Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-24М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Бершадський О.С.
«____» _____ 2025 р.

Керівник проекту
доктор технічних наук, професор
_____ Коваленко О.В.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

АНОТАЦІЯ

Бершадський О.С. Дослідження та програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Метою розробки є дослідження та програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Об'єктом дослідження є процес емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Предметом дослідження є методи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Методи дослідження базуються на методах комп'ютерної логіки, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, логічна адреса, лінійна адреса, фізична адреса

ABSTRACT

Bershadsky O.S. Research and software implementation of the emulator system for converting logical addresses into linear and linear to physical for educational purposes. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the second (master's) level of higher education, software has been developed that is intended for the emulator system for converting logical addresses into linear and linear to physical for educational purposes.

The purpose of the development is the research and software implementation of the emulator system for converting logical addresses into linear and linear to physical for educational purposes.

The object of the research is the process of the emulator for converting logical addresses into linear and linear to physical for educational purposes.

The subject of the research is the methods of the emulator for converting logical addresses into linear and linear to physical for educational purposes.

Research methods are based on computer logic methods, mathematical statistics methods, software development methods.

The result of the work is a software implementation of the emulator system for converting logical addresses into linear and linear addresses into physical ones for educational purposes.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with OS Windows 10/11.

The program was developed in the Python environment.

Keywords: computer engineering, logical address, linear address, physical address

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АЛП– арифметико–логічний пристрій;

АЦП – аналогово–цифровий перетворювач;

ЕОМ – електронна обчислювальна машина;

ЗПР– запит переривання;

ОС – операційна система;

ПДП – теж саме що і DMA;

ПЗП – постійний запам’ятовуючий пристрій;

ПЗП – постійний запом’ятовуючий пристрій;

ПК – персональний комп’ютер;

ППЗП – перепрограмувальний постійний запом’ятовуючий пристрій;

ПЧ – перетворювач частоти;

РА – регістр адреси;

РК – регістр команд;

РП – регістр пріоритетів;

ТД – технічна документація;

ТЗ – технічна задача;

ЦАП – цифро–аналоговий перетворювач;

ВСТУП

Актуальність теми. Студентам усіх галузей обчислювальної техніки потрібні знання обчислювальних пристроїв, включаючи реалізацію програмного забезпечення на машинному рівні. Кілька курсів у навчальних програмах з інформатики розглядають ці низькорівневі деталі, такі як архітектура комп'ютера та мови асемблера. Для таких курсів є переваги у вивченні реальних архітектур замість спрощених прикладів. Однак реальні архітектури та набори інструкцій вносять складність, що ускладнює їх опанування в рамках одного семестру. Методи візуалізації можуть допомогти полегшити це навантаження, на жаль, існуючі інструменти часто важко використовувати, а отже, важко впроваджувати в курсі, де час вже обмежений. Щоб вирішити цю проблему, у дані роботі розробляється програмне забезпечення емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей. Дане програмне забезпечення графічно ілюструє ключові відмінності між знайомими мовами високого рівня та незнайомими мовами низького рівня, а також ілюструє, як знайомі програми високого рівня поводяться на реальних архітектурах. Ключем до цього інструменту є те, що ми використовуємо простий веб-інтерфейс, який не потребує налаштування, що полегшує перешкоди для впровадження курсу. Ми також включаємо кілька функцій, які ще більше підвищують його корисність в умовах класу. Ці функції включають графічні зв'язки між кодом високого рівня та машинним кодом, чітко проілюстровані покрокові переходи станів машини, кольорове кодування для зрозумілості поведінки інструкцій та ілюстрацію вказівників.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

– Дослідження системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

– Програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Об'єктом дослідження є процес емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Предметом дослідження є методи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Методи дослідження базуються на методах комп'ютерної логіки, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

– Розроблено вітчизняний продукт емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2025 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Одним з важливих аспектів початкової стадії навчання студента правильно і цілком правильно зрозуміти будову персонального комп'ютера, без цих знань можливо неправильне тлумачення студентом роботи комп'ютера в цілому, що приводить до низького рівня знань майбутнього фахівця і неможливості написати повноцінні програмні продукти.

Існує трирівнева система навчання будови комп'ютера, а саме:

– користувальницький рівень. На даному рівні студент взнає загальні поняття і принципи роботи ЕОМ і термінологію застосовувану в даній спеціалізованій області.

– рівень операційної системи. Тут розглядаються питання заглибленого розгляду компонентів системи і їхня взаємодія між собою у певній операційній системі з використанням високорівневих мов програмування і усіляких віртуальних пристроїв.

– мікроархітектурний рівень. На останньому рівні студент познає, як відбувається взаємодія портів введення-виведення, також роботу електронної складової ЕОМ і елементи використовувані в операційних системах. Також студент учиться зневажати обмеженнями операційних систем.

Усі вище описані рівні в досить повному обсязі повинні бути описані у розробленому програмному продукту.

Це розширює можливості застосування програми, не тільки для навчання студентів вузкоспеціалізованих спеціальностей, але і для навчання школярів, студентив професійно технічних училищ, а також інших навчальних закладів.

Сучасній людині важко представити своє життя без *електронно-обчислювальних машин* (ЕОМ).

У цей час будь-якому бажаючому під силу зібрати в себе на робочому столі повноцінний обчислювальний центр, ступінь функціональності якого може бути обмежена тільки фантазією й фінансовими можливостями його власника.

1.2 Область застосування

Область застосування даної продукції величезна. Вивчення механізму перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей. Що дасть змогу у подальшому зрозуміти архітектуру ПК в цілому та особливості роботи підсистем.

Область застосування системи що розробляється – необхідна студентам вищих навчальних закладів, школярам та іншим особам які вивчають основу будови ПК, що дасть змогу у подальшому стати системним програмістом.

Програмний продукт який розроблюється повинен задовольняти наступним вимогам:

– мати інтуїтивно зрозумілий і зручний інтерфейс;

– відображати в головному вікні програми структурну схему;

– мати добре розгалужену мережу повідомлень взаємодії модулів проекту;

– мати модулі які описують роботу заданих блоків комп'ютера, що задовольняють таким вимогам:

а) опис на користувальному рівні основних можливостей відповідного блоку комп'ютера;

б) опис усіх регістрів відповідного блоку комп'ютера;

в) мати список використовуваної літератури з авторськими правами на той чи інший документ;

– емулювати процес перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей;

– мати модуль переходу до всіх створених блоків із вказівками і рекомендаціями переходу;

– мати форму документації та глоссарію, а також розділ з посиланнями на сайти.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

1. Canvas: LMS для легкого навчання та продуктивності

Instructure, технологічна компанія, що стоїть за хмарним освітнім проривом Canvas, розробила його спеціально як програмне забезпечення для студентив від дитячого садка до вищої освіти. За роки роботи Canvas стала однією з найкращих LMS для шкіл. Canvas можуть використовувати школи будь-якого розміру та типу, незалежно від того, чи це невеликий клас, чи великий університет із гібридною або повністю дистанційною системою навчання.

Canvas – це високофункціональне, гнучке та надійне навчальне рішення, що робить його системою управління навчанням (LMS) для легкого навчання та продуктивності. Найкращі функції та кваліфікація є причиною того, що LMS-рішення охоплює 21% академій та 27% студентив. Воно також пропонує Bridge, його корпоративний аналог з відкритим кодом.

Основні характеристики:

- Спільний робочий простір.
- Інтегровані навчальні інструменти.
- Функція копіювання та вставки з веб-браузера.
- Персоналізований контент.
- Відкритий API.
- Спільні ресурси.
- Матеріали для веб-конференцій.
- Центр програм Canvas.
- Інструмент звітності про графічну аналітику.

- Інтегрована медіа-репортаж.
- Запис або завантаження аудіо та відео.
- Веб-браузер.
- LTI-інтеграції.
- Персоналізовані профілі користувачів.
- Аудіо- та відеоповідомлення.
- Інтеграції зовнішніх сервісів.
- Підтримка RSS.
- Сповіщення про курс.
- Відкрита безпека.

Ціна:

Canvas пропонує безкоштовний пробний період і план на основі цінової пропозиції. Відвідайте їхній веб-сайт для отримання додаткової інформації.

2. Blackboard Learn: LMS для оцінювання та звітності про контент

Академічні кола почали використовувати Blackboard Learn наприкінці 1990-х років. Відтоді він перейшов на обслуговування шкіл K-12, вищих навчальних закладів, урядових та військових ініціатив, а також бізнесу. Його можна використовувати через SaaS-впровадження, самостійно розміщення або керований хостинг. З того часу його також було оновлено для належної роботи з мобільними пристроями.

Blackboard Learn було створено для того, щоб забезпечити студентам ефективні онлайн-курси та навчання з меншою кількістю або взагалі без особистих зустрічей. Це програмне забезпечення, розміщене в Інтернеті, демонструє систему управління курсами, персоналізовану відкриту архітектуру та розширений дизайн, що дозволяє користувачам інтегрувати його з інформаційними системами студентів та процедурами автентифікації. Ви можете інтегрувати його з низкою бізнес-систем та програм, таких як BrainHoney, Epsilen та Moodle Learning Management System, серед інших.

Blackboard Learn широко відома як LMS для оцінювання та звітності про контент. На підтвердження цього, одне дослідження повідомляє, що 28% інститутів та 37% студентів використовують її (Fenton, 2018).

Основні характеристики:

- Створення контенту.
- Покращене управління контентом.
- Управління студентами.
- Можливості дистанційного навчання.
- Адміністрування сертифікації.
- Персоналізовані курси.
- Розширене підприємство.
- Умови соціального навчання.
- Портфоліо та розширені хмарні профілі студентів.
- Спливаюче вікно для попереднього перегляду зустрічей та обговорень, безпечного призначення та планування.
- Система інтеграції рядків.
- Управління даними.
- Управління групою.
- Підходи до покращення оцінювання.
- Привід Blackboard та редактор контенту.
- Центр утримання та реєстрація в програмах.
- Прогресивний контент та активна співпраця.
- Інтуїтивно зрозумілий календар.
- Стрічка активності.
- Функції гейміфікації.

Ціна:

- Безкоштовна пробна версія для нових користувачів.
- 2500 доларів США на рік для шкіл.

3. Google Classroom: LMS для створення курсів

Google Classroom належить до Google Apps for Education, помітного сегменту екосистеми додатків цього гіганта програмного забезпечення. Багато хто вважає його найкращою системою управління навчанням (LMS) для шкіл, що співпрацюють з Google Space (Gerwitz, 2020).

Google Classroom поєднує надійні системи управління освітою з видатними комунікаційними рішеннями. Отримана технологія дозволяє академіям безперешкодно спілкуватися зі студентами та батьками. Для багатьох цього достатньо, щоб заслужити репутацію LMS для створення курсів.

Google Classroom можна використовувати лише в освітніх приміщеннях.

Основні характеристики:

- Доступність комп'ютера, планшета та смартфона.
- Можна використовувати Документи Google для виконання завдань, надсилати відео з YouTube та завантажувати файли з Диска Google.
- Можна залишати відгуки через коментарі до файлів.
- Відстежує та призначає віртуальну роботу, пов'язану зі змішаним навчанням.
- Можна створювати онлайн-класи для обміну навчальними матеріалами, які можна завантажити та переглянути.
- Можна створювати онлайн-завдання для відстеження успішності студентив.
- Персоналізовані дозволи під час спільного доступу до екранів та блокування всього.
- Можна покращити онлайн-навчання за допомогою Gmail, Google Docs, Google Slides, Google Drive та Calendar.
- Може передавати оцінки студентів до ШІС.

Ціна:

- Разом з рештою G Suite for Education, Google Classroom є безкоштовним для шкіл. Він надає необмежену кількість користувачів та 100 ГБ сховища.

– Вартість річної підписки на G Suite for Education з додатковими функціями становить 48 доларів.

4. Wisenet: Провідний хмарний освітній та навчальний застосунок

Wisenet має понад 20 років великого досвіду у наданні хмарних додатків постачальникам освітніх та навчальних послуг. Підключений додаток для управління навчанням орієнтований на вищі навчальні центри, постачальників короткострокових курсів, корпоративне навчання тощо. Він дозволяє академічним установам генерувати ефективні результати навчання, посилювати залучення зацікавлених сторін, підвищувати продуктивність та досягати зростання бізнесу.

Особливості:

– Асинхронне навчання.

– Змішане навчання.

– Інтуїтивна система управління навчанням (LMS).

– Сертифікація та ліцензування.

– Реєстрація на заняття.

– Управління класом.

– Заліковий журнал.

– Портал для студентив.

– Мобільне навчання.

– Відповідність SCORM.

Ціна:

– Стартовий план, 195 доларів США на місяць, необмежена кількість користувачів.

– Стандартний план, 330 доларів США на місяць, необмежена кількість користувачів.

– Преміум-план, 495 доларів на місяць, необмежена кількість користувачів.

5. Workday Student: веб-інструмент управління кампусом

Workday Student – це веб-інструмент управління кампусом, який здатний об'єднати весь навчальний заклад в одному простому у використанні та налаштовуваному додатку, що працює через хмару. Він об'єднує набір студентів, вступ, фінансову звітність студентів та управління навчальними програмами. Він також допомагає в налаштуванні додатків, запуску кампаній, покращенні процесів набору та обчисленні статусу зарахування.

Workday Student було розроблено для підтримки академічних установ. Він безперебійно працює з Workday Human Capital Management, Workday Payroll, Workday Financial Management та Workday Grants Management.

Особливості:

- Управління прийомом.
- Інформація для студентів.
- Студентський портал.
- Заліковий журнал.
- Управління навчальними програмами.
- Управління фінансовою допомогою.

Ціна:

Тільки за ціновою пропозицією. Зверніться до постачальника, щоб дізнатися про його цінові пакети.

6. Nearpod: інтерактивний інструмент для уроків зі спільними завданнями

Nearpod надає навчальним закладам можливість запускати інтерактивні уроки зі спільними завданнями, включаючи вікторини, симуляції, віртуальну реальність тощо. Програмне забезпечення для управління класом дозволяє викладачам проводити класні заняття в прямому ефірі та онлайн.

Nearpod має панель активності, яка дозволяє викладачам завантажувати слайди Google, відео, PDF-файли, презентації PowerPoint та багато іншого для проведення інтерактивних уроків. За допомогою цього програмного забезпечення

викладачі можуть надсилати аудіовідповіді, перевіряти матеріали в режимі реального часу на студентському порталі та проводити формувальне оцінювання для вивчення успішності студентив. Воно також дозволяє викладачам запускати уроки на основі ігор та аналізувати роботи студентив, оцінюючи звіти після семінару.

Основні характеристики:

- Розклад занять.
- Інструменти для співпраці.
- Гейміфікація.
- Управління класом.
- Створення курсу.
- Управління навчальними програмами.
- Управління оцінюванням.
- Управління зворотним зв'язком.
- Інформаційний центр.
- Інтеграція з LMS.
- Мобільний додаток.
- Показники ефективності.
- Опитування/голосування.
- Дані в режимі реального часу.
- Трансляція в режимі реального часу.
- Аналітика.
- Самостійне навчання.
- Оцінювання навичок.
- Студентські записи.
- Управління студентами.
- Студентський портал.
- Підтримка відео.

Ціна:

- Доступні безкоштовна пробна та безкоштовна версії.
- Його платний план починається від 120 доларів на рік.

7. ClassDojo: Залучення студентив та батьків

ClassDojo допомагає навчальним закладам залучати студентив та батьків за допомогою персоналізованих повідомлень, зображень та відео. Він дозволяє викладачам створювати та ділитися онлайн-портфоліо, дозволяючи студентам та батькам входити в систему, скануючи QR-код, та робити записи в щоденнику, записувати відео та надсилати роботи за допомогою зображень.

Він надає історії, які допомагають викладачам ділитися фотографіями, посиланнями, файлами чи оголошеннями, а також відео, використовуючи приватні зображення. ClassDojo також пропонує корисні функції, такі як музика в класі, вказівки щодо проектної діяльності, вимірювач шуму, зворотний зв'язок, обмін інформацією про події тощо. Генератор випадкових студентив дозволяє викладачам вибирати волонтерів під час уроків чи заходів.

Особливості:

- Чат/повідомлення.
- Сповіщення/сповіщення.
- Розклад занять.
- Управління курсом.
- Відстеження відвідуваності.
- Відстеження активності.
- Управління активністю.
- Заліковий журнал.
- Моніторинг у режимі реального часу.
- Студентські записи.
- Управління студентами.
- Студентський портал.

Ціна:

- Безкоштовна версія для вчителів.
- Зверніться до постачальника, щоб дізнатися про преміум-тарифні плани для студентів та батьків.

8. SplashLearn

SplashLearn – це інноваційна онлайн-платформа навчання, розроблена для студентів від дошкільного віку до 5-го класу, яка поєднує захопливий ігровий процес з академічною складністю. Маючи понад 45 мільйонів користувачів по всьому світу, вона пропонує унікальний підхід до навчання, роблячи уроки інтерактивними, персоналізованими та цікавими. SplashLearn пропонує комплексну навчальну програму з математики та читання, з різними інструментами для батьків і вчителів, щоб контролювати та підтримувати прогрес дітей.

Основні характеристики для батьків:

- Щоденний навчальний шлях.
- Узгодження навчальної програми.
- Інтерактивні ігри та винагороди.
- Робочі аркуші та живі заняття.
- Багатокористувацькі ігри.

Основні характеристики для вчителів:

- Безкоштовний доступ до понад 11 000 робочих аркушів.
- Google Classroom та розумна інтеграція.
- Відстеження прогресу студентів.
- Безкоштовний доступ до домашніх завдань.
- Плани уроків на замовлення.
- Основні функції програмного забезпечення:.
- Персоналізовані навчальні шляхи.
- Адаптивний алгоритм.
- Детальні звіти про хід виконання.

– Міжплатформна сумісність.

– Заняття з репетитором у реальному часі.

Ціна:

– SplashLearn безкоштовний для вчителів.

– Зверніться до постачальника, щоб дізнатися про преміум-тарифні плани для студентів та батьків після семиденного безкоштовного пробного періоду.

9. Dyknow: Моніторинг пристроїв студентів

Dyknow відстежує iPad, Chromebook, ПК та MacBook студентів, щоб зібрати корисну інформацію про їхнє ставлення до уроків. Це допомагає викладачам бачити використання Інтернету їхніми студентами під час уроків, щоб допомогти їм покращити їхню успішність у класі. Це дає викладачам можливість блокувати певний контент або вебсайти, які вони сприймають як відволікаючі або неприємні.

Dyknow допомагає викладачам готувати тести та оцінювання. Завдяки своїй програмній базі даних вчителі можуть вибирати питання для тесту. База даних також пропонує різноманітні набори питань з різним рівнем складності для кожного предмета. Вона дає викладачам можливість розкривати ключі відповідей або проводити сесії запитань і відповідей після оцінювання, як вони вважають за потрібне. Крім того, вона містить онлайн-уроки з кількох предметів, щоб студенти могли переглянути їх для своїх щорічних оцінювань.

Основні характеристики:

– Розклад занять.

– Інструменти для співпраці.

– Управління комунікаціями.

– Інтуїтивна система управління навчанням (LMS).

– Тести/оцінювання.

– Студентські записи.

– Підтримка аудіо/зображень/відео.

– Управління навчанням.

- Оцінювання навичок.
- Управління оцінюванням.
- Управління дисципліною.
- Управління відповідностю вимогам.
- Управління записами.
- Переглянути монітори студентив.
- Блокування програм/вебсайтів.
- Моніторинг використання Інтернету.
- Моніторинг та звітність у режимі реального часу.
- Імпорт/експорт даних.
- API.
- Контроль доступу.

Ціна:

Зверніться до Dyknow, щоб дізнатися про їхні тарифні плани.

10. Socrative: Збагачення викладання та навчання

Програмне забезпечення для формувального оцінювання на основі вікторин Socrative пропонує кілька функцій для збагачення навчального середовища. За його допомогою вчителі можуть створювати вікторини, квитки на вихід тощо, щоб збирати та переглядати дані про студентив на льоту, вносити зміни в режимі реального часу та покращувати навчання студентив. Студентам не потрібно створювати облікові записи, щоб мати змогу користуватися Socrative; вчителі можуть просто запросити їх, надавши URL-адресу, щоб надати їм доступ до вікторини, запитання тощо.

