)

CCF – файл конфігурації мікросхеми (англ., Chip Configuration File)

ECS – симулятор кешу для навчання (англ., Educational Cache Simulator)

FIFO – черга з пріоритетом надходження (англ., First-In, First-Out)

LRU – найдавніше використовуваних (англ., Least Recently Used)

Lx – рівень кешу (англ., Level x)

MDI – багатовіконний інтерфейс (англ., Multiple Document Interface)

MESI – змінений, ексклюзивний, спільний, недійсний (англ., Modified, Exclusive, Shared, Invalid)

SCF – файл дослідження (англ., Study Case File)

UID – унікальний ідентифікатор (англ., Unique Identifier)

# ВСТУП

Курс «Архітектура комп’ютера» є фундаментальним у більшості освітніх програм з інформатики та інженерії, а також одним із найпопулярніших вибіркових курсів серед студентів суміжних інженерних спеціальностей. Висока зацікавленість студентів у цьому курсі є позитивним фактором, однак водночас створює низку викликів. Зокрема, викладачам доводиться докладати значних зусиль для доступного пояснення складних технічних понять студентам-початківцям або слухачам, які не мають достатньої попередньої підготовки.

Одним із ефективних підходів до покращення якості викладання та засвоєння матеріалу є використання візуальних тренажерів. Такий підхід має подвійну мету: по-перше, забезпечити наочне середовище для демонстрації базових понять комп’ютерної архітектури; по-друге, підвищити мотивацію студентів і сприяти глибшому розумінню навчального матеріалу.

Попри наявність широкого спектра візуальних тренажерів для вивчення архітектури комп’ютера, існує дефіцит інструментів, спеціально орієнтованих на моделювання кеш-пам’яті. Це ускладнює пояснення важливих тем, пов’язаних із продуктивністю системи та управлінням пам’яттю. У зв’язку з цим постає потреба у створенні нового візуального симулятора – ECS (Educational Cache Simulator).

У цій роботі продемонстровано можливості симулятора ECS як ефективного навчального інструмента для пояснення роботи багаторівневої кеш-пам’яті у сучасних багатоядерних та багаточипових процесорах. Окрему увагу приділено навчанню принципів когерентності кеш-пам’яті в системах з ієрархічною організацією пам’яті.

Симулятор ECS не лише поглиблює розуміння процесів, що відбуваються на апаратному рівні, а й дозволяє студентам оцінити вплив архітектурних рішень на продуктивність програм. Таким чином, його

використання у навчальному процесі сприяє формуванню системного бачення комп’ютерних технологій, підвищенню інтересу до дисциплін апаратного профілю та розвитку практичних навичок моделювання.

# 1 АНАЛІЗ ПРОБЛЕМИ

## 1.1 Загальні відомості

Курси з вивчення архітектури та організації комп’ютера визнаються однією з ключових складових у системі підготовки фахівців за спеціальністю «Комп’ютерна інженерія». Вони формують фундаментальні уявлення студентів про принципи функціонування обчислювальних систем. Однак постає проблема: мови програмування високого рівня абстрагують апаратну реалізацію, не даючи чіткого розуміння того, як саме комп’ютер виконує програму. Це часто призводить до зниження зацікавленості студентів і поверхневого засвоєння матеріалу, що ускладнює подальше вивчення апаратно-програмних концепцій.

Викладання дисципліни «Архітектура комп’ютера» є складним педагогічним завданням і потребує значних зусиль з боку викладача та активної участі студентів [1]. Зазвичай курс викладається на першому курсі, і для більшості слухачів є новим, що потребує адаптації до нової термінології та концепцій. Застосування візуальних тренажерів у навчальному процесі значно полегшує сприйняття матеріалу і, як засвідчено дослідженнями [2], підвищує інтерес до вивчення апаратної частини комп’ютерних систем. Для досягнення високої ефективності навчання викладач повинен ретельно добирати відповідні практичні завдання, лабораторні роботи й проєкти.

Сучасні обчислювальні системи побудовані на основі багаторівневої кеш-пам’яті [3], що дозволяє мінімізувати затримки між центральним процесором і основною пам’яттю, тим самим підвищуючи загальну продуктивність системи. Тому студентам важливо розуміти не лише загальні принципи архітектури комп’ютера, а й деталі організації систем із кількома процесорними ядрами (мультипроцесорів), зокрема – роботу ієрархії кеш-пам’яті.

Водночас у навчальній практиці складно знайти спеціалізований симулятор, який би давав змогу студентам наочно й ефективно вивчити параметри кешу (розмір, асоціативність, політики заміни тощо) та їх вплив на продуктивність програм. У цій роботі представлено розроблений навчальний симулятор ECS (Educational Cache Simulator), який дозволяє візуалізувати ключові аспекти функціонування кеш-пам’яті: доступ до кешу, заповнення рядків, промахи, потрапляння, а також асоціативність і взаємодію при послідовному та паралельному виконанні алгоритмів [4].

## 1.2 Особливості побудови курсів з вивчання архітектури та організації комп'ютера

Основною метою подібних навчальних курсів є формування у студентів чіткого розуміння принципів побудови комп’ютерних архітектур, особливостей функціонування компонентів обчислювальної системи та оцінки їхньої продуктивності. Курс також охоплює актуальні теми, пов’язані з архітектурою сучасних багатоядерних та багаточипових процесорів, а також розглядає питання структурної організації.

### 1.2.1 Організація курсу

У якості базового розглядається курс «Архітектура комп’ютера» [1], який складається з трьох основних компонентів: теоретичних лекцій (два заняття на тиждень), самостійної роботи студентів (два заняття на тиждень) та лабораторних занять (одне заняття раз на два тижні). Теоретичні заняття проводяться у великих аудиторіях із кількістю слухачів до 100 осіб, тоді як лабораторні роботи виконуються в комп’ютерних класах у малих групах до 20 студентів, кожен з яких працює індивідуально на персональному робочому місці. Умовою зарахування на курс є успішне завершення попередніх курсів із дискретної математики.

Теоретичний курс охоплює базові поняття комп’ютерної архітектури: абстракції та технології комп’ютерних систем, мову асемблера (x86, MIPS), комп’ютерну арифметику, структуру процесора, організацію пам’яті, типи накопичувачів, а також принципи роботи багаточипових і багатоядерних процесорів.

Підручники умовно поділяються на дві частини: перша частина включає теми комп’ютерної арифметики, кодування інформації та параметрів продуктивності; друга частина присвячена цифровим логічним схемам. Практичні лабораторні завдання розроблені відповідно до тематики теоретичних занять. Повний опис концептуальної структури курсу «Архітектура комп’ютера» подано в [1].

### 1.2.2 Проблеми та мотивація

Групою дослідників було проаналізовано результати успішності студентів [2], у результаті чого з’ясовано, що найбільші труднощі виникають під час опанування тем теоретичних лекцій, особливо у другому семестрі курсу. До таких тем належать: процесор, пам’ять, введення/виведення та розпаралелювання. Аналіз показав, що, хоча зазначені теми формально викладаються на лекціях, вони практично не представлені у вигляді теоретичних або практичних посібників. Натомість лабораторні заняття переважно присвячені розробці цифрових логічних схем.

Водночас у рекомендаціях IEEE Computer Society та Association for Computing Machinery (ACM) підкреслюється необхідність приділяти більше уваги архітектурі та організації багатоядерних процесорів, ніж традиційним темам, пов’язаним із проєктуванням апаратної логіки [5].

З метою уточнення думки студентів було проведено опитування, яке включало два запитання:

Питання 1. Чи доцільно вилучити практичні та лабораторні заняття з тем проєктування апаратної логіки та замінити їх на теми, пов’язані з

багатопроцесорною архітектурою та організацією?

Питання 2. Чи варто запровадити візуальний симулятор для лабораторних занять з багатопроцесорної архітектури та організації комп’ютерів?

Участь в опитуванні взяли 61 студент. Варіанти відповідей на обидва запитання були бінарними: «так» або «ні». Результати першого запитання наведено на рисунку 1.1: майже 64 % респондентів висловилися за зміну тематики лабораторних занять на користь багатопроцесорної архітектури та організації.

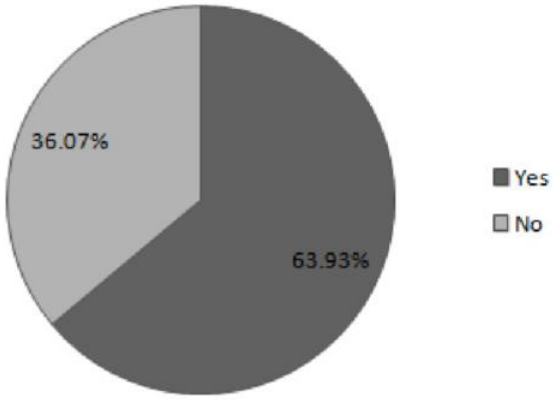


Рисунок 1.1 – Результати відповідей на питання 1

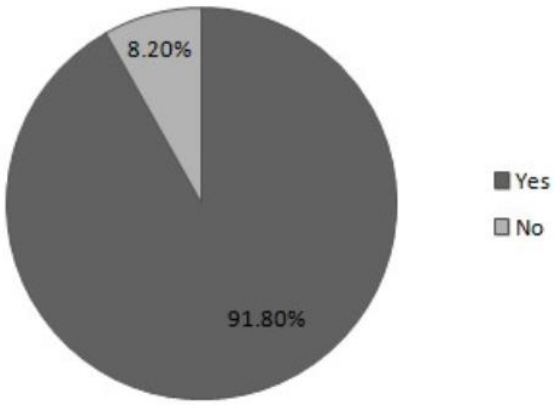


Рисунок 1.2 – Результати відповідей на питання 2

Результати другого запитання подано на рисунку 1.2. Очікувалося, що відповіді на обидва запитання будуть співвідносними, однак друге запитання викликало ще більшу підтримку: близько 92 % студентів виявили зацікавленість у використанні візуального симулятора для вивчення архітектури багатопроцесорних систем.

Результати опитування також засвідчили наявність значної частки студентів (близько третини), які висловили зацікавленість у докладному лабораторному посібнику з тематики логічного проєктування апаратного забезпечення. Це свідчить про потребу в додаткових навчально-методичних матеріалах для глибшого розуміння відповідних понять. Крім того, відповіді на друге запитання підтверджують доцільність використання візуальних тренажерів як засобу покращення ефективності навчального процесу.

Загалом результати опитування обґрунтовують потребу в розробці нового візуального симулятора, орієнтованого на вивчення архітектури та організації багатопроцесорних комп'ютерних систем. Такий симулятор є важливою складовою для реалізації змісту курсу «Архітектура комп'ютерів».

## 1.3 Супутні роботи

Існує чимало візуальних симуляторів, які допомагають студентам подолати певні фундаментальні частини архітектури та організації комп'ютера. Однак єдиного симулятора, який охоплює всі теми архітектури комп'ютера, не існує [6]. Більшість візуальних симуляторів не призначені для навчання студентів ні ієрархії та організації кеш-пам'яті, ні її внутрішнім параметрам, таким як розмір кешу, рядок кешу, асоціативність кешу, включення кешу тощо.

Visual EduMIPS64 – це навчальний посібник для конвеєрства інструкцій, виявлення та вирішення небезпек, обробки винятків, переривань та ієрархій пам'яті [7]. Це дуже потужний інструмент навчання, який імітує повну архітектуру конвеєра процесора MIPS64, але він не пропонує повного

огляду того, як працює ієрархія кеш-пам'яті або впливає на виконання. Крім того, тренажер вимагає, щоб студенти були вже знайомі з MIPS64 ISA, перш ніж вони зможуть використовувати симулятор, що неможливо для студентів першого курсу інформатики.

Dinero IV – це симулятор кешу, який моделює ієрархію пам'яті з різними кешами [8]. Це потужний і точний інструмент, але він може моделювати лише одноядерні системи. Однак це лише інструмент командного рядка, який не пропонує ні візуалізації, ні пояснення, коли студенти його використовують. Fraguela та ін. [9] визначають повністю параметризовані моделі, застосовні до n-шляхових асоціативних кешів, але лише для політики заміни LRU (найдавніше використовуваних). Створюваний симулятор ECS повинен моделювати дисципліну заміни рядка кешу FIFO (першим прийшов, першим вийшов) і LRU для всіх рівнів кешу.