Доступ до Socrative можна отримати через Chrome, Windows та Apple. Користувачам не потрібно нічого завантажувати. Він також працює на смартфонах, планшетах та настільних комп’ютерах.

Основні характеристики:

- Моніторинг віртуального комп'ютера.
- Інтерактивні вікторини.

- Адміністрування студентських завдань.
- Збір студентських завдань.
- Звітування про прогрес.
- Бібліотеки спільного вмісту.
- Різноманітність питань.
- Оцінювання в режимі реального часу.
- Автоматизоване оцінювання.
- Гейміфікація.
- Аналітична панель інструментів.
- Звітність.

Ціна:

- Безкоштовно для дітей від дитячого садка до 12 класу (версії Pro).
- 59,99 доларів США щорічно.
- 99,99 доларів США щорічно за вищу освіту.

11. Schoology: LMS для шкіл K-12 та інтеграції зі сторонніми розробниками

Schoology спеціалізується на викладанні курсів, адмініструванні навчальних програм, а також обміні матеріалами та співпраці. Вона пропонує найповнішу платформу для змішаного навчання, надаючи викладачам персоналізовані навчальні ролі та покращуючи загальні результати для студентив. Для багатьох навчальних закладів першокласні функції роблять її найкращою системою управління навчанням (LMS) для шкіл K-12 та інтеграцій зі сторонніми розробниками.

Вже майже 10 років Schoology підтримує всі типи навчання, такі як дистанційні програми та змішане навчання. Він є частиною Powerschool Unified Classroom™ та має 20 мільйонів користувачів у 50 штатах США.

Основні характеристики:

- Уроки та створення контенту.
- Розклад тестів та вікторин.

– Швидке створення завдань за допомогою інструментів розробки контенту та редактора форматованого тексту.

– База даних, куди користувачі можуть завантажувати матеріали курсу та завдання.

– Переглядайте прогрес оцінювання, переглядаючи профілі студентив.

– Кольорове порогове навантаження, яке показує, чи перевантажений студент.

– Успіхи студентив можуть побачити їхні батьки.

– Надійна комунікація та співпраця, а також індивідуальне навчання, дані та аналітика.

– Адміністрування оцінювання та сумісність, а також асинхронне навчання.

– Візуалізуйте складність або простоту уроків за допомогою інструментів HTML та CSS.

– Веб-розміщені та нативні додатки для iOS, Android та Kindle, що дозволяють викладачам та студентам працювати разом.

– Розширені функції, які можна інтегрувати з існуючими платформами.

– Запис, функціональність шкали оцінювання, нова аналітика, звітність та інтеграції зі сторонніми розробниками.

– Автоматизовані системи оцінювання.

– Відповідність, тестування та оцінювання SCORM.

Ціна:

– Безкоштовний пробний період на один місяць.

– 10 доларів на місяць.

12. GoReact: Інструмент для аналізу та покращення навичок студентив

Хмарне програмне забезпечення для оцінювання відео GoReact пропонує динамічні, гнучкі та зрстуденти функції для аналізу розвитку навичок та їх удосконалення. Воно забезпечує зворотний зв'язок, оцінювання та аналіз

успішності студентів. Воно оптимізує та гуманізує навчання на основі навичок за допомогою перевіреного поєднання відео та зворотного зв'язку.

Навчальні заклади та навчальні центри по всьому світу покладаються на GoReact для управління розширюваним відеоспостереженням, оцінюванням та коучингом. Він доступний за ціною та може використовуватися студентами та викладачами різних дисциплін.

Основні характеристики:

- Готовий контент.
- Різноманітність питань.
- Мультимедійний контент.
- Оцінка на місці.
- Індивідуальні навчальні шляхи.
- Гнучке навчання.
- Сумісність з мобільними пристроями.
- Інтеграції.
- Аналітика.

Ціна:

- 60 доларів США за робоче місце на рік для ліцензій на відділи та сайти.
- 149,95 доларів США за п'ятирічну студентську ліцензію.

13. Kahoot!: Інструмент для взаємодії зі студентами

Система управління навчанням Kahoot! базується на покращенні взаємодії викладачів зі студентами шляхом проведення ігор та віртуальних сесій, спрощення процесів віртуального навчання та відстеження успішності студентів за допомогою формувального оцінювання. Вона проводить ігри в реальному часі з різними типами питань, такими як «правда» чи «неправда», питання з вибором однієї правильної відповіді тощо. Ви можете переглядати кількість питань або гравців, перевіряти успішність студентів та створювати звіти, щоб визначити складні питання.

Kahoot! дозволяє навчальним закладам запитувати відгуки студентів за допомогою опитувань та змінювати час занять. Він також дозволяє вищим навчальним закладам спілкуватися зі студентами через відеоконференції та вивчати навчальний матеріал для полегшення підготовки до іспитів.

Основні характеристики:

- Асинхронне навчання.
- Змішане навчання.
- Інструменти для співпраці.
- Панель активності.
- Налаштовувані шаблони.
- Налаштовувані питання.
- Гейміфікація.
- Бібліотека контенту.
- Бібліотека питань.
- Створення тестів/вікторин.
- Управління навчанням.
- Управління комунікаціями.
- Опитування/голосування.
- Показники ефективності.
- Звіти про прогрес та відстеження.
- Спеціальна звітність.
- Інструменти аналізу даних.
- Аналітика, дані та звітність у режимі реального часу.
- Налаштовувані звіти.
- Управління сертифікацією.
- Багатомовний.
- Інтеграція PowerPoint.
- Трансляція презентації.
- Віддалений доступ/керування.

– Інтеграції зі сторонніми розробниками.

Ціна:

– Безкоштовно для базової версії.

– 3 долари США за вчителя на місяць для Pro.

– 6 доларів США за вчителя на місяць за преміум-клас.

14. Embrace: Програмне забезпечення для оптимізації процесу IEP

Embrace – це професійна та розширена платформа, створена для оптимізації процесів IEP та мінімізації розчарування персоналу. Вона пропонує, серед іншого, контрольні показники рівня навчання на рівні штату та основних навчальних програм, моніторинг та сповіщення про часові рамки, форму відстеження інформації та журнали даних батьківських контрактів.

Основні характеристики:

– Створення індивідуальної навчальної програми (ІП).

– Бібліотека цілей та завдань.

– Управління даними студентів.

– Спільний доступ до файлів.

– Відстеження зустрічей.

– Функції звітності.

– Відповідність нормативним вимогам.

Ціна:

Зверніться до Embrace, щоб дізнатися деталі щодо цін.

15. Програмне забезпечення Frontline Special Education Management: розроблене для студентив з особливими потребами

Програмне забезпечення Frontline Special Education Management може задовольнити будь-які проривні вимоги та терміни виконання у сфері спеціальної освіти. Воно покращує якість та виконання індивідуальної освітньої програми (IEP). Воно також своєчасно генерує звіти про успішність студентив. Крім того, воно безпечно співпрацює з батьками, сторонніми постачальниками послуг та системою загальної освіти.

Основні характеристики:

- Управління даними студентів.
- Бібліотека цілей та завдань.
- Відстеження зустрічей.
- Спільний доступ до файлів.
- Налаштовані робочі процеси.
- Функції звітності.
- Створення індивідуальної навчальної програми (ІП).
- Інтеграції/API.
- Відповідність нормативним вимогам.

Ціна:

Зверніться до Frontline Special Education Management Software, щоб дізнатися про їхні цінові пакети.

16. SEIS: Ваш централізований віртуальний доступ до IEP, записів спеціальної освіти тощо

SEIS забезпечує централізований віртуальний доступ до IEP (індивідуальних освітніх планів), записів про спеціальну освіту, звітності CALPADS та моніторингу послуг. Платформа може похвалитися бібліотекою посилань та коментарів, якщо потрібно більше робіт, окрім самого IEP. Нещодавно вона запустила нову функцію електронного підпису.

Доступ до SEIS можна отримати з будь-якого пристрою, якщо він підключений до Інтернету. Крім того, форми IEP вже заповнені необхідною інформацією з облікового запису учня.

Основні характеристики:

- Інформація для студентів.
- Звітність.
- Записи.
- Відстеження історії обслуговування.

Ціна:

Зверніться до SEIS, щоб отримати детальну інформацію про ціни.

17. Quizizz: Програмне забезпечення для гейміфікації

Quizizz – це навчальне рішення, яке пропонує захопливий та захопливий контент за допомогою гейміфікації. Завдяки йому студенти можуть брати участь у живому та асинхронному навчанні за допомогою пристрою, як дистанційно, так і особисто. Водночас тренери та вчителі отримують миттєві дані та зворотний зв'язок.

Основні характеристики:

- Інтерактивний контент.
- Гейміфікація.

Ціна:

Quizizz пропонує різні тарифні плани з оплатою щомісяця або щороку. Зверніться до постачальника для отримання додаткової інформації.

18. Quizlet: Система управління навчанням (LMS) для вивчення мов

Quizlet – це проста у використанні та інтуїтивно зрозуміла система управління навчанням (LMS) для вивчення мов. Вона здебільшого використовується студентами та викладачами мов для ведення нотаток, запам'ятовування та інших навчальних цілей. Ефективна простота вікторин та карток пояснює, чому Quizlet використовує половина всіх студентив середніх шкіл у США (Fenton, 2018).

Основні характеристики:

- Найкраще підходить для покращення словникового запасу та запам'ятовування.
- Можна створювати, ділитися та шукати набори досліджень інших користувачів.
- Надає різноманітні режими навчання.
- Відстежує та призначає дистанційну роботу, пов'язану зі змішаним навчанням.

Ціна:

- 35,88 доларів США на рік.
- Quizlet Plus з додатковими функціями коштує 47,88 доларів на рік.

19. Pocket Study: Програмне забезпечення для проведення дистанційних занять та створення цифрових навчальних інститутів

Pocket Study – це інструмент дистанційного навчання, який допомагає академічним закладам проводити дистанційні заняття та створювати цифрові навчальні інститути. Студенти можуть прослуховувати певні уроки та фільтрувати теми на основі тегів для легкого доступу. Одночасно викладачі можуть поширювати навчальні ресурси серед студентив, створювати канали з навчальними відео та записувати певні зображення чи теми під час лекцій. Інститути також можуть контролювати поточні заняття та взаємодіяти зі студентами під час живих сесій.

Pocket Study має інтуїтивно зрозумілий відеоплеєр YouTube та інтегрується з Vimeo. Він пропонує мобільні додатки для Android та iOS, а також підтримку електронною поштою та телефоном.

Основні характеристики:

- Асинхронне навчання.
- Синхронне навчання.
- Змішане навчання.
- Розклад занять.
- Відстеження відвідуваності.
- Створення курсу.
- Управління курсом.
- Управління навчанням.
- Самостійне навчання.
- Відстеження курсу.
- Двосторонній аудіо та відеозв'язок.
- Відеоконференції.

- Підтримка відео.
- Контроль доступу.
- Інтеграції зі сторонніми розробниками.
- Звітність та статистика.

Ціна:

Цінові плани Pocket Study починаються від 2 доларів на рік.

20. Brainscape: Гнучке програмне забезпечення для мобільної та веб-освіти

Використовуючи картки, Brainscape допомагає студентам та співробітникам навчатися швидше. Він повністю розроблений для забезпечення доступності, щоб кожен міг легко створювати, знаходити та ділитися картками, а також вивчати їх, використовуючи найсучасніші методи когнітивної науки. Його торгова площадка пропонує мільйони колекцій карток майже з кожного предмета у світі.

Ціна:

Зверніться до Brainscape, щоб дізнатися більше про плани передплати.

21. Альта: Персоналізація навчальних процесів

Alta – це гнучкий навчальний інструмент, який повністю підтримує вищі навчальні заклади в управлінні індивідуальними навчальними процесами. Завдяки платформі студенти просто обирають підхід відповідно до своїх потреб. Вони можуть отримати до нього доступ з будь-якого зручного для них місця.

Основні характеристики:

- Простий у використанні та креативний інтерфейс.
- Широкий вибір курсів.
- Мобільний.
- Оцінювання успішності студентів.

Ціна:

Зверніться до компанії Alta, щоб отримати додаткову інформацію про тарифні плани.

22. Mendeley: Найкраще програмне забезпечення для дослідників

Mendeley було розроблено та створено спеціально для того, щоб допомогти дослідникам в організації та обміні науковими роботами, а також у пошуку дослідницьких даних. Потужне програмне забезпечення для керування посиланнями дозволяє їм зберігати, записувати та цитувати посилання, переглядати статті та імпортувати документи безпосередньо зі свого робочого столу. Воно пропонує бібліотеку контенту, можливості для співпраці, обмін файлами, зберігання документів, панель активності тощо. Воно дозволяє користувачам налаштовувати робочі процеси та створювати посилання, цитати та бібліографії в різних стилях журналів, подібно до функцій спеціалізованого програмного забезпечення для написання дисертацій.

Mendeley надає модуль блокнота, який дозволяє видавцям упорядковувати виділені нотатки з різних PDF-файлів та зберігати їх в єдиному місці.

Основні характеристики:

- Панель активності.
- Сповіщення/сповіщення.
- Інструменти для співпраці.
- Зберігання документів.
- Спільний доступ до файлів.
- Управління нотатками.
- Імпорт/експорт даних.
- Підтримка кількох пристроїв.
- Звітність/аналітика.

Ціна:

Зверніться до Mendeley, щоб дізнатися про їхні цінові пакети.

23. RedCube Papers: програмне забезпечення для динамічного керування та пошуку посилань

RedCube Papers – це відзначене нагородами довідкове програмне забезпечення, яке обслуговує дослідників, видавців, освітні та комерційні фірми.

Воно демонструє динамічний набір рішень для управління посиланнями та пошуку, допомагаючи дослідникам шукати, переглядати, упорядковувати, читати, ділитися та цитувати дослідницьку літературу.

Основні характеристики:

- Інтуїтивно зрозумілі пошукові системи.
- Розширені фільтри пошуку.
- Індивідуальні рекомендації.
- Веб-імпорт.
- Підтримка інституційного проксі.
- Завантаження PDF-файлів одним клацанням миші.
- Легкий імпорт інструментів.
- Стрічки пов'язаних статей.
- Автоматичне зіставлення метаданих статті.
- Повний пошук у бібліотеці текстів.
- Розширене сортування та фільтрація.
- Рстуденти та розумні колекції.
- Теги ключових слів, мітки та рейтинги статей.
- Покращений перегляд PDF-файлів.
- Вбудовані посилання з гіперпосиланнями, браузери зображень високої роздільної здатності та автоматично завантажені додатки.
- Покращені показники статей.
- Інструменти для вбудованих та стікерів, малювання та виділення тексту.
- Функція перетворення тексту на мовлення.
- До п'яти приватних спільних колекцій.
- Працюйте з до 25 користувачами кожної колекції.
- Поділитися посиланнями.
- Група обговорення статей.
- Розміщуйте посилання з особистих/спільних бібліотек або скористайтеся інтуїтивно зрозумілою пошуковою системою.

- Понад 8000 стилів цитування.
- Швидке копіювання цитати в bibtex, ris.
- Список експортних посилань для сторонніх платформ.
- Можна використовувати у Word 2016+.
- Необмежене хмарне сховище для особистої бібліотеки.
- Синхронізація всіх списків бібліотек, нотаток тощо на різних пристроях.

Ціна:

- Від 3 доларів США на місяць за індивідуальні ліцензії (академічні).
- 20 доларів США за робоче місце на місяць для корпоративної версії.

24. Wizeprep: Доступна підтримка студентів у хмарі

Wizeprep надає навчальну підтримку студентам коледжів та університетів Північної Америки вже понад десять років. З 2019 року ця підтримка надається на 100% через онлайн-платформу, яка пропонує додаткові навчальні матеріали для кампусів та курсів.

Wizeprep пропонує три основні продукти: 1. універсальну підписку зі спеціалізованими навчальними ресурсами для курсів STEM, бізнесу та письма; 2. сесії підготовки до іспитів у прямому ефірі (групові заняття), що пропонуються через Zoom у вибраних місцях; та 3. єдиний курс підготовки до тесту MCAT, Elite 515, який гарантує отримання 515 балів на MCAT або повернення грошей!

Основні характеристики:

- Асинхронне навчання 24/7.
- Міні-відеоуроки.
- Практичні завдання, схожі на екзаменаційні.
- Повні рішення та пояснення практичних задач.
- Конспекти курсів, доступні для завантаження.
- Бібліотека ресурсів, що стосуються коду курсу, з можливістю пошуку.
- Курси та брошури MCAT.
- Персоналізована панель інструментів для студентів.
- Функція питань і відповідей зі справжніми інструкторами.

Ціна:

- Безкоштовна пробна версія 10 безкоштовних заходів.
- Щомісячна підписка $29.95.
- MCAT Elite 515 $3,999.
- Підготовка до іспитів у прямому ефірі за вибором.

25. Text Blaze

Text Blaze – це програмне забезпечення для викладачів, які хочуть позбутися повторюваного друку, щоб заощадити час на виконанні поширених завдань, таких як написання електронних листів, оцінювання та надання відгуків про завдання студентив. Використовуючи комбінації клавіш Text Blaze, ви можете створювати шаблони, які можна використовувати будь-де в Інтернеті (на будь-якому сайті та в найчастіше використовуваних програмах).

Незалежно від того, чи хочете ви автоматизувати звичайні відповіді на електронні листи, оцінювання чи повторювані відгуки про завдання, Text Blaze допоможе вам. Наприклад, ви можете ввести «/duedate», і воно швидко заміниться на «Вітаю, термін здачі вашого наступного завдання – неділя о 23:59», що заощадить ваш час під час відповіді на електронні листи студентів.

За допомогою Text Blaze ви можете спілкуватися та надавати високоякісний зворотний зв'язок за лічені хвилини. Text Blaze заощаджує ваші години повторюваного друку, покращує результати навчання та зменшує вигорання.

Основні характеристики:

- Text Blaze безкоштовний назавжди. Ви можете отримати цінність від Text Blaze, не сплачуючи жодного долара.
- Форми (заповнювачі) допомагають додавати контент, такий як імена, у режимі реального часу.
- Команди дати та часу допомагають динамічно вставляти терміни виконання у шаблони.

– Логіка та формули дозволяють зробити ваші шаблони потужними за допомогою умовного вмісту та обчислень.

– Text Blaze доступний через Chrome та Windows.

Ціна:

Text Blaze безкоштовний назавжди. Вартість тарифного плану Pro становить $2,99 на місяць, а річна плата – приблизно $36.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – динамічна інтерпретована об’єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Офіційний сайт мови програмування Python https://www.python.org/. Python – багатоцільова мова програмування, яка дозволяє писати код, що добре читається. Відносний лаконізм мови Python дозволяє створити програму, яка буде набагато коротше свого аналога, написаного на іншій мові. Python – багатоплатформова мова програмування. Це означає, що програми на Python можна запускати в різних операційних системах без будь-яких змін.

Ще однією перевагою Python є його стандартна бібліотека, яка встановлюється разом з Python і містить готові інструменти для роботи з операційною системою, веб-сторінками, базами даних, різними форматами даних, для побудови графічного інтерфейсу програм тощо. Програми, написані на мові програмування Python, можуть бути як невеликими скриптами, так і складними системами. Python абсолютно безкоштовний.

Швидкість виконання коду Python

Один з можливих недоліків Python – швидкість виконання коду. Python не є компільованою мовою. Код на Python спочатку компілюється у внутрішній байт-код, який потім виконується інтерпретатором Python. У більшості випадків

при використанні Python виходять програми повільніші в порівнянні з такими мовами, як C.

Втім, сучасні комп'ютери мають таку обчислювальну потужність, що для більшості застосунків швидкість розробки важливіша швидкості виконання, а програми на Python зазвичай пишуться набагато швидше.

Окрім того, Python легко розширюється модулями, написаними на C або C++. Такі модулі можуть використовуватися для виконання частин програми, що створюють інтенсивне навантаження на процесор.

Використання Python

Python використовується для різних цілей: для створення ігор і веб-застосунків, розробки внутрішніх інструментів для різноманітних проектів. Мова також широко застосовується в науковій області для досліджень і розв'язування прикладних завдань.