CMP$im – це симулятор, заснований на інструменті Pin binary instrumentation [10]. Це кращий симулятор, який пропонує багатоядерну підтримку та збір даних для всіх рівнів кешу. Однак отримання результатів є більш складним, ніж пропонований симулятор ECS. Симулятор HC-Sim також заснований на Pin, який генерує сліди під час виконання та моделює декілька конфігурацій кешу за один запуск [11].

Herruzo та співавт. [12] розробили конфігурований симулятор кеш-пам’яті, що дозволяє студентам моделювати процес пошуку та запису даних у кеш-пам’ять. У даному рішенні підтримується налаштування таких параметрів, як розмір блоку, кількість блоків у наборі та загальна ємність кешу. Проте ці параметри обмежені кількома попередньо визначеними значеннями, а сам симулятор не підтримує багатоядерні архітектури та ієрархії кешів.

Місєв і Гусєв [13] створили візуальний симулятор, орієнтований на динамічне виконання команд за позачерговою схемою (ILP – instruction-level parallelism). Цей інструмент охоплює ключові компоненти позачергового виконання, зокрема перейменування регістрів, видачу інструкцій, зберігання

в буферах і механізми повторної відправки.

Модуль Cachegrind, що входить до складу Valgrind, є одним із найпоширеніших профайлерів для аналізу кеш-поведінки [14]. Незважаючи на свою популярність, Cachegrind вимагає детального аналізу коду на рівні окремих функцій і обмежується інформацією про кеші лише першого (L1) та останнього рівнів (L3). Це суттєво обмежує його придатність для дослідження поведінки кешу середнього рівня (L2), який активно використовується у сучасних багатоядерних процесорах. Крім того, Cachegrind не підтримує багатопотокове середовище зі спільною пам’яттю, що є суттєвою вадою для моделювання паралельних програм.

На відміну від вищезазначених інструментів, симулятор ECS спеціально розроблений для освітніх цілей. Його головною перевагою є підтримка повного циклу трирівневої кеш-ієрархії (L1, L2, L3) та можливість моделювання багатоядерних архітектур. ECS забезпечує покрокове виконання з можливістю зупинки та аналізу результатів кожного звернення до пам’яті. Це дозволяє студентам наочно простежити як попадання, так і промахи в кешах на кожному рівні ієрархії. Окрім того, ECS є кросплатформеним рішенням, що спрощує його впровадження в навчальний процес.

Загалом, попри наявність потужних візуальних або функціональних інструментів, жоден із розглянутих симуляторів не був створений із фокусом на навчання організації кеш-пам’яті. Пропонований ECS усуває цю прогалину, забезпечуючи баланс між наочністю, функціональністю й освітньою цінністю.

## 1.4 Симуляція кеш з протоколом когерентності MESI

Протокол MESI (Modified, Exclusive, Shared, Invalid) є одним із найпоширеніших методів забезпечення когерентності кеш-пам’яті в ієрархічних системах пам’яті [15]. Він визначає чотири основні стани, у яких

може перебувати блок кешу: змінений, виключний, спільний та недійсний. Кожен блок, до якого здійснюється доступ, повинен перебувати в одному з цих станів, а переходи між ними регулюються відповідно до специфікації протоколу MESI. У сучасних комп'ютерних системах, зокрема в архітектурах процесорів Intel та AMD, реалізовано протокол MESI або його розширені варіанти. Вивчення принципів його роботи є критично важливим для студентів, що опановують дисципліни з комп'ютерної архітектури.

Модуль MESI симулятора кеш-пам'яті ECS є ефективним освітнім інструментом, який дозволяє досягти низки важливих дидактичних цілей:

- демонструвати принципи функціонування когерентних протоколів типу write-back на прикладі MESI;
- забезпечити візуальне представлення станів блоків кеш-пам'яті та їх змін відповідно до взаємодії процесорів у багатопроцесорному середовищі;
- формувати розуміння причин та механізмів переходу між станами кешу внаслідок дій центрального процесора, переривань введення/виведення або операцій DMA;
- сприяти засвоєнню концепцій ієрархічної організації пам'яті, включаючи багаторівневу кеш-структуру (L1, L2, L3);
- поглиблювати знання про параметри кеш-пам'яті, зокрема: асоціативність, політику заміщення, розміри блоків кешу, політику запису тощо.

Розробка та впровадження модуля MESI у навчальний процес ґрунтується на освітніх принципах, окреслених у [1]. Симулятор забезпечує зручне візуальне середовище для практичного закріплення знань, дозволяючи студентам експериментувати з різними архітектурними конфігураціями. Такий підхід має не лише навчальну цінність у традиційному аудиторному форматі, а й є особливо ефективним у контексті дистанційного навчання та онлайн-курсів, де симулятор може використовуватись як самостійний інструмент для закріплення матеріалу.

Існує низка програмних засобів, функціональність яких частково

перетинається з можливостями модуля MESI симулятора кеш-пам’яті ECS. Найбільш показовими прикладами таких інструментів є:

- SMPCache – симулятор, який моделює системи кеш-пам’яті в середовищах симетричних мультипроцесорів. Його доцільно використовувати для вивчення кешованих багатопроцесорних систем у паралельних обчисленнях. Один із варіантів конфігурації підтримує застосування протоколу MESI;

- LIMES – інструмент, що реалізує протокол MESI для забезпечення когерентності кеш-пам’яті в середовищах паралельних багатопроцесорних систем;

- MINT – симулятор, який дозволяє досліджувати ієрархію пам’яті в системах зі спільною пам’яттю. Підтримує моделювання різних протоколів когерентності, зокрема MESI, MOESI та MOES.

Попри наявність базової підтримки протоколу MESI, жоден із перелічених інструментів не надає можливості детального спостереження за внутрішніми механізмами функціонування протоколу когерентності кеш-пам’яті. Вони, як правило, обмежуються імітацією результатів або наданням статистичних даних, без відображення конкретних переходів станів кеш-блоків.

Натомість модуль MESI симулятора ECS реалізує наочне моделювання процесів когерентності, зокрема відстеження змін станів (Modified, Exclusive, Shared, Invalid) для кожного кеш-блоку. Це дозволяє студентам не лише бачити результат, а й аналізувати хід змін та відповідні події в архітектурі багатопроцесорної системи. Такий підхід забезпечує глибше розуміння принципів роботи когерентних систем пам’яті, що робить ECS унікальним та надзвичайно корисним навчальним інструментом.

# 2 РЕАЛІЗАЦІЯ СИМУЛЯТОРА ECS

Симулятор ECS є незалежною від платформи програмною системою, реалізованою мовою Java. Основна логіка моделювання реалізується через набір спеціалізованих класів, кожен із яких відповідає за певні параметри кеш-пам’яті процесора. ECS орієнтований на навчальне використання та дає змогу студентам моделювати багаторівневу кеш-ієрархію у багатоядерному процесорі, аналізуючи її поведінку під час послідовного та паралельного виконання алгоритмів.

Основні функціональні можливості симулятора ECS наведено нижче.

1. Підтримка одноядерних і багатоядерних мультипроцесорних систем.

Симулятор дозволяє створювати однорідні мультипроцесори з довільною кількістю ядер, кожне з яких може працювати з певною швидкістю. Це забезпечує можливість дослідження різних архітектур багатопроцесорних систем.

2. Гнучка конфігурація зв’язку між ядрами та кешами.

Кожне ядро процесора може бути пов’язане з певними екземплярами кешів різних рівнів (L1, L2, L3). Наприклад:

- кеші рівнів L1 та L2 зазвичай виділяються окремо для кожного ядра;
- кеш L3 часто спільний для кількох або всіх ядер одного процесора.

Ця особливість дозволяє моделювати сучасні варіанти організації кеш-пам’яті на практиці.

3. Повна параметризація рівнів кешу.

Кожен рівень кеш-пам’яті може бути налаштований за такими параметрами:

- ємність кешу;
- розмір рядка;
- ступінь асоціативності;
- вибір політики заміщення (наприклад, LRU або FIFO);

- інклюзивність між кешами різного рівня.

Таким чином, ECS підтримує гнучку симуляцію складних ієрархічних конфігурацій кеш-пам’яті.

4. Збір і візуалізація статистичних даних.

У ході симуляції ECS акумулює й відображає ключові статистичні показники, зокрема:

- загальну кількість звернень до кешу на кожному рівні;
- кількість попадань і промахів для кожного ядра;
- розподіл звернень залежно від структури кеш-пам’яті (спільний або виділений кеш);
- детальний аналіз ефективності окремих параметрів кешу.

Ці статистичні дані дозволяють студентам порівнювати різні конфігурації кешів та оцінювати вплив архітектурних рішень на продуктивність.

## 2.1 Інтерфейс користувача ECS

Ефективне навчання з використанням програмного забезпечення значною мірою залежить від інтуїтивно зрозумілого і зручного графічного інтерфейсу. Кожен візуальний тренажер, розроблений для освітніх цілей, повинен забезпечувати легкість у користуванні та підтримку навчального процесу за допомогою візуальних елементів. Основною метою розробки симулятора ECS було створення простого, доступного та функціонального інтерфейсу, що не потребує додаткової підготовки для його опанування.

Інтерфейс користувача ECS реалізовано з використанням парадигми багатовіконного графічного інтерфейсу (MDI – Multiple Document Interface), що забезпечує гнучку взаємодію з декількома вікнами одночасно. Це особливо корисно у навчальному середовищі, де студент може паралельно працювати з різними етапами моделювання, конфігураціями або статистичними результатами.

Симулятор ECS підтримує два основні режими роботи:

- Design (Проєктування);
- Simulation (Моделювання).

У режимі проєктування (Design) користувачі можуть налаштовувати конфігурацію мультипроцесорної системи. Зокрема, студентам доступна можливість:

- визначати кількість процесорних ядер;
- встановлювати архітектуру кеш-пам’яті (кількість рівнів, асоціативність, розміри блоків, політика заміни тощо);
- конфігурувати відношення кешів до окремих ядер;
- зберігати створені конфігурації у файлах для подальшого використання.

У режимі моделювання (Simulation) користувачі можуть здійснювати симуляцію роботи системи на основі раніше збереженої або імпортованої конфігурації. Основні функції цього режиму включають:

- завантаження трасувального файлу (набору адрес пам’яті);
- запуск моделювання у двофазному режимі:
- автоматичне виконання з інтервалами часу між кроками, що дозволяє спостерігати динаміку;
- режим покрокового виконання, де студент сам ініціює кожну операцію моделювання, що сприяє глибшому розумінню алгоритмів роботи кешу.

Гнучкість інтерфейсу ECS дозволяє не лише моделювати типові архітектурні схеми, а й адаптувати симулятор до різного рівня складності навчальних завдань. Ця особливість робить його особливо корисним як для початкового знайомства з архітектурою кеш-пам’яті, так і для більш поглибленого аналізу взаємодії в багатоядерних системах.

### 2.1.1 Режим проєктування

У режимі проєктування студенти можуть налаштовувати параметри кеш-пам’яті та створювати екземпляри відповідних рівнів кешу з подальшим їх розподілом між ядрами мікросхеми. На рисунку 2.1 представлено загальний огляд інтерфейсу користувача симулятора ECS у цьому режимі.

Головне вікно інтерфейсу складається з двох основних панелей. Ліва панель надає студентам змогу обрати тип екземпляра рівня кешу, який вони бажають створити. Для кожного рівня кешу слід задати такі параметри: політика заміщення, асоціативність, обсяг кеш-пам’яті (у КБ або МБ), розмір рядка кешу, а також унікальний ідентифікатор (UID), який слугуватиме для подальшого посилання на цей рівень.