Застосування мови програмування Python:

1. BitTorrent – протокол для обміну даними.

2. Ubuntu Software Center – вільне програмне забезпечення для пошуку, установки і видалення пакунків в системі Ubuntu Linux.

3. Blender – програма для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, вимальовування, пост-обробки відео, а також створення відеоігор.

4. GIMP – растровий графічний редактор, із підтримкою векторної графіки.

5. World of Tanks.

6. Вільна енциклопедія Вікіпедія.

7. Пошукова система Google.

8. DropBox – файловий хостинг, що включає персональне хмарне сховище, синхронізацію файлів і програму-клієнт.

9. YouTube – популярне відеосховище.

Версії Python

Мови програмування з часом змінюються – розробники додають в них нові можливості, а також виправляють помилки. Так з'являються різні версії мови. Наприклад, код написаний на Python 2 у більшості випадків не буде працювати у версії Python 3 без внесення додаткових змін.

Процесор є найважливішим компонентом в комп'ютері. Одна з основних функцій процесора – це обробка даних згідно комп'ютерної програми, яка є списком інструкцій, шляхом виконання арифметичних і логічних операцій над фрагментами даних.

Кожна інструкція в програмі – це команда, яка «повідомляє» процесору, яку операцію він повинен виконати. Процесор комп'ютера може розуміти лише ті інструкції, які написані на машинній мові. Машинна мова – це штучна мова, створена для передачі команд комп'ютеру. За допомогою машинної мови створюються ефективні програми, оскільки розробник отримує доступ до всіх можливостей процесора. Машинна мова – мова низького рівня.

Інструкція машинної мови існує для кожної операції, яку процесор здатний виконати – є інструкція для додавання чисел, є інструкція для віднімання чисел і т.д. Увесь набір інструкцій, який центральний процесор може виконати, відомий як набір інструкцій процесора.

Наприклад, у вас є певна програма, яка зберігається на диску вашого комп'ютера. Для виконання програми, ви здійснюєте подвійний клік на значку програми. Це змушує програму копіюватися з диска в оперативну пам'ять, після чого процесор комп'ютера виконує копію програми, яка знаходиться в оперативній пам'яті.

Коли процесор виконує інструкції програми, він бере участь у процесі, який є відомим як цикл fetch – decode – execute (отримати – декодувати – виконати). Цей цикл виконується для кожної інструкції у програмі і складається з трьох кроків:

Отримати

Програма – це послідовність інструкцій на машинній мові. Першим кроком циклу є завантаження (отримання) наступної інструкції з пам'яті в процесор.

Декодувати

Інструкція машинної мови – це двійкове число, яке представляє команду, що повідомляє процесору виконати певну операцію. На цьому кроці процесор декодує інструкцію, яку було «витягнуто» з пам'яті, для визначення того, яка операція повинна виконуватись.

Виконати

Останній крок циклу – виконати операцію.

Хоча процесор комп'ютера розуміє тільки машинну мову, людині непрактично писати програми на машинній мові. Така програма може мати тисячі або навіть мільйони бінарних інструкцій, і написання такої програми буде дуже обтяжливим процесом.

З цієї причини була створена мова асемблера як альтернатива машинній мові. Замість використання двійкових чисел для написання інструкцій, мова асемблера використовує короткі слова, відомі як мнемокоди.

Незважаючи на те, що мова асемблера не вимагає двійкових інструкцій, як у випадку машинної мови, проте вона вимагає високих знань про процесор. Використовуючи мову асемблера, навіть для найпростішої програми, необхідно написати велику кількість інструкцій.

Мова програмування високого рівня дозволяє створювати складні програми, не знаючи, як працює процесор, і не записуючи великої кількості інструкцій низького рівня. Крім того, більшість мов програмування високого рівня використовують слова, які легко зрозуміти.

Python – одна із популярних сучасних мов програмування високого рівня. Python – інтерпретована мова програмування. Python – це високорівнева інтерпретована мова програмування, на відміну від C++, яка є прикладом

компільованої мови програмування. Назва Python відноситься як до мови програмування, так і до інтерпретатора – комп'ютерної програми, яка зчитує початковий код (написаний на Python) і виконує інструкції (команди).

Для перекладу мови високого рівня на машинну мову доступні два типи програм:

1. Компілятор.

2. Інтерпретатор.

Завантаження Python

Версії інтерпретатора Python для різних операційних систем доступні для безкоштовного завантаження за адресою https://www.python.org/downloads.

Середовище програмування для Python

Для написання програм використовують текстові редактори або інтегровані середовища розробки, які включають в себе різні інструменти для роботи з кодом: засіб для написання коду (текстовий редактор), інтерактивний інтерпретатор, відлагоджувач тощо.

Текстові редактори та інтегровані середовища програмування для Python:

– IDLE – стандартний редактор Python. Встановлюється разом з Python для користувачів Windows, окремим пакунком для користувачів Linux.

– Notepad++ – безкоштовний текстовий редактор початкового коду, який підтримує велику кількість мов, в тому числі і Python. Лише для користувачів Windows.

– Visual Studio Code – це легкий, але потужний редактор початкового коду, який розповсюджується безкоштовно і доступний у версіях для платформ Linux, Windows і macOS.

– PyScripter – інтегроване середовище розробки для мови програмування Python. Для користувачів Windows. Поширюється безкоштовно.

– Wing IDE 101 – вільне інтегроване середовище для Python, розроблене для навчання програмістів-початківців. Для користувачів Linux, Windows і macOS. Поширюється безкоштовно.

– Geany – вільний текстовий редактор з базовими елементами інтегрованого середовища розробки, доступний для операційних систем Linux, Windows і macOS.

– PyCharm – інтегроване середовище розробки для мови програмування Python. PyCharm є власницьким програмним забезпеченням. Наявна безкоштовна версія Community з усіченим набором можливостей. Для користувачів Linux, Windows і macOS.

– Thonny – IDE для вивчення програмування мовою Python. Для користувачів Linux, Windows і macOS.

– Mu – редактор коду Python для програмістів-початківців. Для користувачів Linux, Windows і macOS.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на

екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Проблема підготовки висококваліфікованих фахівців дуже гостро постала у середині двохтисячних років минулого сторіччя, коли на ринку вакансій стали з'являтися усе більше і більше оголошень з потребою у кваліфікованих програмістів, саме фахівців, а не досвідчених користувачів.

На сьогоднішній день, ринок висококваліфікованих фахівців стабілізувався, але все одно кадрові агентства шукають по всій країні вузькоспеціалізованих комп'ютерних фахівців за замовленням приватних підприємців і комп'ютерних фірм.

В процесі навчання у вищих навчальних закладах виникає гостра потреба в добре організованих і детально розроблених навчальних системах.

У даній магістерській дипломній роботі, зроблена спроба реалізувати навчальну систему емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Коли студент починає вивчати внутрішню будову персонального комп'ютера, він в першу чергу стикається з перетворенням логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

У рамках архітектури ПК існує й розвивається ряд мікроархитектур. Раніше було показано, що поняття архітектури комп'ютера ієрархично й що існують загальні й індивідуальні властивості архітектури. У контексті обговорення архітектури процесорів Intel має сенс також позиціонувати різні їхні архітектурні властивості й принципи як загальні й індивідуальні.

До індивідуальних архітектурних властивостей і принципів можна віднести існуючі в рамках різних мікроархітектур. Що стосується загальних архітектурних властивостей і принципів IA-32, то до них відносяться ті, які мають

місце для всіх процесорів Intel або, принаймні, існують поза рамками конкретної мікроархітектури для великої кількості моделей процесорів. В зв’язку з тим, що процесор в основному визначає логіку роботи комп'ютера, то й назви більшості загальних архітектурних властивостей і принципів IA-32 збігаються з назвами аналогічних властивостей і принципів комп'ютера: номенклатура програмно-доступних регістрів; організація й способи адресації пам'яті; номенклатура режимів роботи процесорів; організація й розрядність зовнішніх інтерфейсів ЕОМ; способи подання й формати даних; набір і формати машинних команд ЕОМ; порядок обробки переривань. Практично всі ці загальні архітектурні властивості й принципи становлять програмну модель процесора, що буде розглянута надалі.

Варіанти мікроархітектури процесорів Intel

Поняття мікроархітектури вперше було визначено Intel для процесорів сімейства Pentium Pro. Її введення пояснювалося необхідністю правильного позиціонування нових процесорів серед існуючих. Зовнішня програмна модель (логічна) 32-розрядних процесорів змінювалася тільки убік розвитку, у той час як їх виконавча (фізична) частина могла бути зроблена різною. Поняття мікроархітектури орієнтоване на опис особливостей виконавчої частини процесорів, тобто того, якими способами і якими засобами процесор виконує обробку машинного коду (рис. 3.1). На сьогоднішній день у рамках IA-32 існує дві мікроархітектури процесорів Intel: P6 і NetBurst

Мікроархітектура P6. Ця мікроархітектура є триходовою (three-way) суперскалярною конвеєрною архітектурою. Термін «триходова» означає підтримку технологій паралельного обчислення, що дозволяють процесору одночасно (за один такт) обробляти до трьох інструкцій. Проблема оптимальної обробки потоку машинних команд є ключовою при розробці будь-якого процесора. Процесори Intel відносяться до групи CISC-процесорів, у яких для виконання однієї команди може вимагати від одиниць до декількох десятків процесорних тактів.

Рисунок 3.1 – Подання комп'ютера у вигляді рівнів

При такій обробці команд збільшення продуктивності може бути досягнуто тільки підвищенням частоти генерації машинних тактів. Просте збільшення частоти роботи процесора не має сенсу, тому що є фізично обумовлена верхня границя, до якої її можна піднімати. У ході виконання команди є й інше вузьке місце – вибірка команди з пам'яті. Це витратна за часом операція.

Часткове рішення проблеми було знайдено у вигляді буфера вибірки, що попереджає. Розвитком цієї й реалізацією інших ідей став конвеєр – спеціальний пристрій, що існує на рівні архітектури виконавчої частини комп'ютера. Завдяки конвеєру виконання команди розбивається на кілька стадій, кожна з яких реалізує деяку елементарну операцію загального процесу обробки команди.

Конвеєр має п'ять щаблів, які відповідають перерахованим далі стадіям обробки машинної команди.

1. Вибірка команди з кеш-пам'яті або з оперативної пам'яті.

2. Декодування команди.

3. Генерація адреси, у ході якої визначаються адреси операндів у пам'яті й виконується вибірка операндів.

4. Виконання операції за допомогою АЛП.

5. Запис результату (місце запису результату залежить від алгоритму роботи конкретної машинної команди).

Чергова команда після її вибірки попадає в блок декодування. У такий спосіб блок вибірки звільняється й може вибрати наступну команду. У результаті на конвеєрі можуть перебувати в різній стадії виконання п'ять команд. Швидкість обчислення в результаті істотно зростає.

У процесорах Pentium конвеєрна архітектура була вдосконалена й одержала назву суперскалярної. На відміну від скалярної архітектури (з одним конвеєром), перші моделі процесорів Pentium мали два конвеєри. В ідеалі такий суперскалярний процесор повинен виконувати дві команди за машинний такт. Але все не так просто.

Реально два конвеєри Pentium не були функціонально рівнозначними. У зв'язку із цим вони навіть мали різні назви – u-конвеєр (головний) і v-конвеєр (другорядний). Головний конвеєр був повнофункціональним і міг виконувати будь-які машинні команди.

Функціональність другорядного конвеєра була обмежена основними цілочисельними командами й однією командою із плаваючою точкою (FXCH).

Процесори мікроархітектури P6 мають іншу структуру конвеєра. Конвеєризація полягає в тому, що весь процес обробки команд розбитий на 12 стадій, які виконуються різними блоками процесора.

Скільки саме команд обробляється процесором, сказати важко. Термін триходова означає лише те, що для виконання із вхідного потоку вибираються до трьох команд. Відома верхня межа – у процесорі в кожний момент часу можуть перебувати до 30 команд у різній стадії виконання. Деталі цього процесу сховані за поняттям динамічне виконання з порушенням вихідного порядку проходження машинних команд (out of order), що означає виконання команд у порядку, обумовленому виконавчим пристроєм процесора, а не вихідною послідовністю команд. В основу технології динамічного виконання покладені три концепції:

– Пророкування правильної адреси переходу. Основне завдання механізму пророкування – виключити перезавантаження конвеєра. Під переходом розуміється запланована алгоритмом зміна послідовного характеру виконання програми. Як показує статистика, типова програма на кожні 6-8 команд містить 1 команду переходу.

Наслідки обробки переходу пророчити нескладно: при наявності конвеєра через кожні 6-8 команд його потрібно очищати й заповнювати заново відповідно до адреси переходу. Всі переваги конвеєризації губляться. Тому в архітектуру Pentium до складу пристрою вибірки/декодування був уведений блок пророкування переходів.

– Динамічний аналіз потоку даних. Аналіз проводиться з метою визначення залежностей команд програми від даних і регістрів процесора з наступною оптимізацією виконання потоку команд. Головний критерій тут – максимально повне завантаження процесора.

– Спекулятивне виконання здатність процесора виконувати машинні команди на ділянках програми з умовними переходами й циклами до того, як ці переходи будуть дозволені алгоритмом програми. Якщо перехід передвіщений правильно, то процесор до цього моменту вже має виконаний код, у противному випадку весь конвеєр потрібно очищати, завантажувати й виконувати код нової галузі програми, що дуже накладно.

Розглянемо порядок функціонування виконавчого пристрою мікроархітектури Рб і реалізацію з його допомогою описаних раніше технологій (рис. 3.2).

Зі схеми видно, що структурно процесор мікроархітектури Р6 складається з декількох підсистем.

– Підсистема пам'яті складається із системної шини, кешу другого рівня L2, пристрою шинного інтерфейсу, кешу першого рівня L1 (інструкцій і даних), пристрою зв'язку з пам'яттю й буфера переупорядкування запитів до пам'яті.

– Пристрій вибірки/декодування складається із пристрою вибірки команд, блоку пророкування переходів, у який входять блоки міток переходу й обчислення адреси наступної інструкції, пристрою декодування, пристрою мікропрограмного керування й таблиці псевдонімів регістрів.

– Буфер команд та пристрій диспетчеризації/виконання містить буфер мікрооперацій, готових до виконання, і п'ять виконавчих пристроїв (два – для виконання цілочисельних операцій, два – для виконання операцій із плаваючою точкою, а також пристрій зв'язку з пам'яттю).

– Блок видалення й відновлення.

Розглянемо підсистему пам'яті. Для безперебійної роботи процесора в мікроархітектурі P6 використовується два рівні кеш-пам'яті. Кеш-пам'ять першого рівня складається з кешей команд і даних розміром по 8 Кбайт, розташованих усередині процесора в безпосередній близькості до його виконавчої частини. Кеш-пам'ять другого рівня є зовнішньою стосовно процесора (але в єдиному конструктиві з ним), має значно більший розмір (256 Кбайт, 512 Кбайт або 1 Мбайт) і з'єднаний з ядром процесора за допомогою 64-розрядної шини.

Поділ кеш-пам'яті на дві частини (для коду й даних) забезпечує безперебійну поставку машинних інструкцій і елементів даних у виконавчий пристрій процесора. Вихідні дані для кеш-пам'яті першого рівня надає кеш-пам'ять другого рівня. Пристрій шинного інтерфейсу звертається до оперативної пам'яті системи через зовнішню системну шину.

Запити від команд на одержання операндів з пам'яті у виконавчому пристрої процесора обслуговуються за допомогою пристрою зв'язку з пам'яттю й буфера переупорядкування запитів до пам'яті. Пристрій вибірки/декодування витягає однин 32-байтний рядок кешу команд (L1) за такт і передає в декодер, що перетворить його в послідовність мікрооперацій. Потік мікрооперацій (поки він ще відповідає послідовності вихідних команд) надходить у буфер команд.

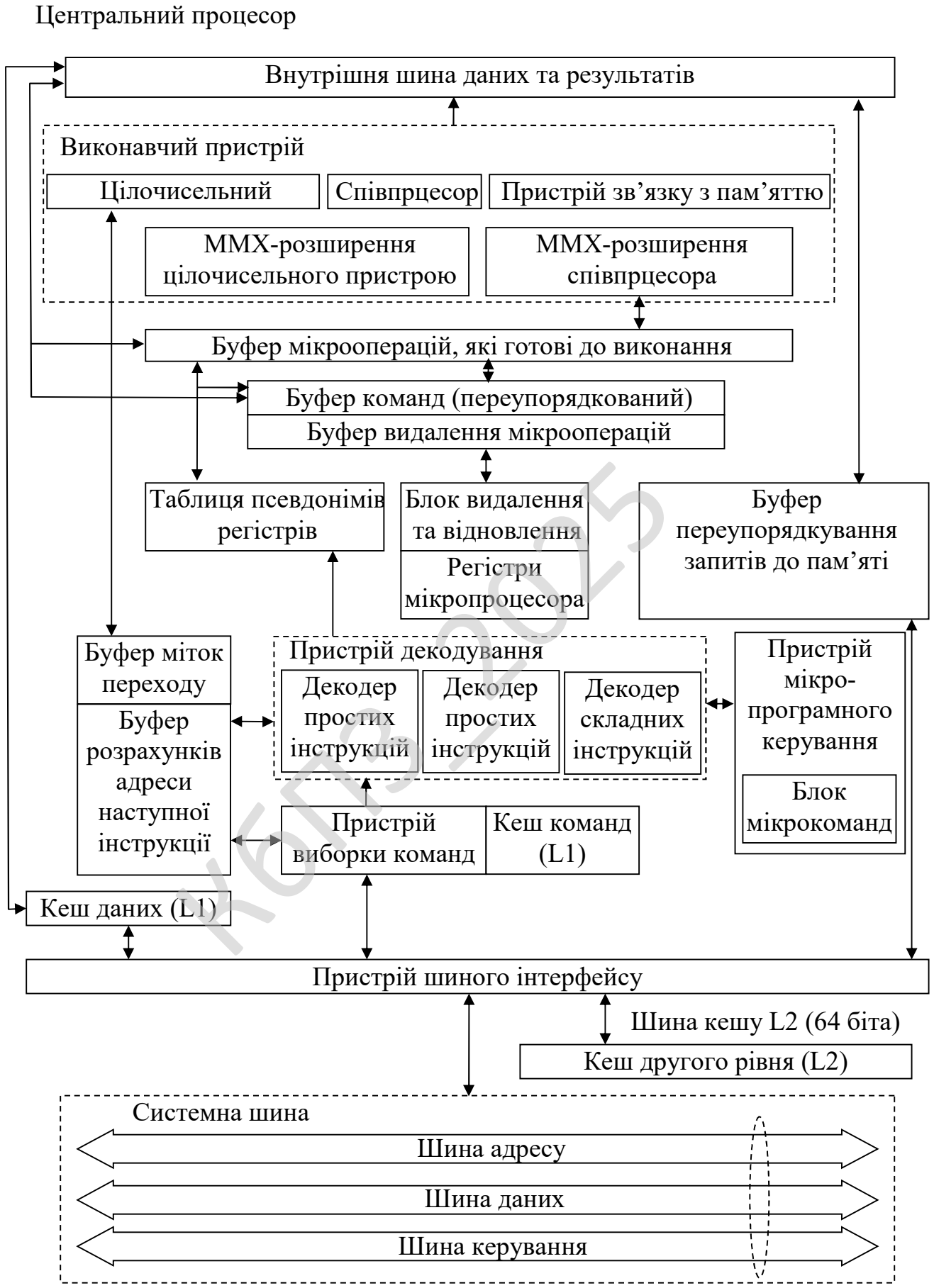

Рисунок 3.2 – Структурна схема процесора сімейства Pentium

Пристрій вибірки команд обчислює покажчик на наступну команду, що підлягає вибірці, на підставі інформації трьох джерел: буфера міток переходу, стану переривання/виключення й повідомлення від виконавчого цілочисельного пристрою про помилку в пророкуванні мітки переходу. Важлива частина цього процесу – пророкування мітки переходу, що виконується по спеціальному алгоритмі. У його основі лежить робота з буфером міток переходу. Пристрій вибірки команд вибирає команди для виконання й поміщає їх у пристрій декодування.

Пристрій декодування складається із трьох паралельно працюючих декодерів (два простих і один складний). Саме ці декодери втілюють у життя поняття виконання з порушенням вихідного порядку проходження команд (out of order) і є тими самими трьома входами (three way) у виконавчий пристрій процесора.