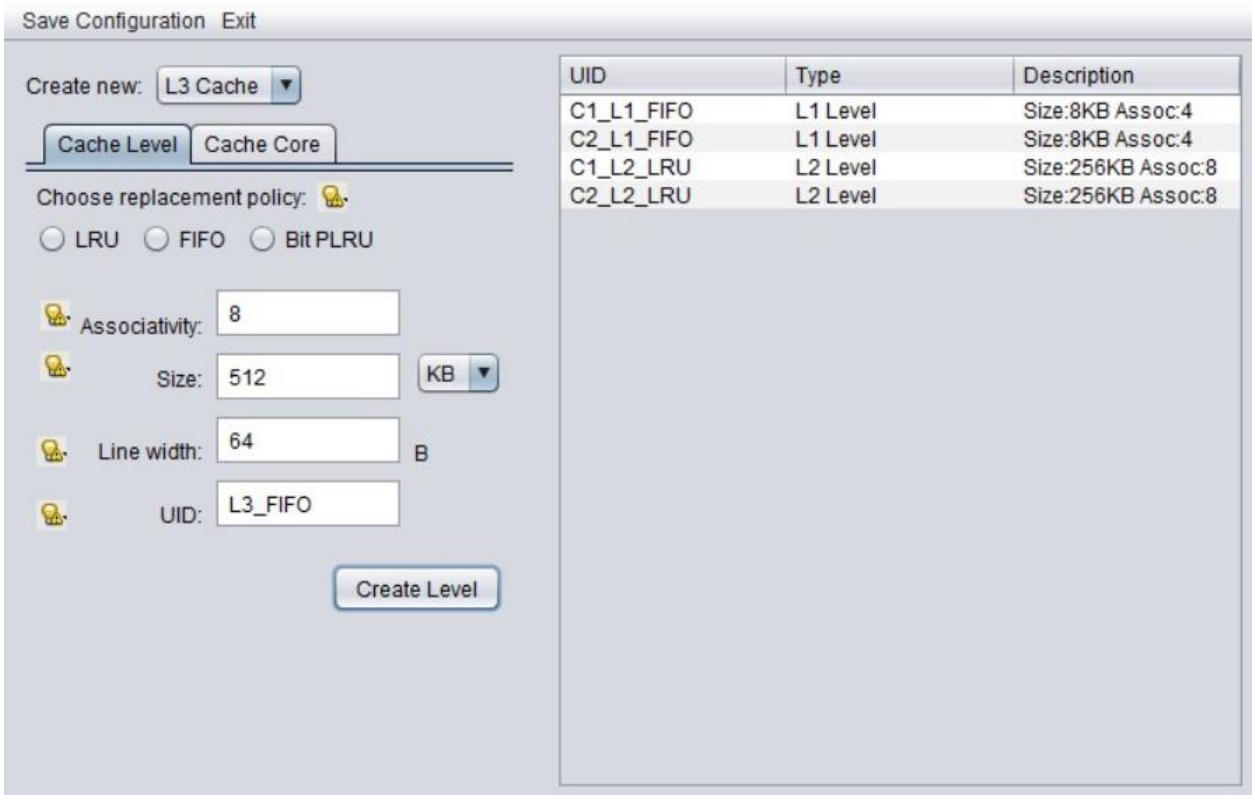


Рисунок 2.1 – Режим проєктування у симуляторі ECS. Процес створення екземпляра кешу L3

UID використовується при створенні екземпляра ядра мікросхеми для визначення того, які з раніше створених рівнів кешу буде призначено як кеші рівнів L1, L2 або L3 для відповідного ядра центрального процесора. На рисунку 2.1 також наведено приклад, що демонструє створення екземпляра кешу рівня L3 з такими параметрами: політика заміщення FIFO, 8-стороння асоціативність, розмір рядка кешу – 64 байти, загальний обсяг кешу – 512 КБ, UID – "L3_FIFO".

Права панель вікна відображає раніше створені екземпляри кешу з зазначенням їх типу, унікального ідентифікатора (UID) та коротким описом. У таблиці наведено, що було створено чотири екземпляри: два для кешу рівня L1 та два для рівня L2, що проілюстровано на рисунку 2.1.

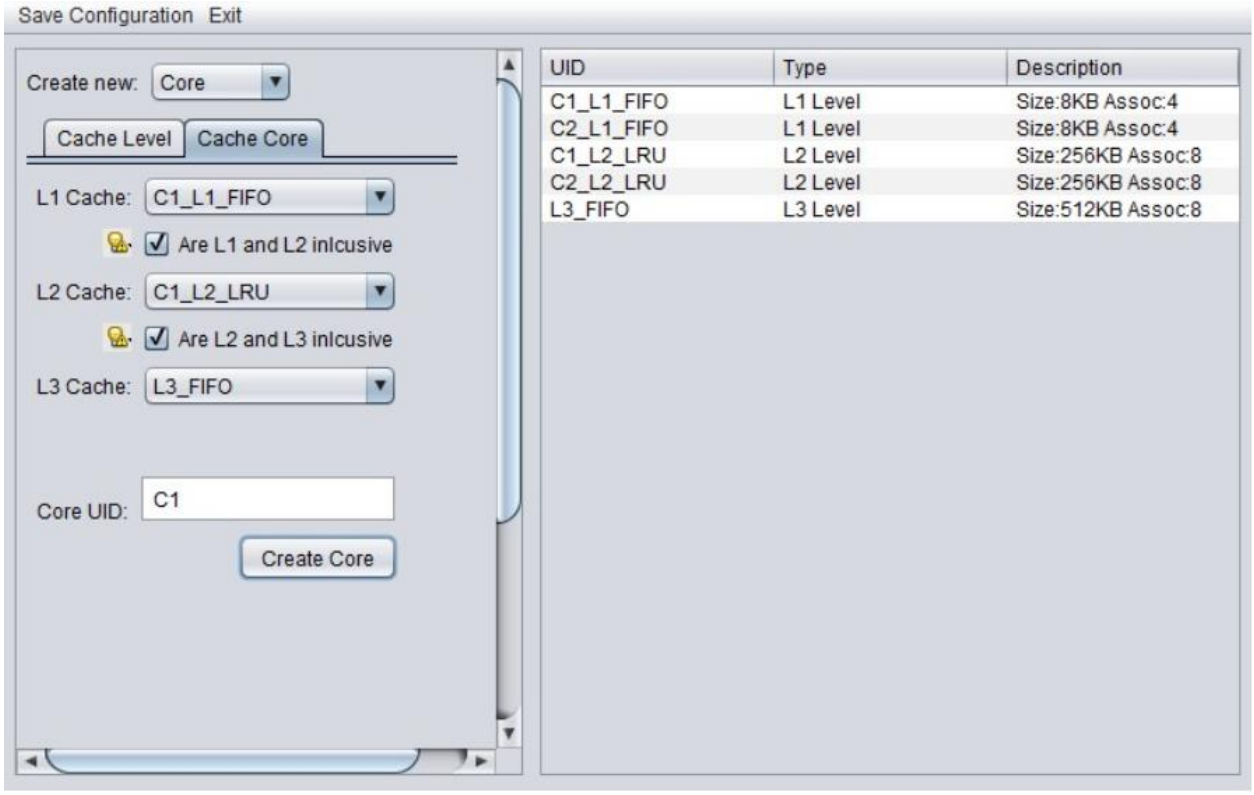


Рисунок 2.2 – Режим проєктування у симуляторі ECS. Процес створення ядра ЦП

Після формування екземплярів кешу студент може перейти до створення ядра процесора. Для цього необхідно вибрати відповідні

екземпляри кешу зі списку, поданого у правій панелі, і призначити їх для рівнів L1, L2 та L3. На рисунку 2.2 зображено приклад побудови ядра з UID C1, у якому кеш рівня L1 та L3 використовують політику заміни FIFO, а кеш рівня L2 – політику заміни LRU.

Симулятор ECS також надає можливість налаштування параметра інклюзивності між рівнями кеш-пам’яті, що дозволяє моделювати більш складні варіанти організації кешу в багаторівневій системі пам’яті.

Основною перевагою симулятора ECS порівняно з аналогічними інструментами є можливість проєктування процесорної системи з двома або більше ядрами (наприклад, C1, C2, C3 і C4), які можуть спільно використовувати один і той самий екземпляр рівня кешу. Зокрема, якщо для всіх ядер вибрати один і той самий екземпляр (наприклад, L3 FIFO) як кеш останнього рівня L3, утворюється спільний кеш цього рівня.

Крім того, можна реалізувати гнучке компонування, за якого, наприклад, два ядра спільно використовують кеш L3, а кеші L1 або L2 можуть бути розподілені або спільні для інших ядер. Такий підхід дозволяє моделювати різні конфігурації багатоядерних процесорів із варіативною структурою підсистеми кеш-пам’яті.

Зрештою, створену конфігурацію можна зберегти як файл, що репрезентує структуру проєктованого процесорного чіпа. Під час збереження студент вказує, які екземпляри ядер мають входити до складу чіпа, після чого система запропонує вибрати місце для збереження конфігураційного файлу. Формат цього файлу буде розглянуто у наступному підрозділі.

Для полегшення процесу навчання архітектурі комп’ютера та опанування функціоналу симулятора ECS передбачено додаткові інформаційні піктограми, розміщені біля кожного поля інтерфейсу. На рисунку 2.3 представлено ці піктограми, які надають короткі пояснення й інструкції щодо призначення відповідного поля, що підлягає налаштуванню. Водночас вони сприяють кращому розумінню особливостей та функціонального призначення параметрів кеш-пам’яті, які ці поля

представляють.

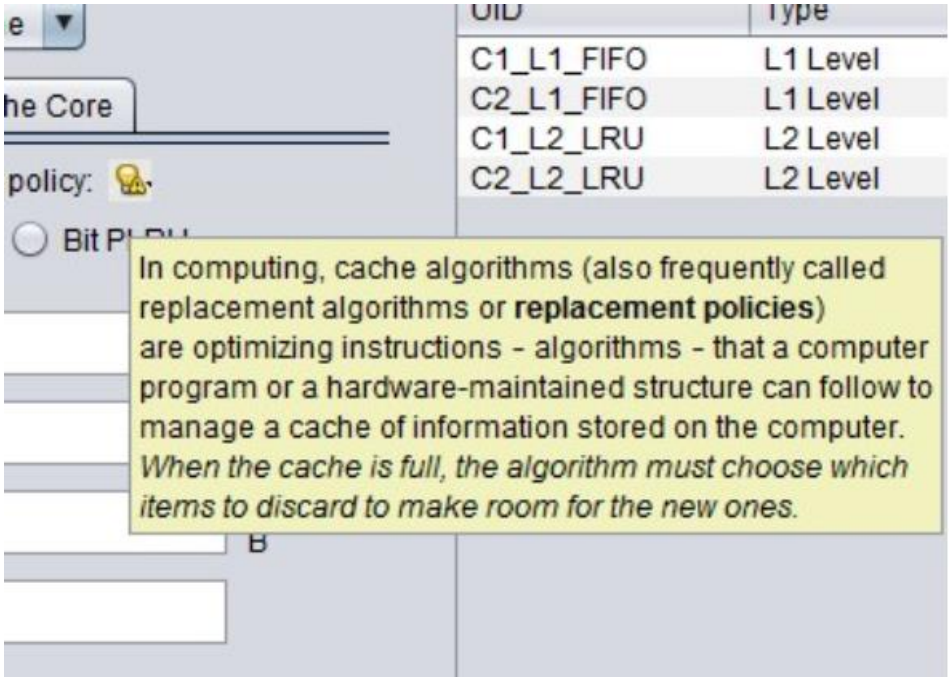


Рисунок 2.3 – Вікно пояснення підказки, що показує інформацію про політику заміни

Пояснення виводяться у вигляді спливаючих підказок при наведенні курсору або натисканні на піктограму. Такий підхід значно підвищує зручність користування та забезпечує навчальний ефект.

Після завершення роботи у режимі проєктування студенти отримують змодельований багатоядерний процесорний чіп із налаштованими рівнями кеш-пам’яті, відповідно до вказаних параметрів і конфігурацій, як описано в цьому розділі.

### 2.1.2 Режим моделювання

Режим проєктування ECS використовується для створення конфігурацій, а режим моделювання – для виконання симуляції та подальшого аналізу результатів. Після налаштування мікропроцесорної системи з кількома ядрами на чіпі та відповідною кеш-пам’яттю для кожного

ядра, користувачі повинні завантажити файл із послідовністю адрес пам’яті та ініціювати моделювання процедури доступу до цих адрес.

На рисунку 2.4 представлено загальний вигляд інтерфейсу режиму моделювання. Основне вікно містить такі структурні елементи:

- панель меню для керування процесом моделювання;
- вікно завантаження трасувального файлу з адресами пам’яті;
- детальне вікно виведення результатів;
- область візуального представлення результатів симуляції.

Нижче наведено опис основних функцій кожного з елементів інтерфейсу.

Панель меню керування моделюванням.

Центральним елементом керування режимом моделювання є рядок меню, який містить два основні розділи: Simulation та Construction, як показано на рисунку 2.5.

Меню Construction включає два пункти:

- Create New Configuration – відкриває режим проєктування для створення нової конфігурації;
- Load Configuration – дозволяє обрати файл конфігурації з файлової системи користувача та завантажити його в симулятор.

У меню Simulation передбачено можливість:

- завантажити файл із дослідницьким сценарієм (Load Study Case);
- завантажити трасувальний файл адрес пам’яті (Load Trace File);
- розпочати моделювання (Start Simulation);
- призупинити або зупинити симуляцію;
- перейти в покроковий режим виконання (Step by Step Simulation).

Усі опції меню Simulation, окрім пункту Load Study Case, є неактивними до моменту завантаження відповідної конфігурації мікропроцесорної системи. Активізація меню та керування симуляцією стають можливими лише після успішного імпорту конфігураційного файлу в симулятор ECS.

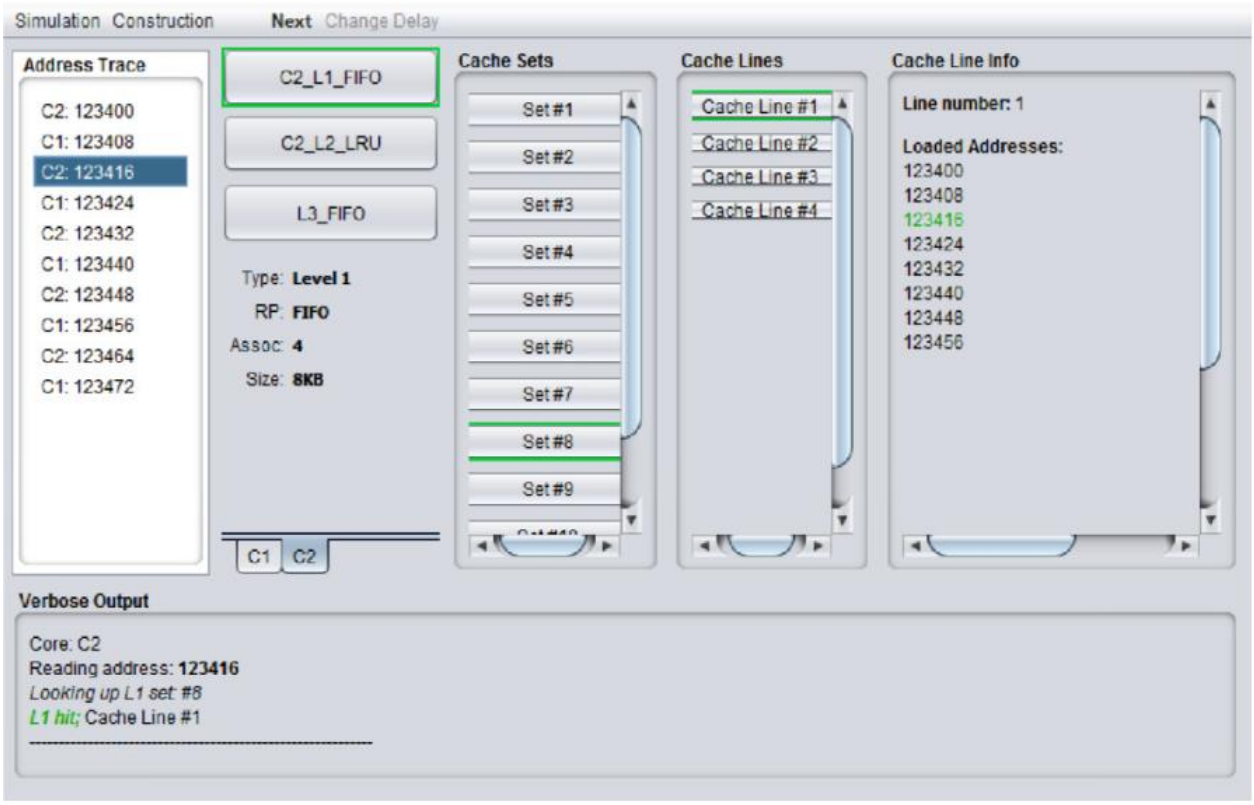


Рисунок 2.4 – Режим моделювання у симуляторі ECS. Попадання у рівень L1 ядра C2, набір №8, рядок №1, адреса 123416

Завантажений кадр трасування адрес.

Область трасування, розміщена у верхній частині лівої частини інтерфейсу, відображає вміст завантаженого трасувального файлу. Цей файл містить послідовність пар значень, де:

- перше значення – це унікальний ідентифікатор ядра (UID), що вказує, яке саме ядро ініціює доступ до пам’яті;

- друге значення – фізична адреса елемента даних, який підлягає завантаженню або доступу.

Під час симуляції активна адреса, що обробляється на поточному етапі, виділяється для полегшення її візуального спостереження та аналізу.

Детальний кадр виведення.

У нижній частині вікна розташована панель докладного виводу, яка містить опис усіх етапів пошуку в кеші, що відбуваються під час симуляції.

Симулятор ECS надає послідовне текстове пояснення кожного звернення до пам’яті, зокрема:

- фізичну адресу, до якої здійснюється доступ ядром;
- виконання пошуку в кеші рівня L1;
- вибір набору, до якого відображається відповідна адреса;
- результат звернення – попадання (hit) або промах (miss);
- номер кеш-рядка у випадку попадання;
- інформацію про витіснений рядок у разі заповненого набору та виникнення промаху.

Усі ці результати автоматично записуються до текстового файлу, який згодом може бути використаний студентом для глибшого аналізу або підготовки звіту.


Рисунок 2.5 – Рядок меню в режимі моделювання

Рамка візуального представлення.

Цей елемент інтерфейсу є ключовою складовою режиму моделювання й забезпечує графічне відображення процесу пошуку в кеш-пам’яті. Рамка візуального представлення складається з чотирьох логічних областей, кожна з яких відповідає за певний рівень деталізації у відображенні архітектури кешу:

- головна панель (Core Panel). Кожна вкладка цієї панелі відображає дані, пов’язані з конкретним ядром процесора, наприклад, C1 або C2 (як на рисунку 2.3). Вибір вкладки відкриває інформацію про відповідні рівні кешу, визначені під час проєктування. Ці рівні кешу представлено у вигляді кнопок з відповідними унікальними ідентифікаторами (UID). Натискаючи на відповідну кнопку, студент отримує доступ до вмісту кешу, який відображається на панелі «Набори кешу»;

- панель наборів кешу. Відображає усі набори кешу, які входять до складу обраного рівня кешу вибраного ядра. Кількість наборів відповідає конфігурації асоціативності, встановленій під час етапу проєктування;

- панель рядків кешу. Демонструє окремі рядки кешу в межах обраного набору. Рядки відображаються відповідно до архітектурних параметрів кешу, визначених користувачем;

- панель інформації про рядок кешу. Виводить адреси блоків, що зберігаються у вибраному рядку кешу. Також можливе відображення додаткової статистики, такої як кількість звернень до цього рядка (читання, записи) або поточний стан блоку.

Фрейм візуального представлення динамічно оновлюється під час кожного кроку моделювання. Послідовність дій у процесі моделювання виглядає наступним чином:

Крок 1. Зчитується чергова адреса пам’яті з трасувального файлу разом із UID ядра, яке ініціює звернення.

Крок 2. Запускається пошук у кеші першого рівня (L1), що належить відповідному ядру.

Крок 3. Визначається набір кешу на основі фізичної адреси, асоціативності кешу та кількості наборів.

Крок 4. У межах вибраного набору здійснюється пошук рядка, який містить потрібний елемент. Можливі два варіанти:

- потрапляння (hit): відповідний блок знайдено. Усі пов’язані елементи інтерфейсу (рядок кешу, набір, екземпляр кешу, вкладка ядра) підсвічуються

зеленим кольором;

- промах (miss): блок не знайдено. Відповідні елементи підсвічуються червоним кольором, після чого пошук продовжується у кеші наступного рівня (L2 або L3), поки елемент не буде знайдено або всі рівні кешу не будуть перевірені.

Усю текстову інформацію щодо проміжних і фінальних результатів моделювання (попадання/промахи, витіснення рядків, адреси, UID ядра тощо) буде представлено в реальному часі на панелі докладного виведення, а також збережено у звітному текстовому файлі.

### 2.1.3 Процедура пошуку

На рисунку 2.6 представлено послідовність дій при доступі до пам’яті. Адреса пам’яті зчитується з файлу трасування. Пошук розпочинається з перевірки кешу L1 обраного ядра. Вибір набору кешу здійснюється на основі фізичної адреси, рівня асоціативності та кількості наборів у кеші. Далі у рядках кешу відповідного набору виконується пошук необхідного елемента.

У разі знаходження елемента в одному з рядків кешу, відповідний рядок, набір кешу та екземпляр кешу L1 виділяються зеленим кольором на панелі інформації. Якщо ж елемент не знайдено в кеші L1, ті самі компоненти (рядки, набір, кеш) підсвічуються червоним кольором, після чого пошук продовжується у кеші L2 з використанням аналогічної послідовності кроків.

Одночасно формується докладний звіт, який відображається в нижній частині вікна симулятора.

У цьому розділі розглянуто дві типові ситуації: випадок потрапляння в кеш-пам’ять L1, а також ситуацію промаху кешу L1 із подальшим потраплянням у кеш нижчого рівня.

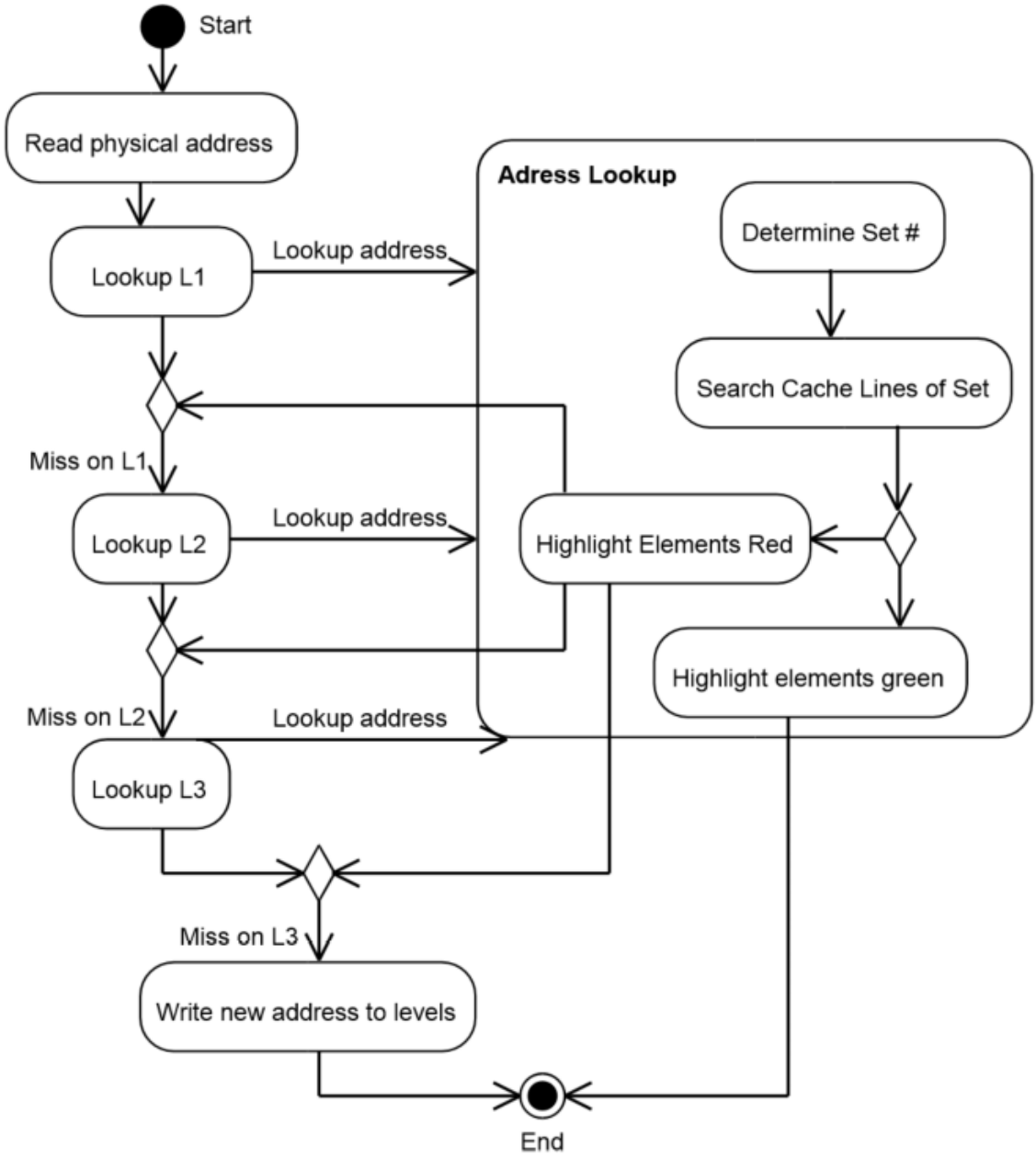


Рисунок 2.6 – Послідовність дій, виконанні яких оновлюється візуальне подання моделі кешу

Ситуація 1. Потрапляння в кеш L1.