Декодери перетворять команди процесора в мікрооперації. Мікрооперації являють собою примітивні команди, які виконуються п'ятьома виконавчими пристроями процесора, що працюють паралельно. Інформація про послідовність мікрооперацій для реалізації конкретної машинної команди втримується в пристрої мікропрограмного керування.

При цьому можуть виникнути проблеми з таким критичним ресурсом, як регістри. Суть тут у тому, що якщо у двох сусідніх фрагментах програми дані заносилися в однакові регістри, звідки вони, можливо, записувалися в деякі області пам'яті, а після переупорядкування ці фрагменти перемішалися, то як розібратися в тому, які регістри й де використовувалися.

Ця проблема зветься проблемою помилкових взаємозалежностей і вирішується за допомогою механізму перейменування регістрів. Інформація про дійсні імена регістрів процесора і їхні внутрішні імена (номери універсальних регістрів) міститься в таблицю псевдонімів регістрів. На закінчення процесу декодування пристрій керування таблицею псевдонімів регістрів додає до мікрооперацій біти стану й прапори, щоб підготувати їх до неупорядкованого

виконання, після чого посилає мікрооперації, що вийшли, у буфер з команд. Пристрій диспетчеризації/виконання може вибирати мікрооперації із цього буфера в будь-якому порядку.

Пристрій диспетчеризації/виконання планує й виконує неупорядковану послідовність мікрооперацій з буфера з команд. Але він не займається безпосередньою вибіркою мікрооперацій з буфера з команд, тому що в ньому можуть утримуватися й не готові до виконання мікрооперації. Цим займається пристрій, що управляє спеціальним буфером, що умовно назвемо буфером мікрооперацій, готових до виконання.

Результати виконання мікрооперацій вертаються в буфер з команд і зберігаються там поряд з іншими мікрооперaціями доти, поки не будуть вилучені пристроєм видалення й відновлення. Пристрій зв'язку з пам'яттю управляє завантаженням і збереженням даних для мікрооперацій.

Останній блок у цій схемі виконання команд вихідної програми – блок видалення й відновлення, завданням якого є повернення обчислювального процесу в рамки, визначені вихідною послідовністю команд.

Мікроархітектура NetBurst. Реалізована в процесорі Pentium IV, є розвитком ідей мікроархітектури P6. Відзначимо найбільш важливі властивості нової мікроархітектури.

– Швидка виконавча частина процесора. АЛП процесора працює на подвоєній частоті процесора.

– Гіперконвеєрна технологія. Гіперконвеєр Pentium IV складається з 20 щаблів. Ціль збільшення довжини конвеєра – спрощення завдань, реалізованих кожної з його щаблів, і, як наслідок, спрощення відповідної апаратної логіки.

– Поліпшена технологія динамічного виконання завдяки більше глибокій «довільності» у порядку виконання коду й удосконаленій системі пророкування переходів.

– Нова підсистема кешування. Відсутній кеш команд першого рівня. Замість нього введений кеш трас. Трасами називаються послідовності

мікрооперацій, у які були декодовані раніше обрані команди. Структурна схема процесора Pentium IV показана на рисунку 3.3.

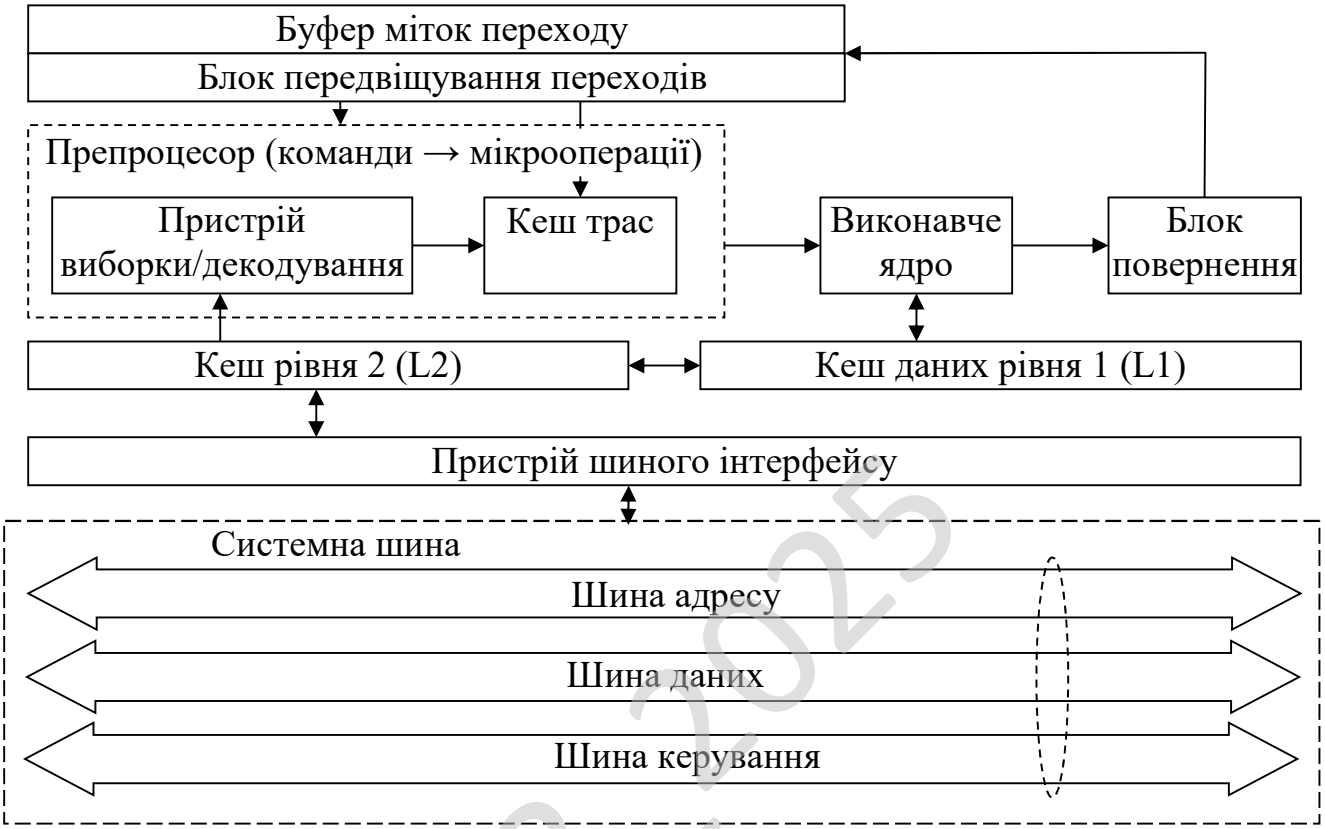

Рисунок 3.3 – Структурна схема процесора Pentium IV

Мікроархітектура NetBurst підтримує ще одну нову технологію – HyperThreading. Дана технологія дозволяє на базі одного фізичного процесора Pentium IV моделювати декілька логічних, кожний з яких має власний архітектурний простір IA-32. Під архітектурним простором IA-32 розуміється сукупність регістрів даних, сегментних регістрів, системних регістрів і регістрів MSR. Кожний логічний процесор має також власний контролер переривань APIC.

Розглянуши структурну схему процесора перейдемо до розгляду структурної схеми ПЕОМ. При розробці структурної схеми відштовхнемося від внутрішньої архітектурної будови персонального комп'ютера. Структурна схема персонального комп'ютера складається:

– Шина PCI (Peripheral Component Interconnect) – шина з'єднання

периферійних компонентів, займає особливе місце в PC-архітектурі, є містком між локальною шиною процесора і шиною введення-виведення ISA/EISA чи MCA;

– Шина ISA – являє собою паралельну шину, створену на базі шини пам'яті і введення/виведення IBM PC/AT;

– контролер клавіатури – взаємодія з клавіатурою в PC AT базується на двох мікропроцесорах Intel 8042. Один мікропроцесор знаходиться в системному блоці, другий – в клавіатурі.

– контролера прямого доступу в пам'ять – прямий доступ до пам'яті (DMA) це метод безпосереднього звертання до пам'яті, минаючи процесор. Процесор відповідає тільки за програмування DMA: настроювання на визначений тип передачі, завдання початкової адреси і розміру масиву даних;

– контролер переривань – в архітектурі PC AT підсистема апаратних преривань складається з двох контролерів 8259A. Вони об'єднані таким чином, що можуть обслужити 15 запитів на переривання;

– послідовний інтерфейс – послідовний інтерфейс RS-232 (КР580ВВ51) обумовлений стандартом EIA для передачі даних в одному напрямку використовує одну сигнальну лінію, по якій інформаційні біти передаються один за одним – послідовно;

– паралельний інтерфейс – паралельний порт використовується для підключення принтера до комп'ютера. Також принтер можливо підключити через асинхронний адаптер і USB порт;

– контролер USB – швидкий, двонаправлений, ізохронний, дешевий, послідовний інтерфейс;

– системний таймер – архітектура PC AT використовує підсистему трьохканального 16-розрядного таймера 8254, як системний таймер. Програмувальний таймер призначений для одержання програмно-керованих тимчасових затримок і генерацій время’задаючих функцій. Таймер дозволяє підвищити ефективність програмування процесів керування і синхронізації

зовнішніх пристроїв, особливо в реальному масштабі часу;

– енергонезалежна пам'ять і годинник реального часу – основне призначення енергонезалежної пам'яті – збереження найбільш важливих параметрів системи при відключенні живлення комп'ютера. Годинник RTC використовуються для установки значення поточного часу в момент ініціалізації системного таймера персонального комп'ютера;

– контролер гнучких дисків (Floppy Disk Controller, FDC) – Керування гнучкими дисками здійснюється мікросхемою (контролером) 8272A;

– контролер твердих дисків з інтерфейсами IDE, EIDE і SCSI;

– робота AGP – спеціалізована надбудова над шиною PCI, що дозволяє створити швидкісний канал обміну даними між графічним акселератором і системною логікою PC.

3.2 Розробка структурної схеми

На рис. 3.4 зображено розроблену структурну схему системи. Розглянемо її роботу детально.

Користувач через головне вікно ПЗ має можливість перейти до довідкової системи з подальшим переглядом документації та проведенням тестування пройденого матеріалу, перейти до модуля налаштування параметрів роботи ПЗ, налаштувати параметри роботи емулятора перетворення.

З блоку емулятора перетворення ми можемо запустити моделювання та провести моделювання крок за кроком з виведення інформації кроку на екран. Емулятор дозволяє наявно подивитись як відбувається перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

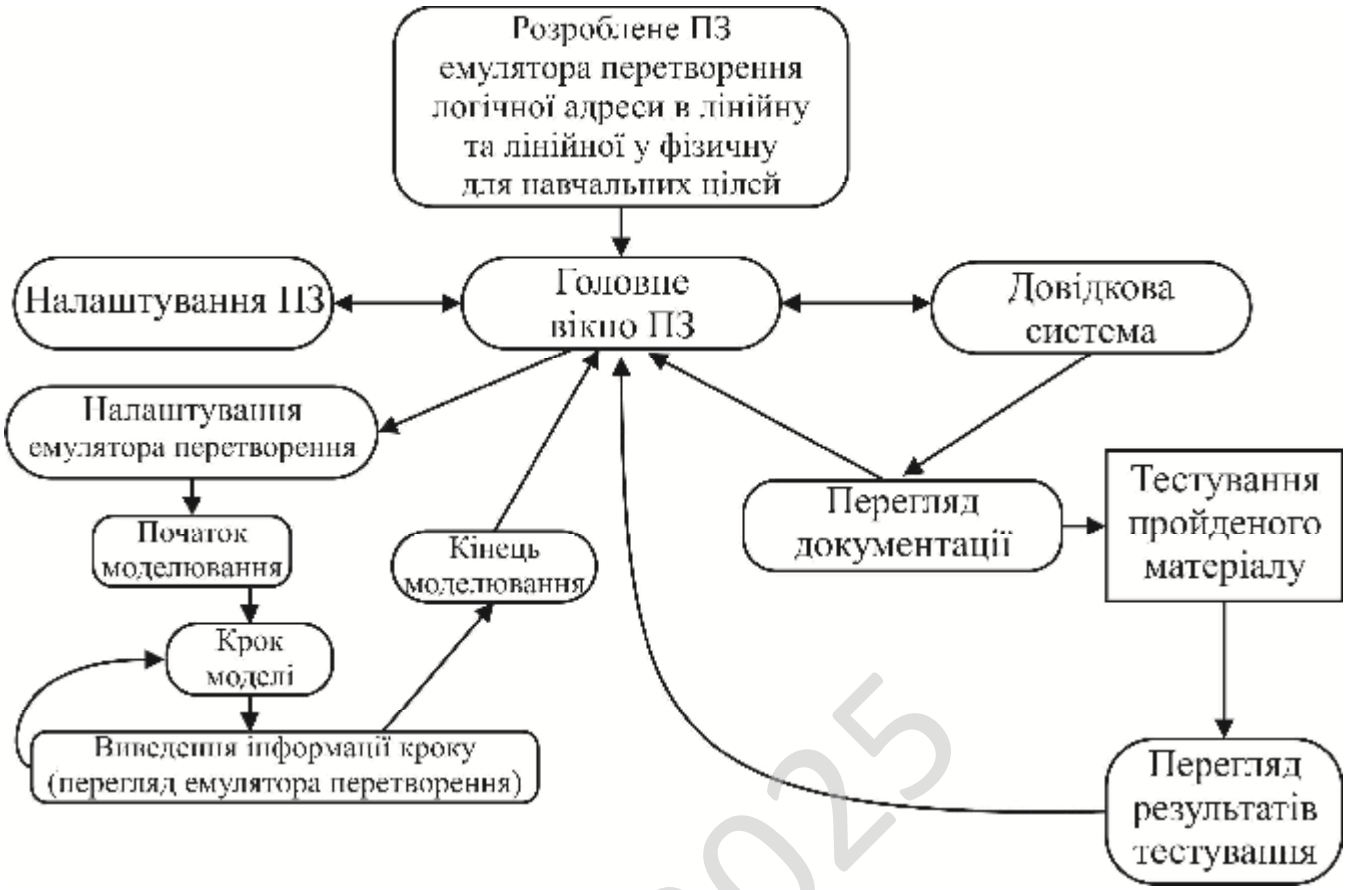

Рисунок 3.4 – Структурна схема системи

3.3 Розробка функціональної схеми

На рисунку 3.5 зображена функціональна схема роботи системи де зображено основну схему емулятора.

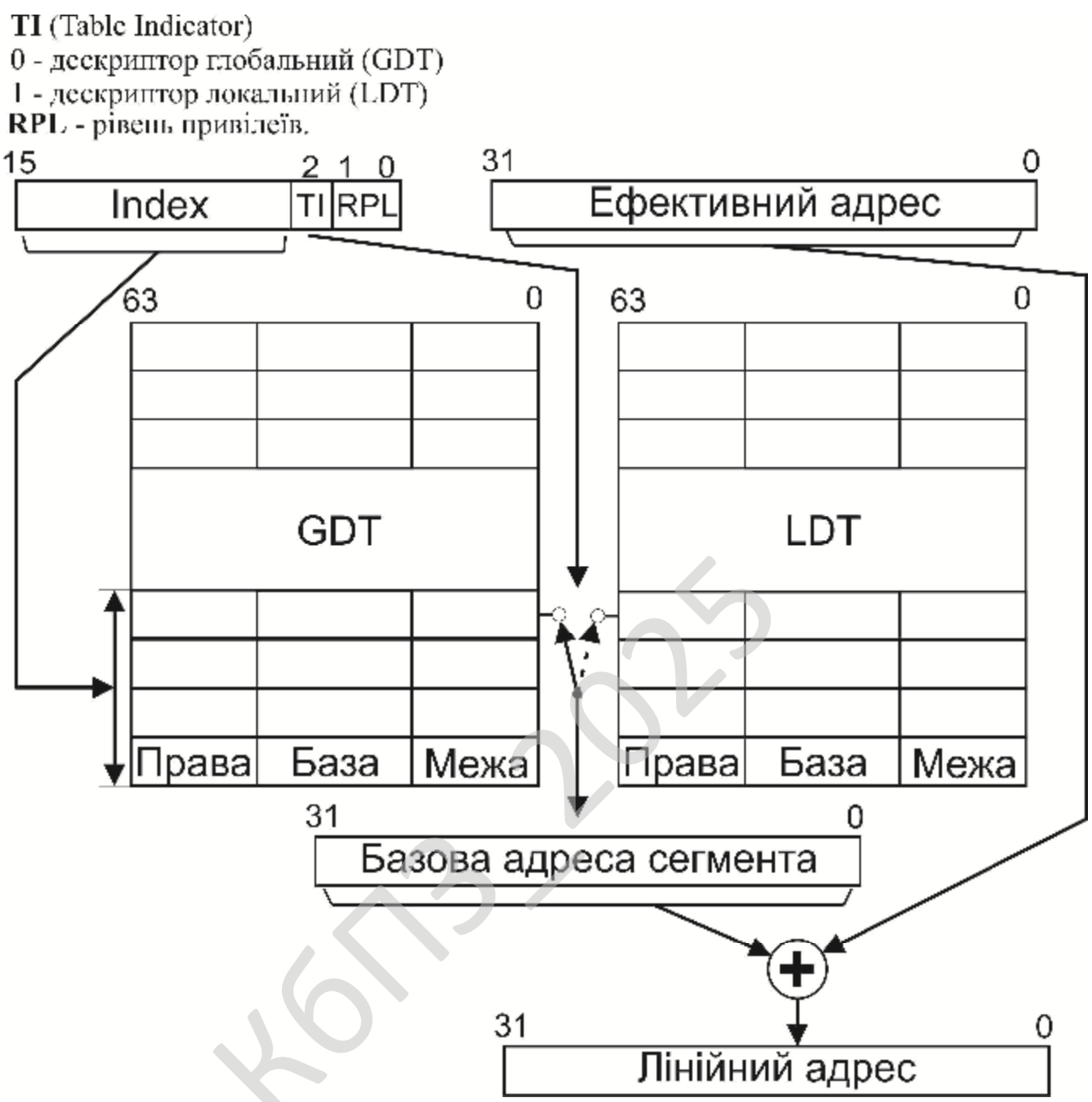

Рисунок 3.5 – Функціональна схема системи

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

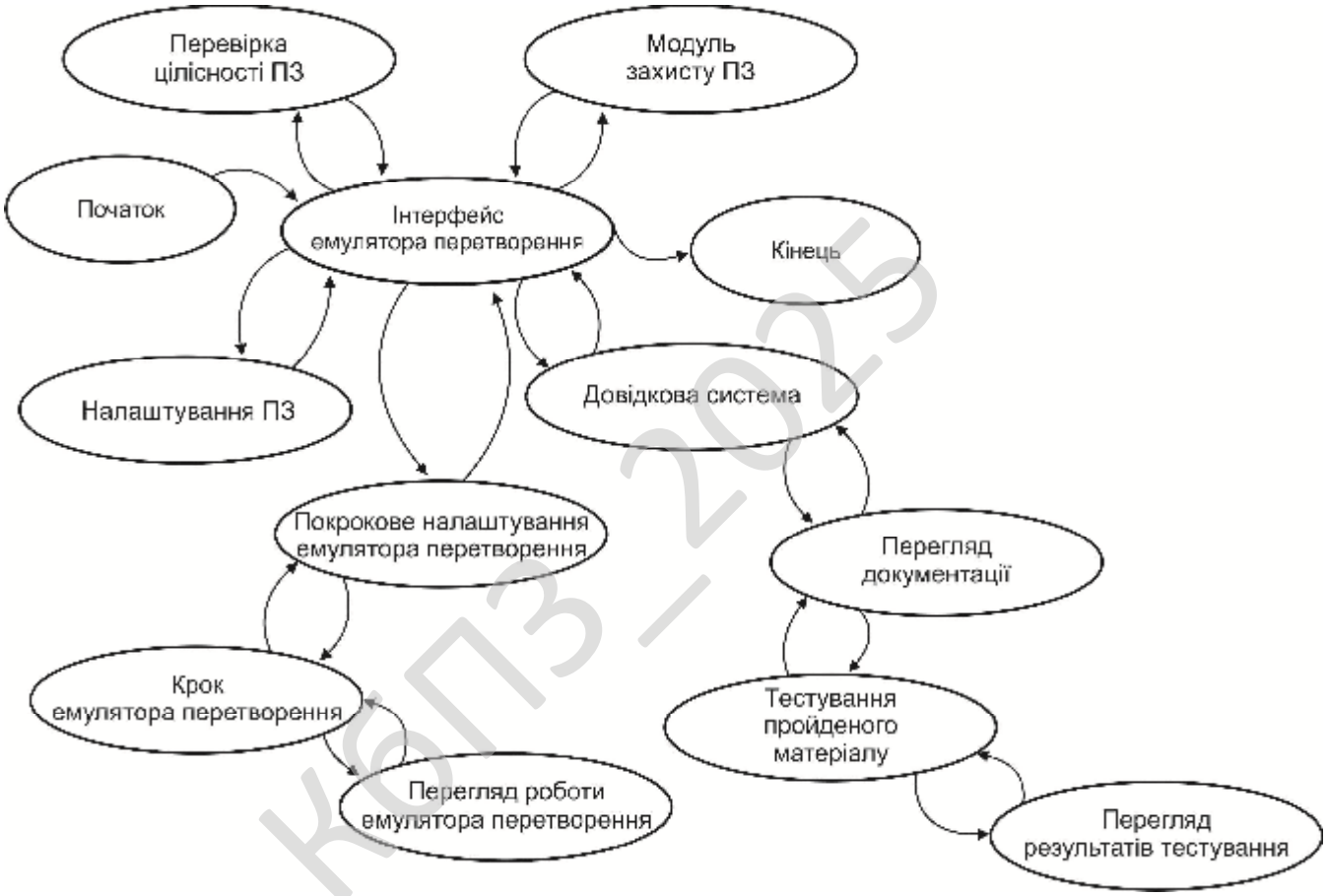

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає

уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над магістерською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

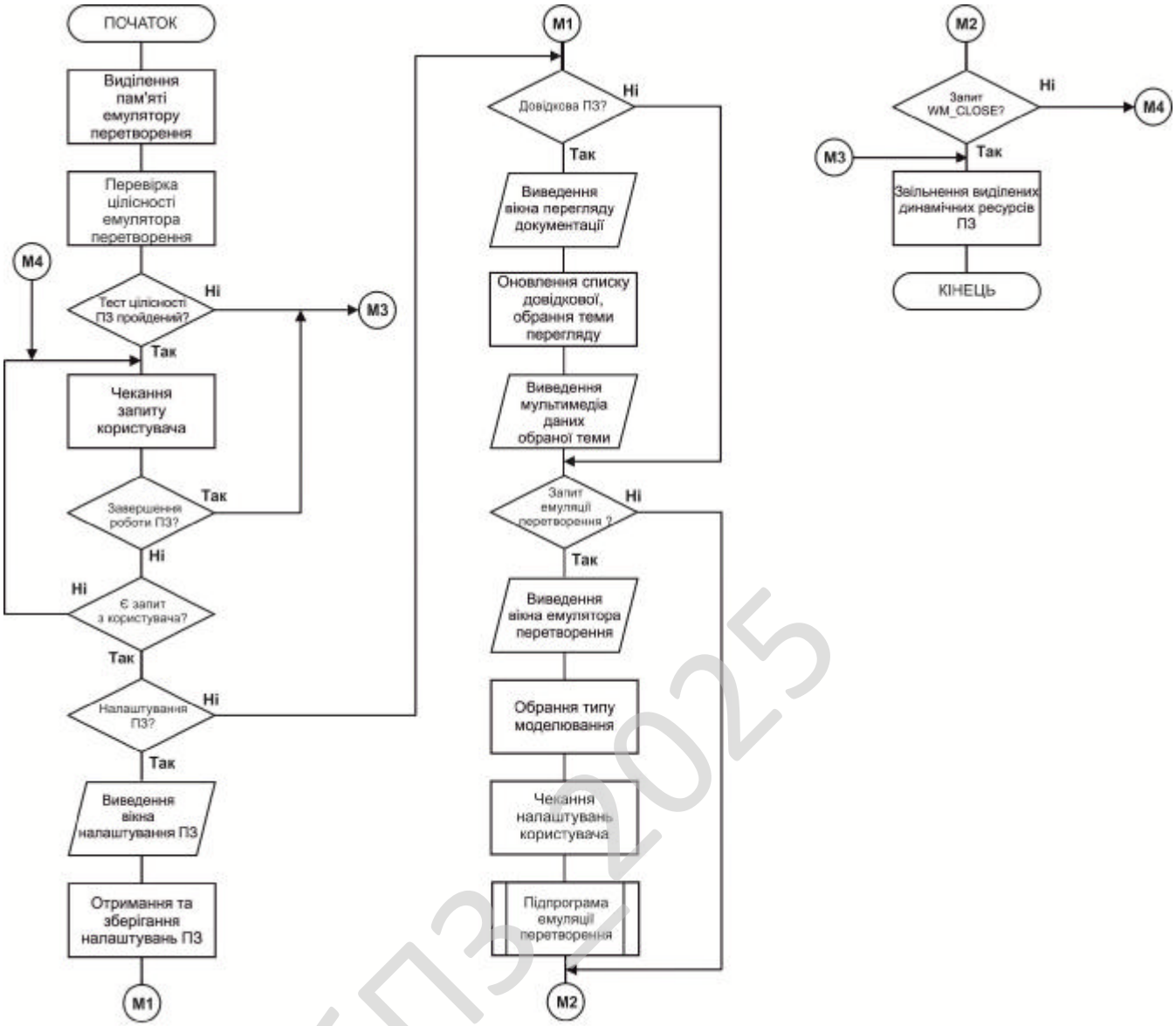

Рисунок 4.1 – Блок-схема основної програми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

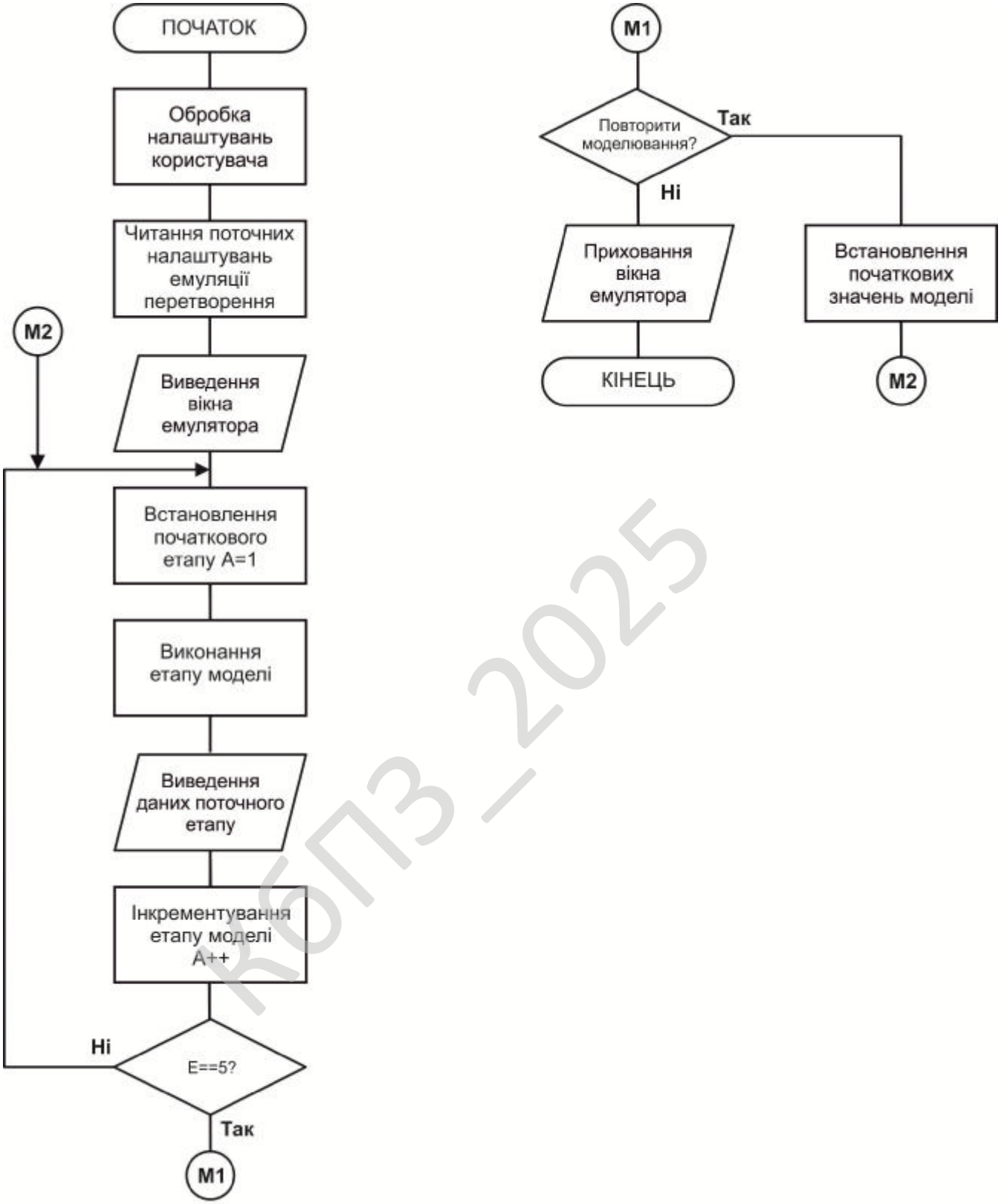

Рисунок 4.2 – Блок-схема роботи підпрограми

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці магістерської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу).

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною

одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Розглянемо розроблені алгоритми та застосовані технології які використовувались при написанні магістерської дипломної роботи.

Система емулятора перетворення логічної адреси в лінійну та лінійної у фізичну у пояснювальній записці описується як навчальний консольний застосунок на Python.

Програма моделює поєднання сегментації та сторінкової організації пам’яті, приймає логічну адресу, покроково обчислює проміжні значення і показує їх студентові у зручному для аналізу вигляді.

Параметри моделі та структури даних

У навчальній конфігурації логічна і фізична адреси мають довжину двадцять чотири біти. Розмір сторінки дорівнює чотири кібібайти, тобто два у дванадцятому. Це означає що молодші дванадцять біт лінійної адреси є зміщенням у межах сторінки, а старші дванадцять біт визначають номер сторінки. Загальна кількість сторінок і фізичних кадрів дорівнює два у дванадцятому, що відповідає чотири тисячі дев'яносто шість. Такий обсяг з одного боку демонструє масштаб справжньої системи пам'яті, а з іншого боку залишається зручним для ручної перевірки розрахунків.

У сегментній частині моделі логічна адреса розбивається на селектор сегмента та зміщення. Селектор займає вісім біт, що дає до двохсот п'ятдесяти шести можливих сегментів. Зміщення займає шістнадцять біт, що дозволяє кожному сегменту мати до шістдесяти п'яти тисяч п'ятисот тридцяти шести байт. Цього достатньо для модельних програм навчального характеру.

У Python ці параметри оформлюються як константи і прості структури даних.

```
from dataclasses import dataclass
LOGICAL_BITS = 24
PHYSICAL_BITS = 24

PAGE_BITS = 12
PAGE_SIZE = 1 << PAGE_BITS

SEGMENT_SELECTOR_BITS = 8
OFFSET_BITS = 16

@dataclass
class SegmentDescriptor:
    base: int
    limit: int
    present: bool

@dataclass
class PageTableEntry:
    frame: int
```

```
    present: bool
```

У поясненні до цього фрагмента зазначається що вибір двадцяти чотирьох біт дає фізичний простір пам’яті шістнадцять мегабайт. При такому просторі таблиці сторінок з кількома тисячами записів займають лише незначну частину пам’яті, що добре ілюструє реальний принцип коли службові структури займають малу частку від усього обсягу.

Логічна адреса та модуль сегментації

Модуль сегментації відповідає за перший етап перетворення. Він зберігає таблицю сегментів для поточного процесу і здійснює перехід від пари селектор та зміщення до лінійної адреси. Таблиця реалізується через словник Python, де ключем є ціле число селектора, а значенням є дескриптор сегмента.

```
class SegmentTable:
    def __init__(self):
        self.segments: dict[int, SegmentDescriptor] = {}

    def add_segment(self, selector: int, desc: SegmentDescriptor):
        self.segments[selector] = desc

def translate_logical_to_linear(self, selector: int, offset: int) -> int:
        desc = self.segments.get(selector)
        if desc is None:
            raise RuntimeError("unknown segment selector")
        if not desc.present:
            raise RuntimeError("segment not present")
        if offset > desc.limit:
            raise RuntimeError("segmentation fault")
        return desc.base + offset
```

Такий код наочно показує студенту що перевірка меж сегмента відбувається ще до звернення до сторінкової підсистеми. Якщо зміщення перевищує межу, програма формує помилку сегментації. Можна навести умовний приклад.

Припустимо, що сегмент коду має базу у лінійному просторі чотири тисячі дев’ятсот дев’яносто шість і межу шістнадцять тисяч.

Якщо студент задає зміщення сто, лінійна адреса дорівнює чотири тисячі дев’ятсот дев’яносто шість плюс сто.

Якщо студент вводить зміщення більше за межу, модуль негайно зупиняє обчислення, що добре демонструє захист пам’яті.

Лінійна адреса і сторінкова схема.

Другий етап перетворення реалізує сторінкову організацію. Лінійна адреса ділиться на номер сторінки і зміщення у межах сторінки. Номер сторінки додатково розкладається на індекс директорії сторінок та індекс таблиці сторінок.

У моделі для цього використовується дві групи по шість біт. Тоді директорія містить шістдесят чотири записи і кожна таблиця також містить до шістдесяти чотирьох записів.

Максимальна кількість можливих комбінацій становить чотири тисячі дев’ятсот шістдесят. Це трохи більше ніж чотири тисячі дев’яносто шість реальних сторінок, тому модель повністю покриває весь адресний простір.

У Python це оформлюється у вигляді класу директорії, який у словнику зберігає таблиці сторінок.

```
class PageDirectory:
    def __init__(self):
        self.tables: dict[int, dict[int, PageTableEntry]] = {}

    def map_page(self, linear_page: int, frame: int):
        dir_index = (linear_page >> 6) & 0x3F
        tbl_index = linear_page & 0x3F
        table = self.tables.setdefault(dir_index, {})
        table[tbl_index] = PageTableEntry(frame=frame, present=True)

    def translate_linear_to_physical(self, linear_addr: int) -> int:
        page = linear_addr >> PAGE_BITS
        offset = linear_addr & (PAGE_SIZE - 1)
        dir_index = (page >> 6) & 0x3F
        tbl_index = page & 0x3F
        table = self.tables.get(dir_index)
        if table is None:
            raise RuntimeError("page directory miss")
        entry = table.get(tbl_index)
```

```
        if entry is None or not entry.present:
            raise RuntimeError("page fault")
        frame_base = entry.frame * PAGE_SIZE
        return frame_base + offset
```

У поясненні до цього фрагмента вказується що функція map_page задає відображення між номером лінійної сторінки та номером фізичного кадру.

Це моделює завантаження сторінок процесу у фізичну пам’ять. Функція translate_linear_to_physical розбиває адресу на частини, знаходить відповідну таблицю і запис таблиці та повертає фізичну адресу.

Якщо у директорії або таблиці запис відсутній, формується подія сторінкового порушення, що демонструє механізм підкачки.

У записці можна навести числовий приклад. Якщо розмір сторінки чотири кібібайти, а лінійна адреса має значення тридцять дві тисячі, номер сторінки дорівнює тридцять дві тисячі поділити на чотири тисячі дев’ятсот дев’яносто шість, а зміщення дорівнює залишку від ділення.

Далі цей номер сторінки розкладається на старші шість біт для директорії та молодші шість біт для таблиці. Після знаходження кадру у таблиці функція додає зміщення і отримує кінцеву фізичну адресу.

Центральний клас емулятора і навчальний сценарій

У головному модулі всі складові поєднуються у клас Emulator. Він створює таблицю сегментів, директорію сторінок та організовує повний цикл перетворення адреси.

Центральний метод приймає логічну адресу, викликає модуль сегментації, а потім сторінковий модуль, після чого повертає усі проміжні значення для відображення студенту.

```
class Emulator:
    def __init__(self):
        self.segment_table = SegmentTable()
        self.page_directory = PageDirectory()

    def translate_full(self, selector: int, offset: int) -> dict:
linear = self.segment_table.translate_logical_to_linear(selector, offset)
        physical = self.page_directory.translate_linear_to_physical(linear)
```

```
            return {
                "selector": selector,
                "offset": offset,
                "linear": linear,
                "physical": physical,
        }

        def demo():
            emu = Emulator()
            code_seg = SegmentDescriptor(base=0x1000, limit=0x3FFF, present=True)
            emu.segment_table.add_segment(1, code_seg)
            emu.page_directory.map_page(linear_page=1, frame=5)
            emu.page_directory.map_page(linear_page=2, frame=9)
            result = emu.translate_full(selector=1, offset=0x0500)
            print("логічна селектор", result["selector"])
            print("зміщення", result["offset"])
            print("лінійна адреса", result["linear"])
            print("фізична адреса", result["physical"])
```

У пояснювальному тексті зазначається що у функції demo створюється сегмент коду з базою у лінійному просторі, задаються кілька сторінок у директорії, після чого виконується повний переклад однієї логічної адреси.

Вивід показує селектор, зміщення, лінійну та фізичну адресу, що дозволяє студенту перевірити ручні обчислення.

При аналізі цього прикладу студент самостійно розкладає лінійну адресу на сторінку та зміщення, а потім порівнює отриману фізичну адресу з результатом програми, що підтверджує правильність обраних параметрів і реалізованих алгоритмів.

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму CAST-128 (або CAST5) у криптографії, це блоковий алгоритм симетричного шифрування на основі мережі Фейстеля, який використовується в цілому ряді продуктів криптографічного захисту, зокрема деяких версіях PGP і GPG і крім того схвалений для використання Канадським урядом.

Основні відомості

Алгоритм був створений в 1996 році Карлайлом Адамсом (Carlisle Adams) і Стаффордом Таваресом (Stafford Tavares) використовуючи метод побудови шифрів CAST, який використовується також і іншим їхнім алгоритмом CAST-256 (алгоритм-кандидат AES).

CAST-128 складається з 12 або 16 раундів мережі Фейстеля з розміром блоку 64 біта й довжиною ключа від 40 до 128 біт (але тільки з інкрементацією по 8 біт). 16 раундів використовуються коли розміри ключа перевищують 80 біт. В алгоритмі використовуються 8x16 S-блоки, засновані на бент-функції, операції XOR і модулярной арифметиці (медулярне додавання й вирахування). Є три різні типи функцій раундів, але вони схожі за структурою й різняться тільки у виборі виконуваної операції (додавання, вирахування або XOR) у різних місцях.

Хоча CAST-128 захищений патентом Entrust, його можна використовувати в усьому світі для комерційних або некомерційних цілей безкоштовно.

Опис

CAST – це популярний 64-бітовий шифр, що допускає розміри ключа аж до 128 біт

Алгоритм CAST використовує 64-бітовий блок і 64-бітовий ключ. CAST стійкий до диференціального й лінійного криптоаналізу. Сила алгоритму CAST укладена в його S-блоках. В CAST немає фіксованих S-блоків і для кожного додатка вони конструюються заново. Створений для конкретної реалізації CAST S-блок уже більше ніколи не міняється. Інакше кажучи, S-блоки залежать від реалізації, а не від ключа. Northern Telecom використовує CAST у своєму пакеті програм Entrust для комп'ютерів Macintosh, PC і робочих станцій UNIX. Обрані ними S-блоки не опубліковані, що втім не дивно.

CAST-128 належить компанії Entrust Technologies, але є безкоштовним як для комерційного, так і для некомерційного використання. CAST-256 – безкоштовне доступне розширення CAST-128, яке ухвалює розмір ключа до 256

біт і має розмір блоку 128 біт. CAST-256 був одним з первісних кандидатів на AES.

Опис алгоритму

CAST-128 заснований на мережі Фейстеля. Повний алгоритм шифрування викладений у наступних чотирьох кроках:

ВХІД: текст m_1 ... m_{64}, ключ K = k_1 ... k_{128}.