На рисунку 2.4 представлено приклад звернення до кешу першого рівня (L1). У даному випадку друге ядро (C2) звертається до фізичної адреси 0x123416. Враховуючи розмір кешу L1, його асоціативність та кількість наборів, симулятор ECS визначає, що ця адреса повинна бути відображена у набір номер 8.

Набір №8 містить чотири рядки кешу. За два кроки до цього ядро C2

зверталося до адреси 0x12400, що спричинило запис у рядок кешу №1. При цьому до кешу було завантажено повний блок даних, який охоплює й адресу 0x123416. Таким чином, при повторному зверненні адреса вже знаходиться в кеші L1. Симулятор ECS індикує успішне потрапляння: відповідний рядок кешу (№1), набір (№8) та FIFO черги ядра C2 в кеші L1 підсвічуються зеленим кольором.

Ситуація 2. Промах кешу L1 і потрапляння в кеш L2.

На рисунку 2.7 показано сценарій, у якому відбувається промах у кеші L1. При спробі доступу до певної фізичної адреси, елемент не виявляється у кеші першого рівня. У такому випадку симулятор ECS підсвічує переглянуті компоненти L1 (рядки, набір, рівень кешу) червоним кольором. Після цього пошук продовжується у кеші другого рівня (L2).

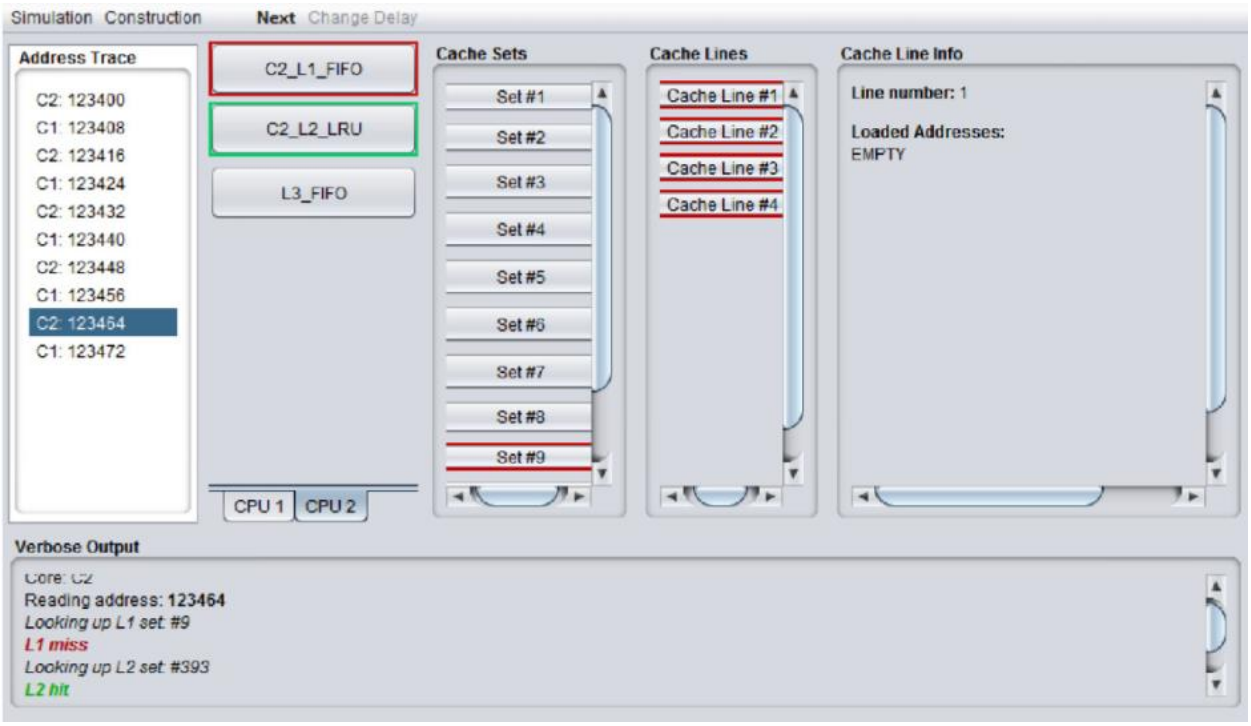


Рисунок 2.7 – Огляд режиму моделювання: промах L1 та попадання L2 під час зчитування адреси 123464

У наведеному прикладі необхідна адреса виявляється в кеші L2. Усі відповідні елементи кешу L2 (рядок, набір, рівень) відображаються зеленим кольором, що вказує на успішне завершення пошуку на цьому рівні. Повна

інформація про перебіг пошуку також відображається у нижньому вікні симулятора у вигляді детального звіту.

## 2.2 Методичні матеріали та демонстраційні приклади

У цьому підрозділі представлено низку демонстрацій та прикладів, що ілюструють використання симулятора ECS у навчальному процесі.

Запропоновані приклади охоплюють характерні шаблони доступу до пам’яті та забезпечують наочну візуалізацію, яка сприяє глибшому розумінню роботи процесора, архітектурних особливостей комп’ютерної системи та організації кеш-пам’яті.

Використання симулятора ECS дозволяє студентам в інтерактивному режимі простежити динаміку обробки запитів до пам’яті, побачити вплив асоціативності, розміру кешу та стратегії витіснення на ефективність роботи процесора.

Більш докладне дослідження реалізованих навчальних сценаріїв та експериментальних занять із застосуванням ECS [16] демонструє ефективність цього інструменту для пояснення ключових понять, пов’язаних із функціонуванням сучасного процесора та його багаторівневої кеш-пам’яті у рамках курсу «Архітектура комп’ютера».

### 2.2.1 Навчальні посібники

На основі існуючих навчальних посібників [19] було розроблено практичні лабораторні заняття для використання симулятора ECS. Наведені нижче вправи відображають основні можливості та призначення симулятора у навчальному процесі.

Вправа 1. Вступ до середовища ECS.

Ця вправа ознайомлює студентів із базовими поняттями кеш-пам’яті в контексті багатоядерних багаточипових процесорів, а також з інтерфейсом

симулятора ECS та можливостями налаштування конфігурації кешу.

Цілі навчання включають щонайменше:

- керування файлами конфігурації;
- моделювання виконання програм за допомогою трасувальних файлів;
- аналіз результатів та інтерпретація візуалізованих даних.

Вправа 2. Дослідження параметрів кешу.

Метою є вивчення впливу параметрів кеш-пам’яті, зокрема розміру, асоціативності та особливостей багаторівневої організації кешу.

Цілі навчання включають щонайменше:

- оцінку впливу параметрів кешу на продуктивність програми;
- аналіз локальності звернень у часі та просторі;
- створення множинних конфігурацій кешу та відповідних трасувальних файлів;
- проведення поглибленого аналізу результатів та обґрунтування впливу окремих параметрів.

Вправа 3. Асоціативність кешу та політики заміщення.

Вправа присвячена вивченню особливостей асоціативності кешу та алгоритмів заміщення – аспектів, які часто викликають труднощі у студентів.

Цілі навчання включають щонайменше:

- дослідження впливу параметрів асоціативності та політик заміщення на ефективність кешування;
- використання різних конфігурацій системи та трасування адрес пам’яті;
- інтерпретацію результатів симуляції та формулювання обґрунтованих висновків.

### 2.2.2 Демонстраційні приклади

Демонстраційні приклади базуються на спеціально підготовлених файлах, що містять посилання на конфігураційні параметри та відповідні трасувальні файли. Кожен приклад ілюструє конкретний шаблон доступу до пам’яті та слугує для кращого розуміння принципів роботи кеш-пам’яті на різних рівнях. Крім візуалізації динаміки кешування, ці приклади дозволяють оцінити середню кількість тактів на звернення до кожного рівня кешу. Нижче подано три характерні крайні сценарії кеш-поведінки.

Приклад 1. Постійне кеш-потрапляння.

Цей приклад демонструє ситуацію, в якій усі звернення до пам’яті успішно обслуговуються на поточному рівні кешу. Трасувальний файл створює шаблон адрес, які відповідають вмісту кешу, забезпечуючи максимальну ефективність. Прикладом може бути операція сортування невеликого масиву елементів, що повністю поміщається в кеш.

Приклад 2. Постійне кеш-промахання через обмежену ємність.

Даний сценарій моделює ситуацію, коли промахи кешу зумовлені недостатньою ємністю кеш-пам’яті. У цьому випадку шаблон звернень створює постійні конфлікти через заміщення даних, які ще можуть бути необхідні. Типовий приклад – доступ до елементів стовпця великої матриці, записаної в рядковому порядку. Такий підхід не дозволяє зберігати всі необхідні елементи в кеші, спричиняючи серію промахів.

Приклад 3. Постійне кеш-промахання через обмежену асоціативність.

Останній приклад ілюструє промахи кешу, спричинені недостатнім рівнем асоціативності. Шаблон звернень сконцентрований таким чином, що всі адреси відповідають одному набору кешу. При низькій асоціативності кешу (наприклад, 2- або 4-рядковому) виникають постійні заміщення, попри наявність вільних наборів в інших частинах кешу. Така ситуація характерна при доступі до елементів стовпців у матрицях, збережених у рядковому порядку (row-major). Вичерпний аналіз наслідків низької асоціативності

подано в роботі Гусєва і Рістова [4].

Більш докладний опис структури файлу демонстраційного прикладу наведено в підрозділі 2.5.

## 2.3 Інтерфейси ECS

ECS використовує ряд файлів для зберігання даних або використання їхнього вмісту як вхідних даних. Існує три типи файлів:

- файл конфігурації мікросхеми (CCF);
- файл трасування адреси (ATF);
- файл дослідження (SCF).

### 2.3.1 Файл CCF

Файл конфігурації кешу (Cache Configuration File, CCF) є ключовим елементом процесу моделювання симулятора ECS. Він створюється на етапі проєктування архітектури та слугує вхідним параметром у режимі моделювання. Структура CCF-файлу базується на форматі XML, що забезпечує зручність обміну даними з іншими інструментами симуляції та аналізу архітектурних рішень.

CCF-файл містить повний опис архітектури мікросхеми, включаючи кількість процесорних ядер, конфігурацію кеш-пам’яті кожного рівня та зв’язки між відповідними компонентами. Кореневим елементом є тег configuration, який містить два основних дочірніх теги: CacheLevels та CacheCores.