ВИХІД: зашифрований текст c_1 ... C_{64}.

1. (розгорнення ключа) становить 16 пару підключів {Km_i, Kr_i} отриманих з K (див. розділи Пари раундових ключів і Неідентичні раунди).

2. (L_0, R_0) <- (m_1. .. m_{64}). (Розділяє текст на ліву й праву 32-бітні половини L_0 = m_1 ... m_{32} і R_0 = m_{33} ... m_{64}).

3. (16 раундів) for i from 1 to 16, обчислити L_i і R_i у такий спосіб: L_i = R_{i-1}; R_i = L_{i-1} ^ F(R_{i-1},Km_i,Kr_i), де F визначена в розділі «Пари раундових ключів» (F має тип 1, тип 2, тип 3 або, залежно від i).

4. c_1 ... c_{64} <- (R_{16}, L_{16}). (Міняємо остаточні блоки місцями L_{16}, R_{16} і поєднуємо, щоб сформувати зашифрований текст.)

Розшифруванні збігається з алгоритмом шифрування, наведеним вище, крім того, що раунди (i, отже, пари підключів), використовуються у зворотному порядку, щоб обчислити (L_0, R_0) з (R_{16}, L_{16}).

Пари раундових ключів

CAST-128 використовує пару підключів за раунд: 32-бітні величини Km використовується в якості "маскування" ключа й Kr використовують як "перестановки" ключа, з яких використовуються тільки початкові 5-біт.

Неідентичні раунди

Три різні типів функції використовуються в CAST-128. Типи виглядає в такий спосіб (де "D" є вхідними даними у функцію F і "Ia"-"Id" є найбільш значимий байт – найменш значимий байт I, відповідно). Зверніть увагу, що "+" і "-" додавання й вирахування по модулю 2 ** 32, "" є побітове XOR і "<<<" є циклічним зрушенням уліво.

Раунди	$I = ((Km_i + R_{i-1}) <<< Kr_i)$
1,4,7,10,13,16	$F = ((S1[I_a]\ \ S2[I_b]) - (S3[I_c])) + S4[I_d]$
Раунди	$I = ((Km_i \wedge R_{i-1}) <<< Kr_i)$
2,5,8,11,14	$F = ((S1[I_a] - S2[I_b]) + (S3[I_c]))\ \ S4[I_d]$
Раунди	$I = ((Km_i - R_{i-1}) <<< Kr_i)$
3,6,9,12,15	$F = ((S1[I_a] + S2[I_b])\ \ (S3[I_c])) - S4[I_d]$

Поля заміни

CAST-128 використовує вісім полів заміни: поля S1, S2, S3 і S4 раундові функції полів заміни, S5, S6, S7 і S8 є ключами розгорнення полів заміни. Незважаючи на те, що 8 полів заміни вимагають у цілому 8 Кбайт для зберігання, зверніть увагу на те, що тільки 4 Кбайта потрібні під час фактичного шифрування / дешифрування, тому що генерація підключа звичайно робиться до будь-якого введення даних.

Ключі розгорнення

Представимо 128-розрядний ключ у вигляді x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf, де x0 старший байт, і xf молодший байт.

Представимо z0..zf проміжними (тимчасовими) байтами. Si[] представляє поле заміни і і " ^ " представляє додавання по Xor'у.

Поля заміни формуються із ключа x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf у такий спосіб.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K1 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K2 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K3 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K4 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K5 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K6 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K7 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K8 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K9 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K10 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K11 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K12 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K13 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K14 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K15 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K16 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

половина, що залишається, ідентична тому, що дане вище, продовження від останнього створило x0..xf, щоб генерувати ключі K17 – K32.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K17 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K18 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K19 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K20 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K21 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K22 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K23 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K24 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K25 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K26 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K27 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K28 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K29 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K30 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K31 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K32 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

Маскування й перестановка підключів

Km_1, ..., Km_{16} 32-розрядні підключи маскування (один на раунд).

Kr_1, …, Kr_{16} 32-розрядні перестановки підключів (один на раунд); тільки молодші 5 бітів використовуються в кожному раунді.

for (i=1; i<=16; i++) { Km_i = Ki; Kr_i = K16+i; }

Змінний розмір ключа

CAST-128 Алгоритм шифрування був розроблений, щоб розмір ключа міг варіюватися від 40 до 128 біт, в 8-бітному кроці (тобто припустимі розміри ключа рівняються 40, 48, 56, 64..., 112, 120, і 128 бітам). Для змінної роботи розміру ключа специфікація наступні:

1) Для розмірів ключа до й включаючи 80 бітів (тобто, 40, 48, 56, 64, 72, і 80 бітів) алгоритм точно такої ж, але використовує 12 раундів замість 16;

2) Для розмірів ключа більше, чим 80 бітів, алгоритм використовує повні 16 раундів;

3) Для розмірів ключа менше, чим 128 бітів ключ доповнений нульовими байтами (у самих правих, або молодших, позиціях) до 128 биток (тому що розклад ключа CAST 128 ухвалює вхідний ключ 128 бітів).

Розшифрування

Розшифрування для CAST-128 відносно проста. Розшифрування працює в тому ж алгоритмічному напрямку, що й шифрування, починаючи із зашифрованого тексту як вхідних даних. При цьому підключ використовуються у зворотному напрямку.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання магістерської дипломної роботи.

Розроблене програмне забезпечення емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Налаштування; Документація; Емулятор; Довідка.

– Блоку представлення даних у вигляді графічної схеми.

– Блоку закладок: Емулятор перетворення; Параметри емулятора; Довідка.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

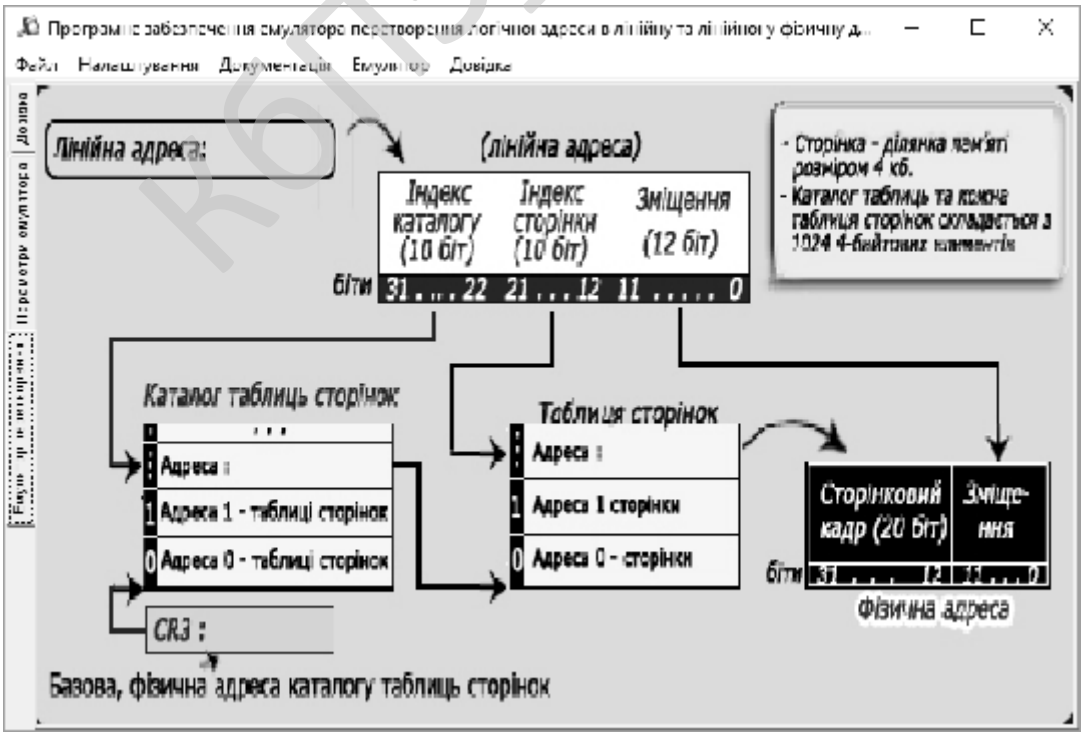

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

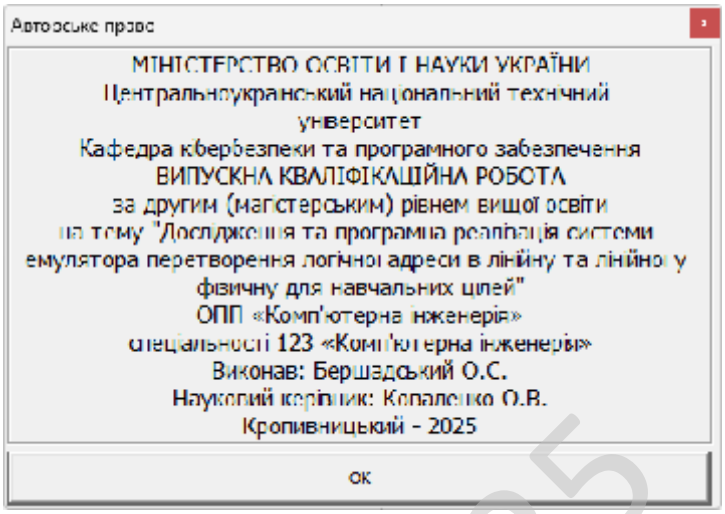

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним

фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Метою розробки є дослідження та програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Об'єктом дослідження є процес емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Предметом дослідження є методи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Методи дослідження базуються на методах комп'ютерної логіки, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

– Розроблено вітчизняний продукт емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей, який має більш широкі можливості, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження такої системи можуть бути насамперед цікавими для студентів технічних спеціальностей, які вивчають архітектуру комп'ютерів, операційні системи або системне програмування. Адже саме на прикладі подібних емуляторів можна краще зрозуміти, як процесор працює з пам'яттю, як відбувається адресація та які етапи проходить дані під час перетворення.

Викладачам університетів і коледжів цей проєкт також може стати корисним інструментом для демонстрації складних теоретичних процесів у більш наочній і доступній формі. Можливість показати студентам, як саме від логічної адреси формується фізична, сприяє кращому засвоєнню матеріалу та робить навчання більш практичним.

Для розробників навчального програмного забезпечення або авторів освітніх курсів такий емулятор може стати частиною інтерактивної платформи або лабораторного модуля. Це дозволяє створювати нові формати дистанційного навчання, що поєднують симуляцію, тестування та візуалізацію.

Також розробка може бути цікавою для науковців, які займаються питаннями комп'ютерної педагогіки, адже вона показує, як складні технічні процеси можна перетворити на зрозумілі моделі для навчального використання.

У підсумку, така система має як освітню, так і методичну цінність – вона допомагає поєднати теорію з практикою і зробити вивчення принципів адресації пам'яті більш доступним та сучасним.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Оцінку привабливості проєкту можна провести за допомогою опитування групи експертів, до якої увійдуть викладачі інформатики, системного програмування, а також фахівці з освітніх технологій. Кожен з них може оцінити систему за такими критеріями, як зручність інтерфейсу, навчальна ефективність, технічна надійність і потенціал для використання у навчальному процесі.

Після цього кожен критерій можна оцінити за шкалою від 1 до 10, і на основі середнього арифметичного значення отримати інтегральний показник привабливості. Якщо, наприклад, середня оцінка перевищує 8 балів, це може свідчити про високу педагогічну цінність та готовність до практичного впровадження.

Важливо, що експертне опитування не лише дає числовий результат, а й супроводжується якісними коментарями. Експерти можуть вказати, які функції варто доопрацювати, як краще адаптувати програму до навчальних планів, або які візуальні елементи допоможуть зробити процес симуляції більш зрозумілим.

Такі результати дозволяють розробникам оцінити не лише технічну, а й дидактичну ефективність проєкту, що особливо важливо для освітніх систем.

У підсумку, метод експертних оцінок допомагає об’єктивно визначити сильні сторони емулятора та його потенціал у навчальному середовищі, перш ніж починати його широке впровадження.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості розробки системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну доцільно застосувати витратний метод. Це пояснюється тим, що подібні проєкти часто мають навчальну, а не комерційну мету, тому визначення собівартості є більш об’єктивним і практичним.

У межах цього підходу оцінюються всі ресурси, які необхідні для створення системи: час програмістів, вартість ліцензійного програмного забезпечення, витрати на тестування, а також супровід і оновлення програми. Важливо враховувати навіть непрямі витрати, наприклад, підготовку методичних матеріалів або навчання викладачів, які користуватимуться системою.

Перевага витратного методу полягає у його прозорості – кожен елемент витрат можна обґрунтувати, що важливо при поданні проєкту на гранти чи фінансування з боку навчального закладу.

Якщо ж система планується до ширшого розповсюдження, доцільно доповнити оцінку витрат аналізом можливого прибутку – наприклад, через продаж ліцензій для інших університетів або інтеграцію в освітні онлайн-платформи.

Таким чином, витратний метод дозволяє отримати реалістичну оцінку вартості проєкту та забезпечити економічне обґрунтування для його подальшої реалізації.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Впровадження нової системи має на меті підвищити якість навчання, скоротити витрати часу викладачів і зменшити потребу у придбанні комерційного ПЗ для моделювання. Вихідні дані для розрахунку економічної ефективності вносимо в таблицю 7.1.

Розрахунок економічного ефекту дає змогу говорити про: економія на ліцензійному ПЗ – 20 000 грн, економія часу викладачів (на підготовці занять) – 40 000 грн/рік; загальний річний економічний ефект – 60 000 грн/рік, термін окупності (Payback Period) – 0,58 року (~7 місяців); коефіцієнт економічної ефективності (E) – 171%.

Таблиця 7.1 – Вихідні дані для розрахунків

Показник	**До впровадження**	**Після впровадження**	**Ефект / Економія**
Вартість використання стороннього програмного забезпечення (щорічна ліцензія)	20 000 грн	0 грн (власна розробка)	–20 000 грн
Витрати на підготовку лекційного матеріалу	15 год × 250 грн = 3 750 грн	8 год × 250 грн = 2 000 грн	–1 750 грн
Кількість лабораторних занять за семестр	8	8	—
Середня тривалість підготовки до одного заняття	4 год	2 год	–2 год
Річна кількість груп, що використовують емулятор	10	10	—
Вартість створення системи (одноразово)	—	—	35 000 грн

Додаткові (немонетарні) вигоди: підвищення ефективності навчального процесу – студенти краще розуміють матеріал завдяки візуалізації процесів адресації; можливість використання емулятора в дистанційній освіті без додаткових витрат; формування цифрової компетентності викладачів і студентів; зменшення навантаження на лабораторну базу, адже навчальні процеси переносяться у віртуальне середовище; створення власного освітнього ресурсу, який може бути використаний іншими закладами.

Впровадження системи емулятора для навчальних цілей є економічно ефективним – розробка окупається менш ніж за рік. Отримана річна економія становить 60 000 грн, що перевищує початкові витрати майже у 1,7 раза. Крім фінансової вигоди, система забезпечує значний освітній ефект, підвищуючи якість навчання, інтерес студентів до предмету та ефективність роботи викладачів.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Просування такого навчального програмного продукту має починатися з презентації його можливостей у навчальних закладах – університетах, коледжах та ІТ-академіях. Важливо показати, як емулятор може допомогти викладачам пояснювати складні процеси і зробити навчання більш наочним.

Наступним кроком може бути створення демонстраційної версії програми, яку можна безкоштовно завантажити для ознайомлення. Це дозволить користувачам протестувати інтерфейс і переконатися у практичній користі продукту.

Важливою частиною просування стане участь у конференціях з освіти та інформаційних технологій, де можна представити проєкт науковій спільності та потенційним партнерам. Це підвищить його впізнаваність і авторитет серед освітян.

Також ефективно створити відеоінструкції або короткі навчальні ролики, які демонструватимуть, як працює система, як виконуються приклади та які переваги вона має перед традиційним викладанням.

У результаті, успішне просування проєкту базується не на агресивному маркетингу, а на створенні цінності для навчального процесу та побудові довіри серед освітніх спільнот.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації реалізації емулятора доцільно використовувати кілька паралельних напрямів. Один з них – це партнерство з освітніми закладами, які можуть впровадити систему у свої лабораторні курси або онлайн-навчальні платформи. Це забезпечить сталість використання та сприятиме поширенню продукту.

Також варто розглянути можливість розповсюдження програми через відкриті платформи, наприклад GitHub чи освітні портали, що спеціалізуються на комп'ютерних науках. Це допоможе залучити користувачів, які цікавляться системним програмуванням, і водночас отримати зворотний зв'язок для вдосконалення проєкту.

Якщо продукт планується монетизувати, можна застосувати ліцензійну модель: безкоштовний базовий доступ для навчання і розширений – для викладачів або установ із підтримкою додаткових функцій. Такий підхід знижує поріг входу і водночас стимулює придбання повної версії.

Ще одним ефективним кроком стане створення навчального курсу чи серії воркшопів, у яких система використовується як головний інструмент. Це допоможе закріпити її репутацію як практичного освітнього рішення.

Таким чином, оптимізація збуту має ґрунтуватися на поєднанні доступності, академічного визнання та постійного вдосконалення продукту відповідно до потреб користувачів.

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключовим фактором успіху цього проєкту є його навчальна цінність. Якщо система реально допомагає студентам краще розуміти принципи адресації пам’яті та бачити зв’язок між логічними, лінійними і фізичними адресами, вона матиме довготривале застосування.

Не менш важливим є зручність інтерфейсу та інтуїтивність використання. Навчальні продукти мають бути простими, адже складність у користуванні може відлякати користувачів навіть при високій технічній якості.

Велике значення має стабільність роботи системи – якщо вона не зависає, коректно відображає результати і не має помилок у логіці, то викладачі будуть схильні довіряти їй і використовувати її в навчальному процесі.

Додатковим фактором успіху є наявність підтримки спільноти користувачів. Якщо студенти та викладачі зможуть обмінюватися матеріалами, прикладами та досвідом, це створить живе середовище навколо продукту.

Зрештою, успіх визначається тим, наскільки система не просто відтворює процеси, а допомагає зрозуміти їхню суть. Коли технологія стає зрозумілою завдяки простому інструменту, вона виконує свою головну освітню місію.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

В охорону праці включають санітарно-гігієнічні, лікувально-профілактичні та організаційно-технічні системи правових і соціально-економічних заходів.

В кожній ІТ компанії є трудові відносини з працівниками. Згідно закону України “Про охорону праці” [1] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні;

– Положення про охорону праці;

– Накази з охорони праці;

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час

роботи з екранними пристроями» [2], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [3].

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [3], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп’ютера на організм людини визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машин (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у тому числі програмісти) необхідно створити належний мікроклімат, параметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у тому числі високочастотні) випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м'язовий апарат.

8.3 Аналіз умов праці на робочому місці фахівця

Робота програміста пов'язана з постійною роботою на ЕОМ, яка відбувається у кімнаті розмірами 4,4 м×6,2 м×2,9 м. Одна з її більших стін має шість двостулкових вікон, розмірами 2 м×1,8 м, які виходять на північний захід. Вікна розташовані рівномірно по всій довжині стіни. Підлога в кімнаті покрита лінолеумом, всі стіни пофарбовані світло оранжевого кольору до висоти 2,8 м, а далі підвісна стеля. Уздовж стін розташовані комп'ютерні столи. На них розташовуються 2 персональні комп'ютери й інша оргтехніка (сканер принтери, телефони й ксерокс). Столи мають пластикове покриття. Габарити їхньої робочої поверхні 1255 мм×845 мм. Висота столів 760 мм. Висота стільців від рівня підлоги становить 430 мм.

Згідно НПАОП 0.00 – 1.28 – 10 «Правила охорони праці під час електронно-обчислювальних машин» площа повинна задовольняти умові – не менш 6 $м^2$ на одне робоче місце. Кратність повітрообміну в приміщенні вузла також регламентується ДСанПіН 3.3.2.007-98 [3], вона повинна становити 20 $м^3$/годину на одне місце. Виконання даних вимог забезпечить підтримку в приміщенні вузла оптимального значення вологості й складу повітря.

Відповідно ДБН В.2.5 – 28 – 2006 [4] роботу програміста можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення вузла можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при сполученому висвітленні), повинен становити 0,5%, освітленість при штучному висвітленні повинна становити 300 лк.

За результатами виміру освітленості відділом охорони праці величина освітленості від системи загального штучного висвітлення лежить у межах 200-250 лк, що не відповідає вимогам, які пред'являються до приміщення.

Відповідно ДСанПіН 3.3.2.007-98 [3] рівні звукового тиску в робочому приміщенні не повинні перевищувати в октавних смугах із середньо геометричними частотами наступних значень, наведених у таблиці 8.1.

У приміщенні перебувають наступні джерела шуму: електродвигуни внутрішнього вентилятора ЕОМ; працюючі принтери; працюючі дисководи.

Таблиця 8.1 – Допустимі спектри рівнів звукового тиску

Робоче місце	Рівень звукового тиску, дБ, в октавних смугах із середньо геометричними частотами, Гц								Рівень звуку і еквівалентний рівень звуку, дБА
	63	125	250	500	1000	2000	4000	8000	
Приміщення конструкторських бюро, програмістів обчислювальних машин, лабораторій для теоретичних робіт і опрацювання експериментальних даних, прийому хворих в медпунктах	71	61	54	49	45	42	40	38	50

Шум, вироблений вентилятором можна класифікувати як постійний, всі інші джерела шуму, як імпульсні. Відповідно паспорта на приміщення рівень звуку, Дб(А), обмірюваний за шкалою (А) шумоміра досяг величини 28,3 Дб(А)

при роботі всього встаткування вузла, включаючи й ксерокс. Це дозволяє зробити висновок про відповідність рівня звуку в приміщенні вимогам нормативних актів.

Ергономічні вимоги до робочого місця працюючого з ВДТ ЕОМ і ПЕОМ нормуються НПАОП 0.00 – 1.28 – 10**.** Оптимальне положення тіла того, що працює забезпечується відповідною конструкцією робочого місця, а також регуляцією висоти робочої поверхні, сидіння, простори й підставки для ніг. Даного місця програміста не мають регульованих параметрів. Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту дані в таблиці 8.2.

Таблиця 8.2 – Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту

Ріст людини, см	Висота робочої поверхні мм,	Висота простору для ніг, мм	Висота робочого сидіння, мм
175	765(740)	655(600)	450(440)

У дужках зазначені реальні значення параметрів робочого місця; всі вони не відповідають параметрам, зазначеним у стандарті.

Параметри мікроклімату можуть мінятися в широких межах, тоді як необхідною умовою життєдіяльності людини є підтримка сталості температури тіла завдяки властивості терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах ДСН 3.3.6.042 – 99 [4] встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. табл. 8.3).

Таблиця 8.3 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	22 – 24°С 40 – 60% до 0,1 м/с
Теплий	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	23 – 25°С 40 ... 60% 0,1 ... 0,2 м / с

8.4 Розробка заходів з поліпшення охорони праці

Провівши аналіз умов праці в розглянутому вище приміщенні, було отримано наступні результати:

– значення мікроклімату в приміщенні не перевищує норму;

– розрахунки розміру робочого місця на одного працівника відповідають нормі;

– рівень шуму в приміщенні не становить вище норми.

З вище перерахованих результатів можна зробити висновок, що основний вплив на продуктивність ІТ-спеціалістів є його психологічний стан. Тому є доцільним зменшити рівень стресу на робочому місці.

Рекомендовані наступні заходи: За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2м. Конструкція робочого місця користувача персонального комп’ютера має забезпечити підтримання оптимальної робочої пози офісного працівника. Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея,

клавіатури, принтера) і документів. Висота робочої поверхні робочого столу має регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1400мм, глибина – 800-1000мм). Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600 мм, завширшки не менше ніж 500 мм, за глибиною (на рівні колін) не менше ніж 450 мм, на рівні простягнутої ноги не менше ніж 650 мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим [5].

8.5 Розрахункова частина

Заземлення – це з'єднання між електричним обладнанням і землею, завдяки якому утворюється нейтральна точка. Вона необхідна для захисту життя і здоров'я людини за рахунок передачі надлишку електроструму з мережі у ґрунт.

Крім того, заземлення сприяє створенню точки відліку для вимірювання електричного потенціалу в системі – це важливо для забезпечення правильної роботи обладнання, а також виявлення та усунення несправностей.

Початкові дані для розрахунку захисного заземлення:

- тип верхнього шару ґрунту – чорнозем,
- тип нижнього шару ґрунту – глина (питомий опір $\rho_2 = 40$ Ом·м),
- умовна товщина верхнього шару ґрунту: Н=0,5 м.
- відстань між вертикальними заземлювачами А=1,7 м.
- глибина закладення горизонтального контура заземлення t=0,6 м.
- опір заземлювача, який нормується: $R_{ЗН} = 4$ Ом.

Необхідно визначити кількість вертикальних заземлювачів та довжину металевої смуги.

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток 80х50х6 мм, (згідно з ДСТУ 8769:2018 «Кутики сталеві гарячекатані нерівнополичні. Сортамент») довжиною L=1,7 м., та горизонтальний електрод – металева смуга з перетином 60·5 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – у ряд (рис. 8.1).

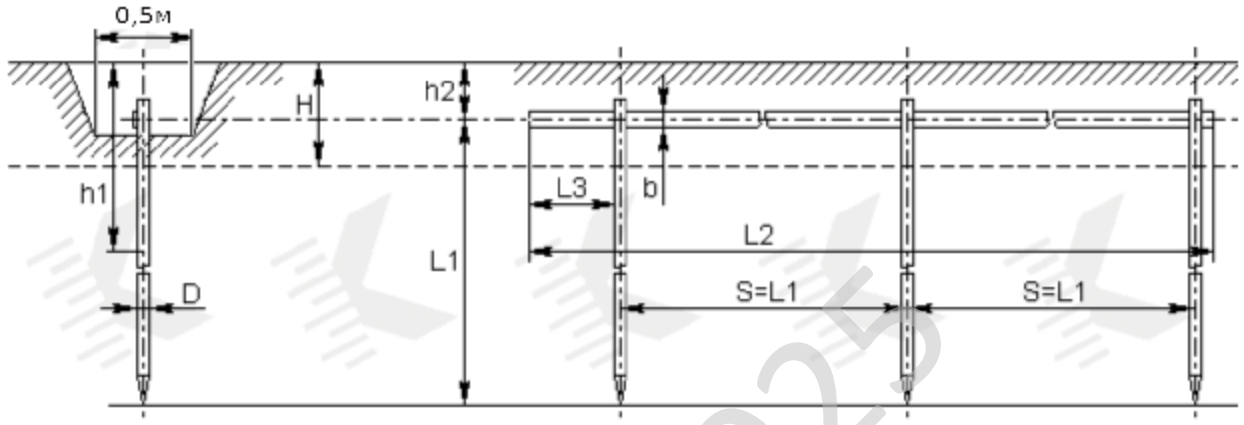

Рисунок 8.1 – Схема штучного заземлення

Розрахунок проведемо за допустимим опором розтіканню струму заземлювача.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t + L/2 = 0,6 +1,7/2 = 1,45 \text{ м.}$$

Розрахунковий питомий опір ґрунту (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі ґрунту):

$$\rho = \psi \rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де

ψ = 1,36 – табличне значення коефіцієнта сезонності для відповідної кліматичної зони у багатошаровому ґрунті [12];

ρ_2= 40 Ом·м. – табличне значення питомого опору нижнього шару грунта (глина) [12].

Еквівалентний діаметр вертикальньного електрода (кутка) [12]:

$$D_в=0,95 \cdot K= 0,95 \cdot 65 = 59,85 \text{ мм.} = 0,62 \text{ м.}$$

де К= (80+50)/2=65 мм. – середній розмір полиці металевого кутка.

Відношення A/L=1,7/1,7=1

Опір розтіканню електричного струму одного електроду вертикального заземлювача з урахуванням заглиблення заземлювача [8]:

$$R_O = 0{,}366 \cdot (\rho/L) \cdot [\lg(2L/Dв) + (1/2) \cdot \lg((4T + L)/(4T - L))] =$$

$$= 0{,}366 \cdot (54.5/1.7)[(\lg(2 \cdot 1.7/0.62) + (1/2) \cdot \lg((4 \cdot 1.45 + 1.7)/(4 \cdot 1.45 - 1.7))] = 21.9 \text{ Ом.}$$

Визначаємо коефіцієнт екранування вертикальних електродів Кев=0,8 при орієнтовній кількості вертикальних електродів, яке дорівнює 5 [8].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без урахування горизонтального заземлювача), при $R_{ЗН}$ = 4 Ом:

$$N = R_O / (Кев \cdot R_{ЗН}) = 21.9 / (0.8 \cdot 4) = 6.85 \approx 7 \text{ шт.}$$

Визначаємо ловжину з'єднуючої смуги:

$$L_П = 1{,}05 \cdot A \cdot N = 1{,}05 \cdot 1{,}7 \cdot 7 = 12{,}22 \approx 12 \text{ м.}$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору ґрунту $K_П$ [8]:

$$R_П = 0{,}366(\rho \cdot K_П / L_П) \lg(2 \cdot L_П^2 / (B \cdot t)) =$$

$$= 0{,}366 \cdot (40 \cdot 5/12) \cdot \lg((2 \cdot 12^2)/(0.06 \cdot 0.6)) = 23.5 \text{ Ом.}$$

де $K_П$=5 – табличне значення кліматичного коефіцієнта питомого опору ґрунту для відповідної кліматичної зони для з'єднуючої полоси [8]:

B = 60 мм = 0.06 м.- ширина з'єднуючої смуги (задана).

Загальний опір розтіканню електричного струму заземлювача [8]:

$$R = (R_O \cdot R_П)/(R_O \cdot \eta_П + N \cdot R_П \cdot Кев) =$$

$$= (21.9 \cdot 23.5)/(21.9 \cdot 0.75 + 9 \cdot 23.5 \cdot 0.8) = 3.48 \text{ Ом.}$$

де $\eta_П$ = 0,75 – табличне значення коефіцієнта екранування з'єднуючої полоси [8].

Умова $R \leq R_{ЗН}$ виконується ($3.48 \leq 4$).

Оскільки при 7 вертикальних електродах R суттєво менше $R_{ЗН}$, зменшимо кількість вертикальних електродів N до 6 і виконаємо перерахунок. У результаті остаточно отримали: R =3,98 Ом. при кількості вертикальних електродів N=6.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

– Досліджена система емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

– На основі отриманих результатів досліджень створена програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CAST-128.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бершадський О.С. Дослідження та програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей // Збірник праць молодих науковців ЦНТУ. – Вип. 15. – Кропивницький: ЦНТУ, 2025.

2. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

3. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.

4. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.

5. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

6. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

7. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

8. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

9. Вінтенко Б., Смірнов О., Миронець І., Смірнова Т., Смірнов С. «Імітаційна модель шляхів вхідних даних комп'ютерної інтелектуальної системи підтримки оператора енергоблоку АЕС». *Комбінаторні конфігурації та їхні застосування: Матеріали XXVII Міжнародного науково-практичного семінару, присвяченого 125-річчю Національного університету «Запорізька політехніка»* (Запоріжжя-Кропивницький-Київ, 4-6 червня 2025 р.). Запоріжжя: НУ

«Запорізька політехніка», 2025. С.82-91.

10. Al-Azzeh, J., Ayyoub, B., Mesleh, A., Smirnova, T., Gnatyuk, S., Drieiev, O., Smirnov, O., Dorenskyi, O. «Cloud-Based Information System for Evaluating Caverns in the Process of Blasting Metal Surfaces of Details». *International Review on Modelling and Simulations* 18 (1), 2025. pp. 32-42.

11. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В., Коваленко О.В., Мацуй А.М. «Модель шляхів отримання вхідних даних комп'ютерної інтелектуальної системи підтримки оперативного персоналу АЕС». *Центральноукраїнський науковий вісник. Технічні науки.* 2025. Вип. 11(42), ч. II. С.52-62.

12. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В. «Методи забезпечення відмовостійкості інтелектуальних систем підтримки оператора». *VIII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології",* м. Кропивницький. 24-25 квітня 2025 р. – Кропивницький: ЦНТУ. – 2025. – С. 44-46.

13. Смірнов, О.А., Константинова, Л.В., Конопліцька-Слободенюк, О.К., Козірова, Н.В, Якименко, Н.М., Доренський, О.П., Буравченко, К.О. «Дослідження інструментів штучного інтелекту для роботи з базами даних та аналізу даних». *Кібербезпека: освіта, наука, техніка.* 2025. №3(27), С. 429–448.)

14. Smirnov O., Fedorov E., Neskorodieva A., Neskorodieva T. «Intellectual Classification method of Gymnastic Elements Based on Combinations of Descriptive and Generative Approache». *CEUR Workshop Proceedings* Volume 3664, 2024, Pages 11-23.

15. Вінтенко, Б., Миронець, І., Смірнов, О., Коваленко, А., Конопліцька-Слободенюк, О., Смірнова, Т., Константинова, Л. «Дослідження застосування систем підтримки оперативного персоналу об'єкту критичної інфраструктури при керуванні енергоблоком АЕС з реактором типу ВВЕР-1000». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 6-26.

16. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

17. Kuznetsov O., Ilchenko O., Kryvinska N., Buravchenko K., Smirnov O., Savchenko Iu. «An Empirical Assessment of Leading Blockchain Financial Services». *2023 IEEE 1st Ukrainian Distributed Ledger Technology Forum (UADLTF),* Kyiv, Ukraine, 2023, pp. 1-6,

18. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

19. Malyukov V., Bebeshko B., Lakhno V., Smirnov O., Malyukova I., Mohylnyi H. «Managing the Purchase-Sale Process of Digital Currencies Under Fuzzy Conditions». *Lecture Notes in Networks and Systems,* 2023, 729 LNNS, pp. 104–112.

20. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

21. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

22. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

23. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023,

вип. 3(73), С. 155-166.

24. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

25. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

26. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

27. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

28. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

29. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

30. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology

(PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

31. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

32. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

33. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

34. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

35. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.

36. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

37. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

38. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise

immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

39. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

40. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

41. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

42. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

43. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

44. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

45. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

46. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

47. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

48. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

49. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

50. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

51. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for Dom Xss vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp.. Georgia. Tbilisi: SCSA – 2018.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 30-13 від 22.08.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати маркетингове та економічне обґрунтування ІТ-проєкту з урахуванням цін на 3 вересня 2025 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 107 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Маркетингове та економічне обґрунтування ІТ-проєкту.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 02.12.2025 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 22.12.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Коваленко О.В.

Дослідження та програмна реалізація системи емулятора перетворення логічної адреси в лінійну та лінійної у фізичну для навчальних цілей