Елемент CacheLevels описує параметри кожного рівня кеш-пам’яті. Він повинен містити щонайменше один тег CacheLevel для кожного типу кешу (L1, L2, L3). Кожен CacheLevel включає такі обов’язкові поля:

- UID – унікальний ідентифікатор конкретного екземпляра кешу;

- Level – номер рівня кешу (1 – для L1, 2 – для L2, 3 – для L3);
- RP – політика заміни (наприклад, LRU, FIFO);
- Size – розмір кешу в байтах;
- LWidth – ширина кеш-рядка (line width) у байтах.

Елемент CacheCores містить інформацію про процесорні ядра, які беруть участь у моделюванні. Кожне ядро описується окремим тегом із наступними полями:

- UID – унікальний ідентифікатор ядра;
- L1 – UID екземпляра кешу L1, пов’язаного з даним ядром;
- L2 – UID екземпляра кешу L2, якщо присутній;
- L3 – UID екземпляра кешу L3, якщо наявний;
- L1InclL2 – логічне значення (істина або хибність), що вказує, чи кеш L1 включений у L2;
- L2InclL3 – логічне значення, що вказує, чи кеш L2 включений у L3.

У лістингу 2.1 представлено приклад структури такого файлу у форматі XML, що дозволяє наочно побачити ієрархію та взаємозв’язки елементів конфігурації.

Лістинг 2.1 – XML-структура CCF

```
<Configuration>
<CacheLevels>
        <CacheLevel>
<UID>id of     instance </UID>
<Level >[1 ,2 ,3] </ Level>
<RP>[FIFO , LRU, BPLRU] < /RP >
            <Size>number in bytes </Size>
            <LWidth> line width in bytes
            </LWidth>
        </CacheLevel>
        . .  .
        <CacheLevel>
            . . .
        </CacheLevel>
</CacheLevels>
<CacheCores>
        <Core>
<UID>UID of core </UID>
<L1>UID of L1</L1>
```

```
<L2>UID of L2</L2>
<L3>UID of L3</L3>
            <L1InclL2 >[true, false]</L1InclL2>
            <L2InclL3 >[true, false]</L2InclL3>
        </Core>
        . . .
        <Core> . . . </Core>
</CacheCores> </Configuration >
```

### 2.3.2 Файл ATF

Файл трасування адрес (Address Trace File, ATF) використовується симулятором ECS для моделювання звернень до пам’яті з боку різних ядер обчислювальної системи. Його структура є лінійною та надзвичайно простою, що забезпечує легкість створення та обробки навіть у навчальних умовах.

ATF-файл складається з послідовності рядків, кожен з яких містить два значення, розділені комами:

- перше значення – це унікальний ідентифікатор ядра (UID), який має відповідати UID, визначеному у відповідному CCF-файлі (Cache Configuration File);
- друге значення – фізична адреса елемента в основній пам’яті, до якої виконується звернення.

Кожен рядок файлу представляє одне звернення ядра до пам’яті. Файл може містити коментарі, які починаються з символу «%». Коментарі можуть розміщуватись на початку файлу або між зверненнями та використовуються для пояснення або маркування окремих ділянок трасування.

Лістинг 2.2 – Приклад файлу ATF із коментарем

```
\%Зразок файлу трасування адреси, припускаючи ядра C1 і C2
C1,  123392
C2,  123400
C1,  123408
C2,  123416
```

### 2.3.3 Файл SCF

Файл SCF (Study Case File) містить дані про дії, призначені для того, щоб студент міг спостерігати за роботою симулятора на основі заданої конфігурації мікросхеми та файлу трасування. Він має структуру XML, подібну до конфігураційного файлу CCF, і служить інтерфейсом між навчальними матеріалами та візуалізаційними можливостями симулятора ECS.

Кореневим елементом SCF-файлу є вузол StudyCase, який включає п’ять прямих дочірніх елементів:

- Title – назва навчального випадку;
- Goal – очікувана навчальна мета, якої має досягти студент;
- Activities – опис дій або кроків, що мають бути виконані;
- ChipConfig – URI (уніфікований ідентифікатор ресурсу) для файлу конфігурації мікросхеми (CCF);
- AddressTrace – URI для файлу трасування звернень до пам’яті (ATF).

Єдиним складним елементом є Activities. Він містить список вузлів Activity, кожен з яких описує окрему дію або інструкцію. Кожен Activity має два обов’язкових дочірніх елементи:

- Name – назва дії або етапу;
- Requirement – короткий опис вимог або спостережень, які повинен зробити студент.

Листинг 2.3 показує структуру файлів sCF.

Лістинг 2.3 – XML-структура SCF

```
<StudyCase>
        <Title >String </Title>
        <Goal> String </Goal>
        <Activities>
              <Activity>
<Name> String </Name>
                    <Requirement> String </Requirement>
              </Activity >
<Activity >    . . .     </Activity>
```

```
        </Activities>
        <ChipConfig>URI</ChipConfig>
        <AddressTrace>URI</AddressTrace>
</StudyCase>
```

### 2.3.4 Вихідний файл симулятора

Вихідний файл симулятора створюється під час виконання моделювання. По суті, це файл дампа для докладного журналу виводу, показаного на панелі докладного виводу. Під час моделювання пошук у кеші пояснюється у доступній для читання формі. Після завершення моделювання зібрана статистика додається до початку файлу, щоб студент, який шукає лише цю інформацію, не прокручував весь вихідний текст.

## 2.4 Висновок і подальша робота

Як допоміжний інструмент у процесі викладання курсу «Архітектура комп’ютера» розроблено новий візуальний симулятор ECS. Він може бути ефективно використаний студентами при вивченні відповідних концепцій функціонування кеш-пам’яті, особливо в сучасних багаточипових і багатоядерних процесорах.

Студенти мають можливість проводити експерименти, самостійно налаштовуючи різні параметри кешу. Це дає змогу:

- наочно зрозуміти основні концепції обробки в сучасних багатоядерних багаторівневих архітектурах та організації кешу;
- моделювати діяльність у кеші, тобто промахи та попадання у кеш-пам’ять у певному наборі кешу та в пам’яті під час послідовного або паралельного виконання алгоритму.

Переваги нового розробленого симулятора ECS є численними. Він має низку переваг порівняно з іншими навчальними тренажерами в галузі архітектури комп’ютерів та їх організації. Зокрема, використовуючи симулятор ECS, студенти можуть в інтерактивному режимі ефективно

засвоювати такі поняття:

- організація та обробка в ієрархії кешу (від рівня L1 до L3);
- організація та спільне використання кеш-пам’яті шляхом ідентифікації власників (виділений кеш для окремого ядра чи спільний для кількох ядер);
- концепція інклюзивності між різними рівнями кешу;
- вплив параметрів кешу, зокрема:
  а) розмірів кеш-пам’яті;
  б) асоціативності кеш-набору;
  в) розмірів рядків кешу;
- ефект використання різних політик заміни кешу, а також можливість тестування таких політик на різних рівнях кеш-пам’яті.

Загальна перевага продуктивності від впровадження симулятора ECS полягає в тому, щоб:

- надати студентам можливість глибоко опанувати основи архітектури та організації комп’ютера;
- реалізовувати експерименти з оцінки продуктивності програмних систем.

Очікуваний ефект полягає в тому, що студенти легко зрозуміють важливість комп’ютерної архітектури та підвищать свою готовність вивчати курси, засновані на апаратному забезпеченні, крім програмного забезпечення.

Використання симулятора ECS не обмежується курсом «Архітектура комп’ютера». Помічено, що його можна використовувати для вивчення апаратних курсів загалом, завдяки його особливостям та інтерфейсу користувача. Остаточний вплив – це підвищення бажання студентів навчатися на курсах, пов’язаних з апаратним забезпеченням, і краще та швидше вивчати відповідні теми. Можні використовувати симулятор ECS для подальших досліджень і, наприклад, знайти оптимальну апаратну платформу для максимізації швидкості та прискорення кеш-інтенсивних

алгоритмів для послідовного та паралельного виконання. Оскільки він спочатку розроблений для студентів, можна зробити наступну версію ECS безкоштовним інструментом, реалізованим як хмарне рішення.

# 3 ВИВЧЕННЯ КОГЕРЕНТНОСТІ КЕШ-ПАМ’ЯТІ ЗА ДОПОМОГОЮ МОДУЛЯ СИМУЛЯТОРА ПРОТОКОЛУ MESI

Модуль MESI симулятора кеш-пам’яті ECS є програмним інструментом, реалізованим мовою Java. Він розроблений спеціально для навчальних цілей і дозволяє демонструвати принципи роботи протоколу MESI, який застосовується для підтримки когерентності кеш-пам'яті в багатоядерних системах.

Модуль забезпечує можливість налаштування кількості рівнів кешу. Для кожного рівня користувач може задати параметри кеш-пам’яті, зокрема: ємність кешу; розмір рядка кешу; ступінь асоціативності; політику заміни; кількість слів на доступ до пам’яті; політику запису тощо.

Крім того, користувач має змогу конфігурувати параметри статистики доступу до пам’яті, зокрема типи промахів кешу, загальну кількість звернень, кількість промахів, типи операцій пам’яті тощо. Результати можуть відображатися у зручному для аналізу форматі.

Код моделювання вводиться у вигляді опорної послідовності звернень до пам’яті.

Після цього користувач може вказати:

- моменти виникнення переривань введення/виведення або DMA;
- адресу початкового блоку пам’яті для операції введення/виведення;
- кількість слів, які необхідно перемістити [3].

## 3.1 Протокол MESI

Протокол MESI, реалізований у модулі MESI симулятора кеш-пам’яті ECS, дозволяє підтримувати когерентність даних у кешованих системах. Він ґрунтується на чотирьох логічних станах, які може мати кожен блок кеш-пам’яті. Назва протоколу є абревіатурою від англійських назв чотирьох

станів: Modified (M), Exclusive (E), Shared (S), Invalid (I). У таблиці 3.1 наведено характеристики кожного стану.

Опис станів:

- «недійсний» (I): блок недійсний; дані або відсутні в кеші, або застарілі, оскільки інший процесор змінив відповідну область пам’яті;
- «спільний» (S): блок дійсний і може дублюватися в інших кешах; всі копії містять актуальні дані, але вони не змінені;
- «ексклюзивний» (E): блок присутній лише в одному кеші, його вміст відповідає основній пам’яті, але не був змінений;
- змінений (M): блок змінений у кеші й більше ніде не дублюється; вміст у основній пам’яті є застарілим, а актуальна копія – лише в кеші.

Таблиця 3.1 – Характеристики станів протоколу MESI

| Назва стану | M (змінений) | E (ексклюзивний) | S (спільний) | I (недійсний) |
|---|---|---|---|---|
| Чи дійсний рядок? | так | так | так | ні |
| Копія в пам’яті | застаріла | дійсна | дійсна | – |
| Поведінка на шині | не видається | не видається | видається | не видається |
| Наявність у інших кешах | ні | ні | може бути | може бути |

Стан кожного блоку кеш-пам’яті в модулі MESI симулятора кеш-пам’яті ECS може змінюватися залежно від дій, які виконує центральний процесор [17]. На рисунку 3.1 представлено діаграму переходів між станами, яка ілюструє логіку роботи протоколу MESI.

Хоча рисунок 3.1 є достатньо наочним, необхідно додатково пояснити логіку переходів. У початковому стані, коли кеш порожній, і центральний процесор виконує операцію запису блоку пам’яті до кешу, цей блок отримує

стан ексклюзивний, оскільки жодна інша копія блоку ще не зберігається в інших кешах. Якщо потім процесор змінює вміст цього блоку, його стан переходить у змінений, оскільки блок залишається лише в одному кеші, але його вміст вже не відповідає основній пам’яті.

У разі, коли блок перебуває у стані ексклюзивний, і інший процесор виконує операцію читання цього блоку, він буде завантажений у кеш цього процесора. Таким чином, блок виявляється збереженим у кількох кешах одночасно, тому його стан змінюється на спільний.

Натомість, якщо центральний процесор намагається записати в блок, що наразі перебуває у стані змінений в іншому кеші, цей блок спершу потрібно витіснити з кешу, де він міститься, та зберегти його оновлену версію в основній пам’яті. Лише після цього блок може бути завантажений до кешу процесора, що ініціює запис, у стані ексклюзивний, оскільки саме він тепер має найактуальнішу копію даних.