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 15

Літера: РП

Кропивницький – 2025 року

```
import math
import random
import unittest
from dataclasses import dataclass
from typing import Dict
from typing import List
from typing import Optional
from typing import Tuple


class LogicalAddressError(Exception):
    pass


class SegmentTranslationError(Exception):
    pass


class PagingTranslationError(Exception):
    pass


class PhysicalMemoryError(Exception):
    pass


@dataclass
class SegmentSelector:
    value: int
    index: int
    table_indicator: int
    requested_privilege_level: int

    @staticmethod
    def from_raw(selector_value: int) -> "SegmentSelector":
        if selector_value < 0:
            raise LogicalAddressError("selector must be non negative")
        index = selector_value >> 3
        table_indicator = (selector_value >> 2) & 1
        requested_privilege_level = selector_value & 3
        result = SegmentSelector(
            value=selector_value,
            index=index,
            table_indicator=table_indicator,
            requested_privilege_level=requested_privilege_level,
        )
        return result

    def to_raw(self) -> int:
        raw_value = 0
        raw_value = raw_value | (self.index << 3)
        raw_value = raw_value | ((self.table_indicator & 1) << 2)
        raw_value = raw_value | (self.requested_privilege_level & 3)
        return raw_value


@dataclass
class LogicalAddress:
    selector_value: int
    offset: int

    def validate(self) -> None:
```

```
        if self.selector_value < 0:
            raise LogicalAddressError("selector must be non negative")
        if self.offset < 0:
            raise LogicalAddressError("offset must be non negative")


@dataclass
class SegmentDescriptor:
    base: int
    limit: int
    present: bool
    writable: bool
    executable: bool
    user_accessible: bool

    def contains_offset(self, offset: int) -> bool:
        if offset < 0:
            return False
        if offset > self.limit:
            return False
        return True

    def compute_linear(self, offset: int) -> int:
        if not self.contains_offset(offset):
            raise SegmentTranslationError("offset out of segment range")
        linear = self.base + offset
        return linear


class DescriptorTable:
    def __init__(self) -> None:
        self.descriptors: Dict[int, SegmentDescriptor] = {}

    def set_descriptor(self, index: int, descriptor: SegmentDescriptor) -> None:
        if index < 0:
            raise SegmentTranslationError("descriptor index must be non
negative")
        self.descriptors[index] = descriptor

    def get_descriptor_by_selector(self, selector: SegmentSelector) ->
SegmentDescriptor:
        if selector.table_indicator != 0:
            raise SegmentTranslationError("this emulator supports only single
table")
        index = selector.index
        if index not in self.descriptors:
            raise SegmentTranslationError("descriptor index not present")
        descriptor = self.descriptors[index]
        if not descriptor.present:
            raise SegmentTranslationError("segment not present")
        return descriptor

    def descriptor_count(self) -> int:
        count = len(self.descriptors)
        return count


@dataclass
class PageTableEntry:
    frame_number: int
    present: bool
    writable: bool
    user_accessible: bool
```

```
    def physical_address_for_offset(self, offset: int, page_size: int) -> int:
        if not self.present:
            raise PagingTranslationError("page not present")
        if offset < 0 or offset >= page_size:
            raise PagingTranslationError("offset out of page range")
        physical_address = self.frame_number * page_size + offset
        return physical_address


@dataclass
class PageDirectoryEntry:
    table: "PageTable"
    present: bool
    writable: bool
    user_accessible: bool


class PageTable:
    def __init__(self) -> None:
        self.entries: Dict[int, PageTableEntry] = {}

    def set_entry(self, index: int, entry: PageTableEntry) -> None:
        if index < 0 or index >= 1024:
            raise PagingTranslationError("page table index out of range")
        self.entries[index] = entry

    def get_entry(self, index: int) -> PageTableEntry:
        if index not in self.entries:
            raise PagingTranslationError("page table entry not present")
        entry = self.entries[index]
        if not entry.present:
            raise PagingTranslationError("page table entry not marked present")
        return entry

    def entry_count(self) -> int:
        count = len(self.entries)
        return count


class PageDirectory:
    def __init__(self) -> None:
        self.entries: Dict[int, PageDirectoryEntry] = {}

    def set_entry(self, index: int, entry: PageDirectoryEntry) -> None:
        if index < 0 or index >= 1024:
            raise PagingTranslationError("page directory index out of range")
        self.entries[index] = entry

    def get_entry(self, index: int) -> PageDirectoryEntry:
        if index not in self.entries:
            raise PagingTranslationError("page directory entry not present")
        entry = self.entries[index]
        if not entry.present:
            raise PagingTranslationError("page directory entry not marked
present")
        return entry

    def entry_count(self) -> int:
        count = len(self.entries)
        return count
```

```
@dataclass
class PagingConfig:
    page_size: int
    address_width: int

    def offset_mask(self) -> int:
        mask = self.page_size - 1
        return mask

    def directory_shift(self) -> int:
        shift = 22
        return shift

    def table_shift(self) -> int:
        shift = 12
        return shift

    def directory_mask(self) -> int:
        mask = 0x3FF
        return mask

    def table_mask(self) -> int:
        mask = 0x3FF
        return mask


class PagingUnit:
    def __init__(self, config: PagingConfig, directory: PageDirectory) -> None:
        self.config = config
        self.directory = directory

    def split_linear_address(self, linear: int) -> Tuple[int, int, int]:
        if linear < 0:
            raise PagingTranslationError("linear address must be non negative")
        offset_mask_value = self.config.offset_mask()
        offset = linear & offset_mask_value
        directory_shift_value = self.config.directory_shift()
        table_shift_value = self.config.table_shift()
        directory_index = (linear >> directory_shift_value) &
self.config.directory_mask()
        table_index = (linear >> table_shift_value) & self.config.table_mask()
        result = (directory_index, table_index, offset)
        return result

    def translate_linear_to_physical(self, linear: int) -> int:
        directory_index, table_index, offset = self.split_linear_address(linear)
        directory_entry = self.directory.get_entry(directory_index)
        table = directory_entry.table
        table_entry = table.get_entry(table_index)
        physical = table_entry.physical_address_for_offset(offset,
self.config.page_size)
        return physical


class PhysicalMemory:
    def __init__(self, frame_count: int, page_size: int) -> None:
        if frame_count <= 0:
            raise PhysicalMemoryError("frame count must be positive")
        if page_size <= 0:
            raise PhysicalMemoryError("page size must be positive")
        self.frame_count = frame_count
        self.page_size = page_size
        self.frames: Dict[int, bytearray] = {}
```

```
        self._init_frames()

    def _init_frames(self) -> None:
        index = 0
        while index < self.frame_count:
            buffer = bytearray(self.page_size)
            self.frames[index] = buffer
            index = index + 1

    def read_byte(self, physical_address: int) -> int:
        if physical_address < 0:
            raise PhysicalMemoryError("physical address must be non negative")
        frame_index = physical_address // self.page_size
        offset = physical_address % self.page_size
        if frame_index not in self.frames:
            raise PhysicalMemoryError("frame not present")
        frame = self.frames[frame_index]
        value = frame[offset]
        return value

    def write_byte(self, physical_address: int, value: int) -> None:
        if physical_address < 0:
            raise PhysicalMemoryError("physical address must be non negative")
        if value < 0 or value > 255:
            raise PhysicalMemoryError("byte value out of range")
        frame_index = physical_address // self.page_size
        offset = physical_address % self.page_size
        if frame_index not in self.frames:
            raise PhysicalMemoryError("frame not present")
        frame = self.frames[frame_index]
        frame[offset] = value

    def frame_usage_bitmap(self) -> List[bool]:
        bitmap: List[bool] = []
        index = 0
        while index < self.frame_count:
            present = index in self.frames
            bitmap.append(present)
            index = index + 1
        return bitmap

    def frame_size_bytes(self) -> int:
        size = self.page_size
        return size


@dataclass
class TranslationTrace:
    selector_value: int
    logical_offset: int
    segment_base: int
    segment_limit: int
    linear_address: int
    directory_index: int
    table_index: int
    page_offset: int
    frame_number: int
    physical_address: int


class AddressTranslator:
    def __init__(
        self,
```

```
        descriptor_table: DescriptorTable,
        paging_unit: PagingUnit,
        memory: PhysicalMemory,
    ) -> None:
        self.descriptor_table = descriptor_table
        self.paging_unit = paging_unit
        self.memory = memory

    def logical_to_linear(self, logical: LogicalAddress) -> int:
        logical.validate()
        selector = SegmentSelector.from_raw(logical.selector_value)
        descriptor = self.descriptor_table.get_descriptor_by_selector(selector)
        linear = descriptor.compute_linear(logical.offset)
        return linear

    def logical_to_physical(self, logical: LogicalAddress) -> int:
        linear = self.logical_to_linear(logical)
        physical = self.paging_unit.translate_linear_to_physical(linear)
        return physical

    def translate_with_trace(self, logical: LogicalAddress) -> TranslationTrace:
        logical.validate()
        selector = SegmentSelector.from_raw(logical.selector_value)
        descriptor = self.descriptor_table.get_descriptor_by_selector(selector)
        linear_address = descriptor.compute_linear(logical.offset)
        directory_index, table_index, offset =
self.paging_unit.split_linear_address(linear_address)
        directory_entry = self.paging_unit.directory.get_entry(directory_index)
        table = directory_entry.table
        table_entry = table.get_entry(table_index)
        frame_number = table_entry.frame_number
        physical_address = table_entry.physical_address_for_offset(offset,
self.paging_unit.config.page_size)
        trace = TranslationTrace(
            selector_value=logical.selector_value,
            logical_offset=logical.offset,
            segment_base=descriptor.base,
            segment_limit=descriptor.limit,
            linear_address=linear_address,
            directory_index=directory_index,
            table_index=table_index,
            page_offset=offset,
            frame_number=frame_number,
            physical_address=physical_address,
        )
        return trace

    def read_byte_from_logical(self, logical: LogicalAddress) -> int:
        physical = self.logical_to_physical(logical)
        value = self.memory.read_byte(physical)
        return value

    def write_byte_to_logical(self, logical: LogicalAddress, value: int) ->
None:
        physical = self.logical_to_physical(logical)
        self.memory.write_byte(physical, value)


class DemoConfigurationBuilder:
    def __init__(self) -> None:
        self.descriptor_table = DescriptorTable()
        self.directory = PageDirectory()
        self.page_size = 4096
```

```
        self.memory = PhysicalMemory(frame_count=64, page_size=self.page_size)
        self.paging_config = PagingConfig(page_size=self.page_size,
address_width=32)
        self.paging_unit = PagingUnit(self.paging_config, self.directory)
        self.translator = AddressTranslator(self.descriptor_table,
self.paging_unit, self.memory)

    def build_segment(self, index: int, base: int, limit: int) -> None:
        descriptor = SegmentDescriptor(
            base=base,
            limit=limit,
            present=True,
            writable=True,
            executable=False,
            user_accessible=True,
        )
        self.descriptor_table.set_descriptor(index, descriptor)

    def build_single_page_mapping(self, linear_base: int, frame_number: int) ->
None:
        directory_index, table_index, offset =
self.paging_unit.split_linear_address(linear_base)
        if directory_index not in self.directory.entries:
            table = PageTable()
            directory_entry = PageDirectoryEntry(
                table=table,
                present=True,
                writable=True,
                user_accessible=True,
            )
            self.directory.set_entry(directory_index, directory_entry)
        else:
            directory_entry = self.directory.get_entry(directory_index)
            table = directory_entry.table
        entry = PageTableEntry(
            frame_number=frame_number,
            present=True,
            writable=True,
            user_accessible=True,
        )
        table.set_entry(table_index, entry)

    def build_identity_mapping_range(self, linear_start: int, length: int) ->
None:
        page_size = self.page_size
        pages = (length + page_size - 1) // page_size
        page_index = 0
        while page_index < pages:
            linear_base = linear_start + page_index * page_size
            frame_number = (linear_base // page_size) % self.memory.frame_count
            self.build_single_page_mapping(linear_base, frame_number)
            page_index = page_index + 1

    def build_demo_segments_and_pages(self) -> None:
        self.build_segment(index=1, base=0x00400000, limit=0x000FFFFF)
        self.build_segment(index=2, base=0x00800000, limit=0x000FFFFF)
        self.build_identity_mapping_range(0x00400000, 0x00100000)
        self.build_identity_mapping_range(0x00800000, 0x00100000)

    def get_translator(self) -> AddressTranslator:
        return self.translator

    def get_memory(self) -> PhysicalMemory:
```

```
        return self.memory

    def get_descriptor_table(self) -> DescriptorTable:
        return self.descriptor_table

    def get_page_directory(self) -> PageDirectory:
        return self.directory


class AddressFormatHelper:
    def __init__(self) -> None:
        pass

    def format_logical(self, logical: LogicalAddress) -> str:
        selector = logical.selector_value
        offset = logical.offset
        text = "sel " + str(selector) + " off " + str(offset)
        return text

    def format_trace(self, trace: TranslationTrace) -> str:
        parts: List[str] = []
        parts.append("selector " + str(trace.selector_value))
        parts.append("logical_offset " + str(trace.logical_offset))
        parts.append("segment_base " + str(trace.segment_base))
        parts.append("segment_limit " + str(trace.segment_limit))
        parts.append("linear_address " + str(trace.linear_address))
        parts.append("directory_index " + str(trace.directory_index))
        parts.append("table_index " + str(trace.table_index))
        parts.append("page_offset " + str(trace.page_offset))
        parts.append("frame_number " + str(trace.frame_number))
        parts.append("physical_address " + str(trace.physical_address))
        text = " | ".join(parts)
        return text


class TeachingScenario:
    def __init__(self, builder: DemoConfigurationBuilder) -> None:
        self.builder = builder
        self.builder.build_demo_segments_and_pages()
        self.translator = self.builder.get_translator()
        self.memory = self.builder.get_memory()
        self.helper = AddressFormatHelper()

    def write_pattern_to_segment(self, selector_value: int, base_offset: int,
count: int, pattern: int) -> None:
        index = 0
        while index < count:
            logical = LogicalAddress(selector_value=selector_value,
offset=base_offset + index)
            value = (pattern + index) & 0xFF
            self.translator.write_byte_to_logical(logical, value)
            index = index + 1

    def verify_pattern_in_segment(self, selector_value: int, base_offset: int,
count: int, pattern: int) -> bool:
        index = 0
        while index < count:
            logical = LogicalAddress(selector_value=selector_value,
offset=base_offset + index)
            expected = (pattern + index) & 0xFF
            actual = self.translator.read_byte_from_logical(logical)
            if actual != expected:
                return False
```

```
            index = index + 1
        return True

    def run_demo(self) -> List[str]:
        selector_code = 1 << 3
        selector_data = 2 << 3
        self.write_pattern_to_segment(selector_data, base_offset=0, count=64,
pattern=42)
        ok = self.verify_pattern_in_segment(selector_data, base_offset=0,
count=64, pattern=42)
        messages: List[str] = []
        if ok:
            messages.append("pattern check success")
        else:
            messages.append("pattern check failure")
        logical_examples: List[LogicalAddress] = []
        logical_examples.append(LogicalAddress(selector_value=selector_code,
offset=16))
        logical_examples.append(LogicalAddress(selector_value=selector_code,
offset=128))
        logical_examples.append(LogicalAddress(selector_value=selector_data,
offset=256))
        for logical in logical_examples:
            trace = self.translator.translate_with_trace(logical)
            description = self.helper.format_trace(trace)
            messages.append(description)
        return messages


# Settings structures for configurable emulator profiles
@dataclass
class SegmentConfig:
    index: int
    base: int
    limit: int


@dataclass
class MappingRangeConfig:
    linear_start: int
    length: int


@dataclass
class EmulatorSettings:
    name: str
    frame_count: int
    page_size: int
    segments: List[SegmentConfig]
    mappings: List[MappingRangeConfig]


class SettingsManager:
    def __init__(self) -> None:
        self.profiles: Dict[str, EmulatorSettings] = {}
        self._init_default_profiles()

    def _init_default_profiles(self) -> None:
        basic_segments: List[SegmentConfig] = []
        basic_segments.append(SegmentConfig(index=1, base=0x00400000,
limit=0x0007FFFF))
        basic_segments.append(SegmentConfig(index=2, base=0x00800000,
limit=0x0007FFFF))
```

```
        basic_mappings: List[MappingRangeConfig] = []
        basic_mappings.append(MappingRangeConfig(linear_start=0x00400000,
length=0x00080000))
        basic_mappings.append(MappingRangeConfig(linear_start=0x00800000,
length=0x00080000))
        basic_settings = EmulatorSettings(
            name="basic",
            frame_count=64,
            page_size=4096,
            segments=basic_segments,
            mappings=basic_mappings,
        )
        self.profiles[basic_settings.name] = basic_settings
        extended_segments: List[SegmentConfig] = []
        extended_segments.append(SegmentConfig(index=3, base=0x00C00000,
limit=0x000FFFFF))
        extended_segments.append(SegmentConfig(index=4, base=0x01000000,
limit=0x000FFFFF))
        extended_mappings: List[MappingRangeConfig] = []
        extended_mappings.append(MappingRangeConfig(linear_start=0x00C00000,
length=0x00100000))
        extended_mappings.append(MappingRangeConfig(linear_start=0x01000000,
length=0x00100000))
        extended_settings = EmulatorSettings(
            name="extended",
            frame_count=128,
            page_size=4096,
            segments=extended_segments,
            mappings=extended_mappings,
        )
        self.profiles[extended_settings.name] = extended_settings

    def list_profiles(self) -> List[str]:
        names = sorted(self.profiles.keys())
        return names

    def get_profile(self, name: str) -> EmulatorSettings:
        if name not in self.profiles:
            raise KeyError("profile not found")
        settings = self.profiles[name]
        return settings

    def create_builder_for_profile(self, name: str) -> DemoConfigurationBuilder:
        settings = self.get_profile(name)
        builder = DemoConfigurationBuilder()
        builder.memory = PhysicalMemory(frame_count=settings.frame_count,
page_size=settings.page_size)
        builder.page_size = settings.page_size
        builder.paging_config = PagingConfig(page_size=settings.page_size,
address_width=32)
        builder.paging_unit = PagingUnit(builder.paging_config,
builder.directory)
        builder.translator = AddressTranslator(builder.descriptor_table,
builder.paging_unit, builder.memory)
        for segment in settings.segments:
            builder.build_segment(index=segment.index, base=segment.base,
limit=segment.limit)
        for mapping in settings.mappings:

builder.build_identity_mapping_range(linear_start=mapping.linear_start,
length=mapping.length)
        return builder
```

```
# Exercise generation helpers for teaching tasks
@dataclass
class ExerciseItem:
    logical: LogicalAddress
    trace: TranslationTrace


class ExerciseGenerator:
    def __init__(self, builder: DemoConfigurationBuilder) -> None:
        self.builder = builder
        self.translator = builder.get_translator()
        self.random_instance = random.Random()

    def random_logical_in_segment(self, segment_index: int) -> LogicalAddress:
        descriptors = self.builder.get_descriptor_table().descriptors
        if segment_index not in descriptors:
            raise SegmentTranslationError("segment not configured")
        descriptor = descriptors[segment_index]
        if descriptor.limit < 0:
            raise SegmentTranslationError("invalid segment limit")
        offset = self.random_instance.randint(0, descriptor.limit)
        selector_value = segment_index << 3
        logical = LogicalAddress(selector_value=selector_value, offset=offset)
        return logical

    def build_exercises(self, count: int, segment_index: int) ->
List[ExerciseItem]:
        items: List[ExerciseItem] = []
        index = 0
        while index < count:
            logical = self.random_logical_in_segment(segment_index)
            trace = self.translator.translate_with_trace(logical)
            item = ExerciseItem(logical=logical, trace=trace)
            items.append(item)
            index = index + 1
        return items


class LogicalToLinearTests(unittest.TestCase):
    def setUp(self) -> None:
        self.builder = DemoConfigurationBuilder()
        descriptor = SegmentDescriptor(
            base=0x00100000,
            limit=0x0000FFFF,
            present=True,
            writable=True,
            executable=False,
            user_accessible=True,
        )
        self.builder.descriptor_table.set_descriptor(3, descriptor)
        self.builder.build_single_page_mapping(0x00100000, 1)
        self.translator = self.builder.get_translator()

    def test_logical_to_linear_inside_segment(self) -> None:
        selector_value = 3 << 3
        logical = LogicalAddress(selector_value=selector_value, offset=0x10)
        linear = self.translator.logical_to_linear(logical)
        expected_linear = 0x00100000 + 0x10
        self.assertEqual(linear, expected_linear)
```

```
    def test_logical_offset_outside_segment(self) -> None:
        selector_value = 3 << 3
        logical = LogicalAddress(selector_value=selector_value,
offset=0x0001FFFF)
        with self.assertRaises(SegmentTranslationError):
            self.translator.logical_to_linear(logical)

    def test_invalid_selector_negative(self) -> None:
        logical = LogicalAddress(selector_value=-1, offset=0)
        with self.assertRaises(LogicalAddressError):
            self.translator.logical_to_linear(logical)

    def test_invalid_offset_negative(self) -> None:
        selector_value = 3 << 3
        logical = LogicalAddress(selector_value=selector_value, offset=-5)
        with self.assertRaises(LogicalAddressError):
            self.translator.logical_to_linear(logical)


class PagingUnitTests(unittest.TestCase):
    def setUp(self) -> None:
        self.directory = PageDirectory()
        self.page_size = 4096
        self.config = PagingConfig(page_size=self.page_size, address_width=32)
        self.paging = PagingUnit(self.config, self.directory)
        table = PageTable()
        entry = PageTableEntry(
            frame_number=5,
            present=True,
            writable=True,
            user_accessible=True,
        )
        table.set_entry(0, entry)
        directory_entry = PageDirectoryEntry(
            table=table,
            present=True,
            writable=True,
            user_accessible=True,
        )
        self.directory.set_entry(0, directory_entry)

    def test_translation_of_first_page(self) -> None:
        linear = 0
        physical = self.paging.translate_linear_to_physical(linear)
        expected = 5 * self.page_size
        self.assertEqual(physical, expected)

    def test_translation_with_offset(self) -> None:
        linear = 123
        physical = self.paging.translate_linear_to_physical(linear)
        expected = 5 * self.page_size + 123
        self.assertEqual(physical, expected)

    def test_missing_directory_entry(self) -> None:
        linear = 1 << 22
        with self.assertRaises(PagingTranslationError):
            self.paging.translate_linear_to_physical(linear)

    def test_missing_table_entry(self) -> None:
        linear = 1 << 12
        with self.assertRaises(PagingTranslationError):
            self.paging.translate_linear_to_physical(linear)
```

```
class PhysicalMemoryTests(unittest.TestCase):
    def setUp(self) -> None:
        self.memory = PhysicalMemory(frame_count=4, page_size=256)

    def test_write_and_read_byte(self) -> None:
        address = 10
        value = 123
        self.memory.write_byte(address, value)
        read_value = self.memory.read_byte(address)
        self.assertEqual(read_value, value)

    def test_invalid_negative_address(self) -> None:
        with self.assertRaises(PhysicalMemoryError):
            self.memory.write_byte(-1, 0)

    def test_invalid_byte_value(self) -> None:
        with self.assertRaises(PhysicalMemoryError):
            self.memory.write_byte(0, 300)

    def test_frame_usage_bitmap_length(self) -> None:
        bitmap = self.memory.frame_usage_bitmap()
        self.assertEqual(len(bitmap), 4)


class FullTranslationTests(unittest.TestCase):
    def setUp(self) -> None:
        self.builder = DemoConfigurationBuilder()
        self.builder.build_demo_segments_and_pages()
        self.translator = self.builder.get_translator()
        self.memory = self.builder.get_memory()

    def test_write_and_read_via_logical(self) -> None:
        selector_value = 1 << 3
        logical = LogicalAddress(selector_value=selector_value, offset=32)
        value = 77
        self.translator.write_byte_to_logical(logical, value)
        read_back = self.translator.read_byte_from_logical(logical)
        self.assertEqual(read_back, value)

    def test_trace_values_consistent(self) -> None:
        selector_value = 1 << 3
        logical = LogicalAddress(selector_value=selector_value, offset=64)
        trace = self.translator.translate_with_trace(logical)
        linear_from_translator = self.translator.logical_to_linear(logical)
        self.assertEqual(trace.linear_address, linear_from_translator)
        physical_from_translator = self.translator.logical_to_physical(logical)
        self.assertEqual(trace.physical_address, physical_from_translator)

    def test_demo_teaching_scenario(self) -> None:
        scenario = TeachingScenario(self.builder)
        messages = scenario.run_demo()
        self.assertGreaterEqual(len(messages), 2)


class SettingsManagerTests(unittest.TestCase):
    def setUp(self) -> None:
        self.manager = SettingsManager()
```

```
    def test_list_profiles_not_empty(self) -> None:
        names = self.manager.list_profiles()
        self.assertGreater(len(names), 0)

    def test_create_builder_for_basic_profile(self) -> None:
        builder = self.manager.create_builder_for_profile("basic")
        descriptor_table = builder.get_descriptor_table()
        self.assertGreater(descriptor_table.descriptor_count(), 0)

    def test_invalid_profile_raises(self) -> None:
        with self.assertRaises(KeyError):
            self.manager.create_builder_for_profile("missing_profile")


class ExerciseGeneratorTests(unittest.TestCase):
    def setUp(self) -> None:
        manager = SettingsManager()
        builder = manager.create_builder_for_profile("basic")
        self.builder = builder
        self.generator = ExerciseGenerator(builder)

    def test_build_exercises_non_empty(self) -> None:
        items = self.generator.build_exercises(count=5, segment_index=1)
        self.assertEqual(len(items), 5)
        for item in items:
            physical_from_trace = item.trace.physical_address
            physical_from_translator =
self.builder.get_translator().logical_to_physical(item.logical)
            self.assertEqual(physical_from_trace, physical_from_translator)


if __name__ == "__main__":
    unittest.main()
```