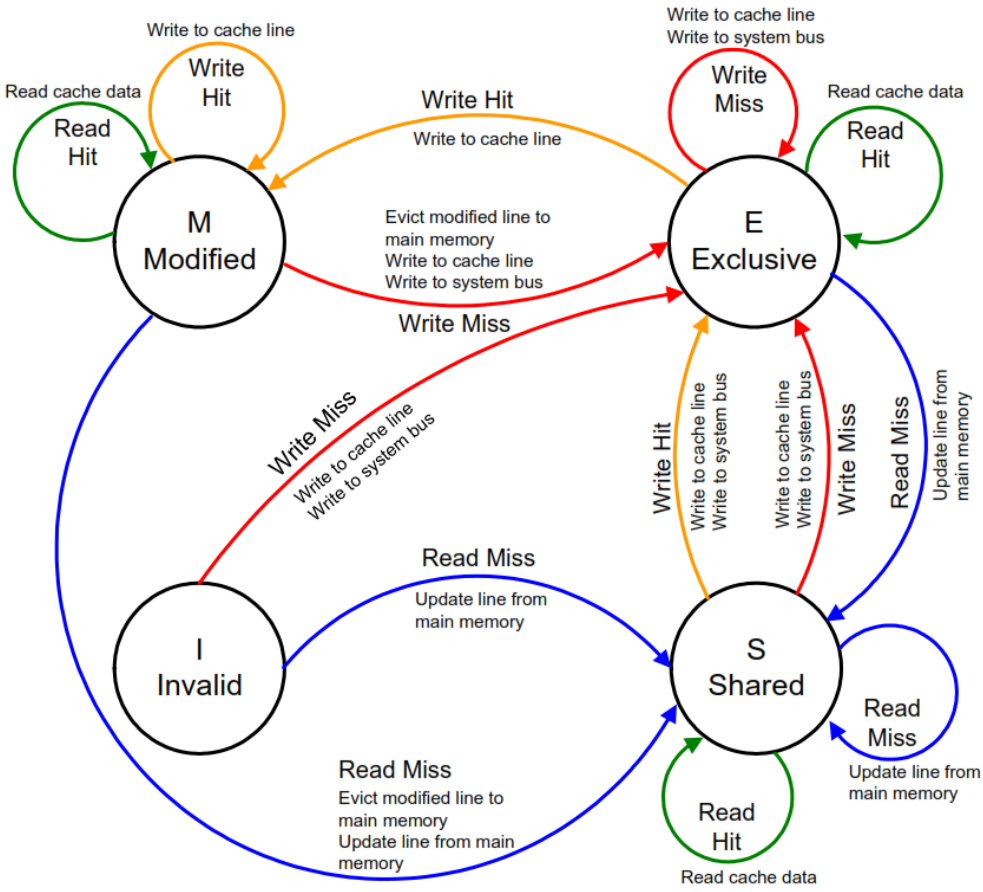


Рисунок 3.1 – Переходи станів кешу, ініційовані локальним ЦП

Якщо центральний процесор намагається прочитати блок, але не знаходить його у власному кеші, це може свідчити про наявність актуальнішої копії в іншому кеші. У такому випадку система повинна ініціювати процедуру витіснення блоку з кешу, де він наразі зберігається, та оновити його в основній пам’яті. Після цього блок зчитується з пам’яті, завантажується до кешу процесора, що ініціює читання, та отримує стан спільний, оскільки відтепер існують щонайменше дві дійсні копії блоку в системі.

Інший сценарій передбачає, що центральний процесор виконує запис у блок, який перебуває у стані спільний. У такому випадку блок змінює свій стан на ексклюзивний, оскільки після операції запису лише один кеш зберігає актуальну копію даних.

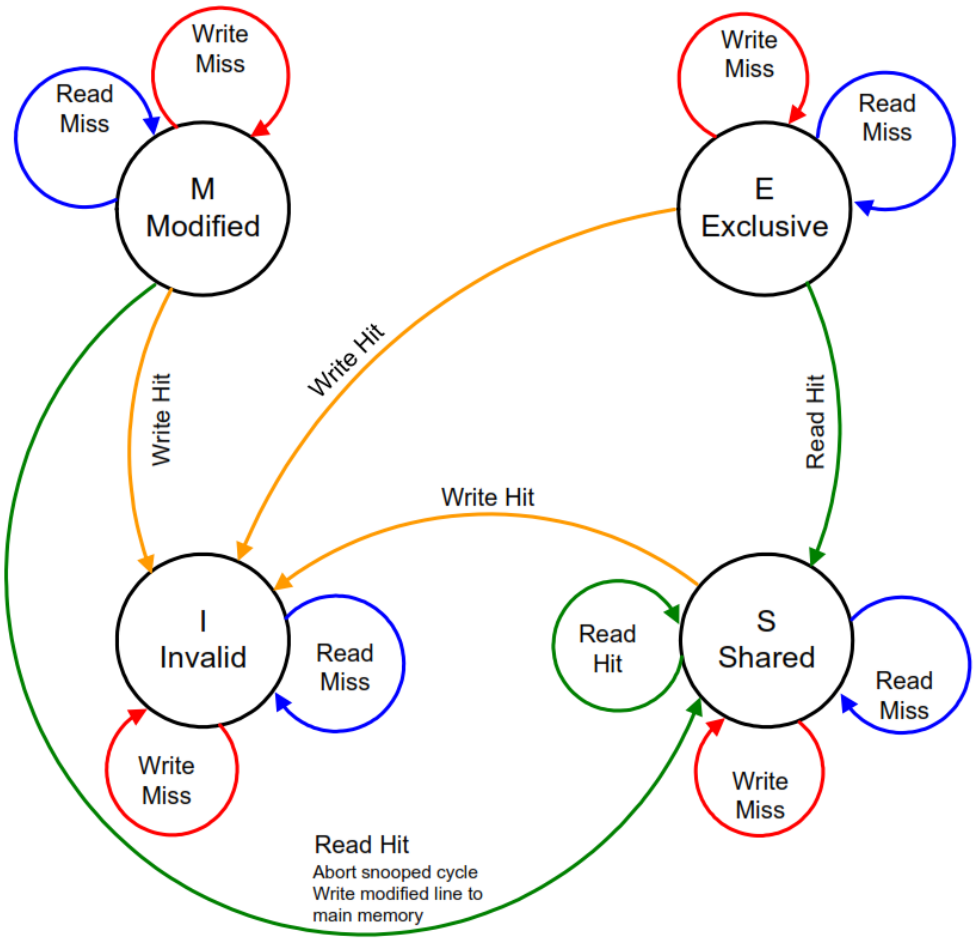


Рисунок 3.2 – Переходи станів кешу, ініційовані з системної шини

Слід враховувати, що стан кеш-блоку може змінюватися не лише внаслідок дій самого процесора, а й у результаті зовнішніх подій, таких як переривання введення/виведення або доступ через Direct Memory Access (DMA). Усі можливі переходи між станами в таких випадках зображено на рисунку 3.2.

Таким чином, процесор завжди оперує дійсними даними, гарантуючи коректність виконання обчислювальних операцій. У випадках, коли процесор модифікує дані, що зберігаються в основній пам’яті, оновлені значення зберігаються у кеш-пам’яті, забезпечуючи швидкий доступ до найактуальнішої інформації. Завдяки використанню протоколу MESI забезпечується когерентність кеш-пам’яті, тобто процесор у кожному випадку отримує найсвіжіші значення даних, необхідні для обробки.

## 3.2 Симуляція MESI

### 3.2.1 Можливості модуля MESI симулятора кеш-пам'яті ECS

Модуль MESI симулятора кеш-пам'яті ECS є програмним засобом навчального призначення, який моделює виконання програмного коду в умовах кешованої багатопроцесорної системи з підтримкою протоколу когерентності MESI. Цей модуль дає змогу візуалізувати динаміку зміни станів кеш-пам’яті під час паралельного доступу до спільних даних.

Інтерфейс модуля є інтуїтивно зрозумілим і доступним для користувача, що забезпечує легкість у використанні. Для запуску симуляції користувачеві необхідно ввести еталонну послідовність пам’яті, яка інтерпретується як дії центрального процесора під час виконання програми. Такі дії включають:

- зчитування інструкцій;
- зчитування даних;
- запис даних;

У процесі виконання моделюються також переривання, пов’язані з введенням-виведенням або ініціативами з боку інших процесорів чи пристроїв DMA. Формат введення таких переривань реалізується у вигляді спеціальних інструкцій у вхідному коді.

### 3.2.2 Методика використання симулятора

Модуль MESI симулятора ECS передбачає широкі можливості конфігурування, що дозволяє адаптувати середовище до конкретних навчальних цілей. Як показано на рисунку 3.3, користувач має доступ до панелі налаштувань, де можна вказати основні параметри:

- об’єм кеш-пам’яті;
- розмір рядка кешу;
- ступінь асоціативності;
- політика заміни;
- кількість слів на одне звернення;
- політика запису.

Крім цього, користувач може налаштувати параметри збору статистики: кількість звернень, кількість промахів, типи промахів, кількість зчитувань і записів у пам’ять.

Симулятор надає два режими виконання:

- автоматичний режим, при якому виконання симуляції розпочинається одразу після натискання кнопки запуску;
- покроковий режим, що дозволяє студенту спостерігати зміну станів системи після кожної інструкції, що особливо корисно при вивченні логіки переходів між станами протоколу MESI.

У нижній частині інтерфейсу візуалізуються чотири стани MESI (Modified, Exclusive, Shared, Invalid). Під час виконання коду відповідний стан блоку кешу підсвічується, що забезпечує зручну візуалізацію для користувача. Ця функція особливо ефективна в поєднанні з покроковим

виконанням симуляції.

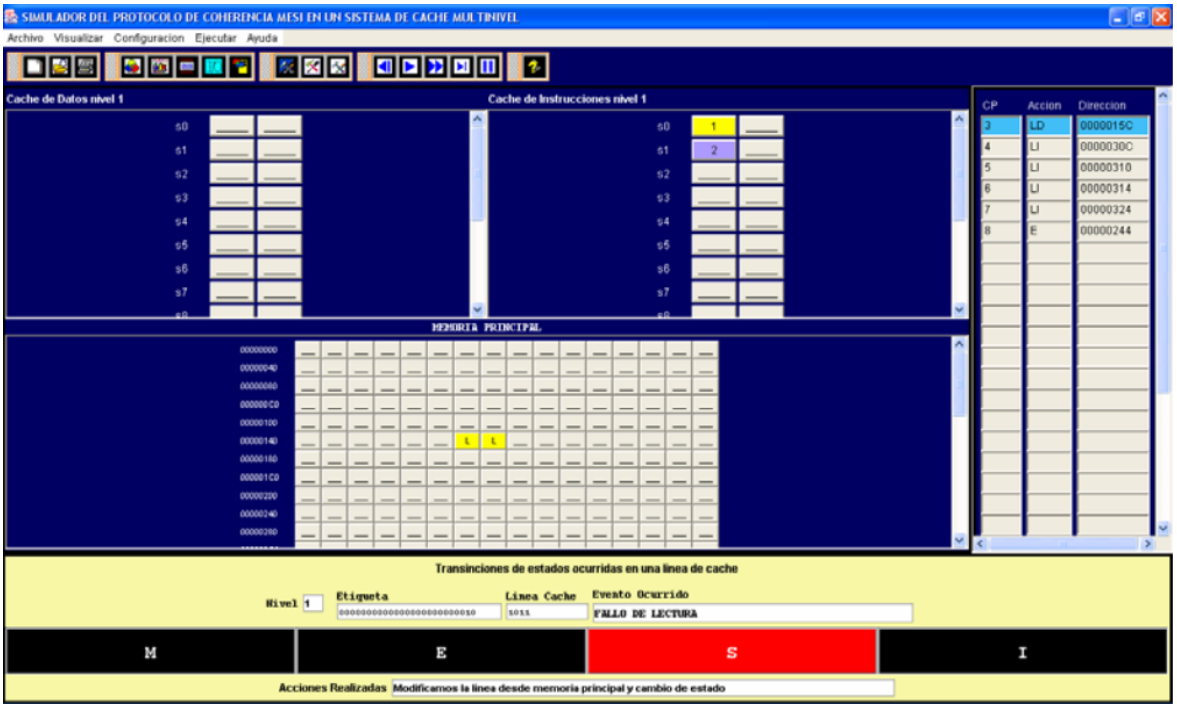


Рисунок 3.3 – Інтерфейс модуля симулятора MESI

По завершенню симуляції користувач отримує статистичні результати, які можуть бути використані для аналізу ефективності роботи протоколу MESI у різних конфігураціях. Завдяки зрозумілому інтерфейсу та інформативному вихідному журналу, модуль може бути використаний як студентами, так і фахівцями з відповідною технічною підготовкою.

### 3.2.3 Генерація конфліктів у доступі до пам'яті

Опція «I/O» у модулі MESI симулятора кеш-пам’яті ECS дозволяє користувачеві задавати сигнали введення/виведення, а також визначати моменти їх активації під час симуляції. Ця функція є корисною для моделювання сценаріїв із участю пристроїв введення-виведення або дій інших процесорів у багатопроцесорному середовищі.

Окрім цього, у симуляторі реалізована опція візуалізації, яка забезпечує доступ до повної статистичної інформації, сформованої на основі виконання

моделювання. Користувач може переглядати:

- статистику звернень до пам’яті, включно з кількістю промахів кешу та їх класифікацією;
- вміст основної пам’яті;
- вміст буфера введення/виведення;
- графічне відображення результатів у вигляді діаграм або візуальних схем.

Ці інструменти забезпечують глибший рівень розуміння принципів функціонування протоколу когерентності та дозволяють студентам проводити більш детальний аналіз роботи системи в різних умовах.

### 3.2.4 Приклади виконання

Наведені нижче приклади демонструють реакцію модуля MESI симулятора кеш-пам’яті ECS на виконання коду реального програмного забезпечення. Як приклад було використано алгоритм множення матриць. Розглянуто дві відмінні конфігурації кеш-пам’яті.

Конфігурація 1 – кеш-пам’ять рівня L1

Параметри:

- розмір блоку: 8;
- об’єм кешу: 1 КБ;
- алгоритм заміщення: LRU;
- тип відображення: асоціативне за групами (8-стороннє);
- тип кешу: уніфікований.

Ця конфігурація забезпечує 91% попадань у кеш. Незважаючи на її простоту, рівень відмов є досить низьким. Це зумовлено тим, що в даній реалізації майже не виникає необхідності в заміщенні блоків кешу – одного з основних джерел промахів. Успішність також залежить від політики заміщення: алгоритм LRU (Least Recently Used) зменшує ризик видалення блоків, які невдовзі можуть знадобитися.

Конфігурація 2 – ієрархічна структура кеш-пам’яті.

Параметри рівня L1:

- розмір блоку: 4;
- об’єм кешу: 1 КБ;
- алгоритм заміщення: випадковий;
- тип відображення: пряме;
- тип кешу: змішаний.

Параметри рівня L2:

- розмір блоку: 8;
- об’єм кешу: 4 КБ;
- алгоритм заміщення: LRU;
- тип відображення: асоціативне за групами (8-стороннє);
- тип кешу: уніфікований.

Результати виконання цієї конфігурації показали 84 % попадань у кеш L1 та 43 % – у кеш L2. Було зафіксовано 7 промахів кешу першого рівня, що спричинили звернення до кешу другого рівня. Оскільки блок L2 має більший розмір, при кожному промаху на цьому рівні завантажується більше слів, ніж у L1. Це підвищує ймовірність знаходження потрібних даних у кеші другого рівня, незважаючи на нижчий відсоток попадань.

Друга конфігурація демонструє вищу кількість збоїв у порівнянні з попередньою. Основні причини полягають у наступному:

- розмір блоку: у першій конфігурації кожен збій призводив до завантаження восьми сторінок у кеш, тоді як у цій – лише чотирьох, що знижує ймовірність повторного використання вже завантажених даних;

- політика заміни: на рівні L1 використовується випадкова політика, яка демонструє гірші результати порівняно з політикою LRU;

- спосіб розподілу кешу: змішана організація призводить до більшої кількості збоїв, ніж уніфікована, оскільки при змішаній структурі під час кожного звернення кешу система завантажує або блок даних, або інструкцій; натомість уніфікований кеш дозволяє завантажити обидва типи даних

одночасно.

### 3.3 Результати експерименту

Модуль MESI симулятора кеш-пам’яті ECS застосовувався у реальному навчальному процесі. До участі в експерименті було залучено 60 студентів, які працювали з 4 різними конфігураціями симулятора. Протягом 10 годин навчальних занять було виконано приблизно 1200 тестів, що у середньому становить 120 тестів на годину. Загальний відсоток успішно виконаних тестів становив 92,5 %. Решта 7,5 % тестів завершилися невдачею з таких причин:

- неправильне введення даних;
- синтаксичні помилки у коді.

У подальших версіях симулятора планується впровадити попередню компіляцію з виявленням таких помилок, що дозволить завчасно їх ідентифікувати та запропонувати виправлення.


Рисунок 3.4 – Екран статистики

Студенти мали можливість експериментувати з різними конфігураціями кеш-пам’яті та параметрами сигналів введення-виведення, що дало змогу наочно продемонструвати кожен можливий перехід станів у

межах протоколу MESI. За результатами виконання завдань студенти формували власні висновки у підсумкових звітах, що свідчить про глибоке розуміння принципів функціонування когерентності кешу.

Таблиця 3.1 – Статистика загального використання та для конкретного студента

| | Загальне | Студ_1 | | Загальне | Студ_1 |
|---|---|---|---|---|---|
| З'єднання | 1187 | 21 | Переходи E–M | 151 | 3 |
| Не вдалося | 89 | 2 | Переходи E–S | 120 | 2 |
| Успішно | 1098 | 19 | Переходи E–I | 117 | 2 |
| Переходи I–S | 141 | 3 | Переходи M–S | 76 | 1 |
| Переходи I–E | 145 | 2 | Переходи M–E | 49 | 1 |
| Переходи S–I | 143 | 2 | Переходи M–I | 46 | 1 |
| Переходи S–E | 109 | 2 | | | |

У таблиці 3.1 наведено статистичні показники використання симулятора як у загальному випадку для всіх студентів, так і для одного студента (умовно позначеного як Студ_1), обраного навмання.

# ВИСНОВКИ

Розроблене програмне забезпечення довело свою ефективність як навчальний інструмент для візуалізації процесів, пов’язаних із використанням кеш-пам’яті під час виконання програм. Симулятор спеціально орієнтований на вивчення таких фундаментальних аспектів кешування, як ємність кеш-пам’яті, розмір рядка, ступінь асоціативності, політика заміщення та інші параметри. Окрім цього, програмне забезпечення демонструє принципи роботи з перериваннями введення/виводу, наочно ілюструючи, як ці процеси впливають на передачу даних між пристроями та пам’яттю.

Особливу увагу в симуляторі приділено демонстрації роботи протоколу когерентності кеш-пам’яті MESI. Завдяки поетапному візуальному представленню кожного з можливих станів і переходів у протоколі, користувачі мають змогу глибоко зрозуміти логіку когерентного управління даними у багатоядерних системах. Результати експериментального впровадження симулятора у навчальний процес підтвердили його доцільність та високу ефективність у реальному аудиторному середовищі.

Модуль MESI симулятора кеш-пам’яті ECS став першим етапом у реалізації комплексного підходу до створення педагогічних засобів навчання сучасним архітектурам комп’ютерних систем. Практична значущість підтверджується тим, що студенти, взаємодіючи із симулятором, демонструють покращене розуміння теоретичних концепцій завдяки набуттю практичного досвіду моделювання.

У перспективі планується розширення функціональних можливостей симулятора шляхом впровадження підтримки інших протоколів когерентності кешу, зокрема MOESI, N+1, Futurebus+, Berkeley та інших, що відкриє нові можливості для поглибленого вивчення когерентності в системах з ієрархічною пам’яттю.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. R. Shackelford, A. McGettrick, R. Sloan, H. Topi, G. Davies, R. Kamali, J. Cross, J. Impagliazzo, R. LeBlanc, and B. Lunt, “Computing curricula 2005: The overview report,” SIGCSE Bull., vol. 38, no. 1, pp. 456-457, Mar. 2006. http://dx.doi.org/ 10.1145/1124706.1121482

2. S. Ristov, M. Stolikj, and N. Ackovska, “Awakening curiosity – hardware education for computer science students,” in MIPRO, 2011 Proc. of the 34th Int. Convention, IEEE Conf. Publications, 2011, pp. 1275 -1280.

3. J. L. Hennessy and D. A. Patterson, “Computer Architecture, Fifth Edition: A Quantitative Approach,” MA, USA, 2012.

4. M. Gusev and S. Ristov, “Performance gains and drawbacks using set associative cache,” Journal of Next Generation Information Technology (JNIT), vol. 3, no. 3, pp. 87-98, 31 Aug 2012. http://dx.doi.org/10.4156/jnit.vol3.issue3.9

5. D. A. Patterson and J. L. Hennessy, “Computer organization and design, forth edition: The hardware/software interface,” MA, USA, 2009.

6. B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic, “A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization,” Education, IEEE Transactions on, vol. 52, no. 4, pp. 449 -458, nov. 2020. http://dx.doi.org/10.1109/TE.2008.930097

7. D. Patti, A. Spadaccini, M. Palesi, F. Fazzino, and V. Catania, “Supporting undergraduate computer architecture students using a visual mips64 cpu simulator,” Education, IEEE Transactions on, vol. 55, no. 3, pp. 406 -411, aug. 2012. http://dx.doi.org/10.1109/TE.2011.2180530

8. J. Edler and M. D. Hill, “Dinero iv trace-driven uniprocessor cache simulator,” 2012. [Online]. Available: http://pages.cs.wisc.edu/ !markhill/DineroIV/

9. B. B. Fraguela, R. Doallo, and E. L. Zapata, “Automatic analytical modeling for the estimation of cache misses,” in Proc. of the Int. Conf. on Parallel

Architecture and Compilation Techniques (PACT '99). IEEE Comp. Society, 1999, pp. 221-231.

10. A. Jaleel, R. S. Cohn, C.-K. Luk, and B. Jacob, “Cmpsim: A pin- based on-the-fly multi-core cache simulator,” in The Fourth Annual Workshop MoBS, co-located with ISCA ’08, 2008.

11. Y.-T. Chen, J. Cong, and G. Reinman, “Hc-sim: a fast and exact l1 cache simulator with scratchpad memory co-simulation support,” in Proc. of the 7-th IEEE/ACM/IFIP Int. conf. on HW/SW codesign and system synthesis (CODES+ISSS ’ll). USA: ACM, 2011, pp. 295-304.

12. E. Herruzo, J. Benavides, R. Quislant, E. Zapata, and O. Plata, “Simulating a reconfigurable cache system for teaching purposes,” in Microelectronic Systems Education (MSE ’07). IEEE Int. Conf. on, 2007, pp. 37 - 38.

13. A. Misev and M. Gusev, “Visual simulator for ILP dynamic OOO processor,” in WCAE '04, Proc. of the workshop on Computer architecture education: in conduction with the 31st Int. Symposium on Computer Architecture, E. F. Gehringer, Ed. ACM, USA, 2004, pp. 87 -92.

14. Valgrind, “System for debugging and profiling linux programs,” [retrieved: Nov, 2012]. [Online]. Available: http://valgrind.org/

15. C. Hamacher, Z. Vranesic, S. Zaky, Organization of Computers. McGraw-Hill/Interamericana of Spain S.A. 2003. 804 p. ISBN: 84-481-3951-8.

16. S. Ristov, B. Atansovski, M. Gusev, and N. Anchev, “Hands-on exercises to support computer architecture students using educache simulator,” University Sts Cyril and Methodius, Faculty of Computer Science, Tech. Rep. IIT-35-2013, 2013.

17. J. Handy, The Cache Memory Book. 2nd Ed. Academic Press. 1998. 229 p. ISBN: 0-12-322980-4.