**Резюме**

до кваліфікаційної роботи

студента групи 4-Е-33

Чесного Владислава Сергійовича

Тема: “ Проєктування віртуальної лабораторії для візуалізації архітектури комп’ютерних систем”

Метою кваліфікаційної роботи є створення інтерактивного навчального засобу, який допомагає студентам у доступній формі вивчати архітектуру комп’ютера, його складові елементи, принципи функціонування та базові обчислювальні поняття.

Результатом проєкту стала віртуальна лабораторія, що включає чотири модулі:

* Компоненти комп’ютера — дозволяє збирати ПК з окремих деталей, отримуючи пояснення про кожну з них;
* Периферійні пристрої — ілюструє підключення зовнішніх пристроїв до відповідних портів і пояснює їх призначення;
* Конвертування систем числення — забезпечує перетворення чисел між різними системами числення;
* Машинний код і асемблер — демонструє взаємозв’язок між низькорівневим кодом і його асемблерним представленням.

Розроблена віртуальна лабораторія може бути застосована у процесі навчання з інформатики та комп’ютерної інженерії як допоміжний засіб, що підвищує якість засвоєння матеріалу і сприяє формуванню практичних навичок.

У результаті обчислень повна собівартість створеної веб-лабораторії становить 8589,45 грн., а відпускна ціна складає 12368,81 грн.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «СУМСЬКИЙ ФАХОВИЙ КОЛЕДЖ НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ХАРЧОВИХ ТЕХНОЛОГІЙ»

ЦИКЛОВА КОМІСІЯ КОМП’ЮТЕРНОЇ ІНЖЕНЕРІЇ

|  |  |
| --- | --- |
| **«До захисту в ЕК»** | **«До захисту допущено»** |
| Завідувачка технічного відділення | Голова циклової комісії |
| \_\_\_\_\_\_\_\_\_ Ірина РЯБІЧЕНКО \_  (підпис) (Власне ім’я ПРІЗВИЩЕ) | \_\_\_\_\_\_\_\_\_ \_Валентина СТОМА  (підпис)  (Власне ім’я ПРІЗВИЩЕ) |
| «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 р. | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 р. |

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

***до кваліфікаційної роботи фахового молодшого бакалавра***

зі спеціальності *123 Комп’ютерна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(шифр та назва спеціальності)

освітньо-професійної програми «*Комп’ютерна інженерія»*

на тему: “Проєктування віртуальної лабораторії для візуалізації архітектури комп’ютерних систем”**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Виконав: студент IV курсу, групи Е-33 Чесной В.С.

(прізвище та ініціали)

Керівник Стома Валентина Миколаївна \_\_\_\_\_\_\_\_\_\_\_\_

(Власне ім’я ПРІЗВИЩЕ) (підпис)

Консультант Тетяна КРУПСЬКА \_\_\_\_\_\_\_\_\_\_\_

(Власне ім’я ПРІЗВИЩЕ) (підпис)

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

(Власне ім’я ПРІЗВИЩЕ) (підпис)

|  |  |  |
| --- | --- | --- |
|  |  | Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань. |
|  |  | Здобувач освіти \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) |

Суми – 2025 р.

ВСП «СУМСЬКИЙ ФАХОВИЙ КОЛЕДЖ

НАЦІОНАЛЬНОГО  УНІВЕРСИТЕТУ ХАРЧОВИХ ТЕХНОЛОГІЙ»

Відділення технічне

Циклова  комісія комп’ютерної інженерії

Освітньо-професійний ступінь *фаховий молодший бакалавр*

Галузь знань *12 Інформаційні технології\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Спеціальність  *123 Комп’ютерна інженерія*

# (шифр і назва)

Освітньо-професійна програма  *Комп’ютерна інженерія*

# (назва)

# 

# 

# ЗАТВЕРДЖУЮ

**Голова  циклової комісії**

**\_\_\_\_\_\_\_\_**Валентина СТОМА

*“5”    травня  2025  року*

## З  А  В  Д  А  Н  Н  Я

### НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ОСВІТИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Чесного Владислава Сергійовича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я,  по батькові)

1.Тема роботи \_\_“Проєктування віртуальної лабораторії для візуалізації архітектури комп’ютерних систем”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник роботи \_\_\_\_\_\_\_\_\_ Стома Валентина Миколаївна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу освіти від *«5» травня 2025 року*  № 42-с\_\_\_\_\_\_\_

2. Строк подання студентом роботи   *16 червня 2025 року             \_*

3. Вихідні дані до роботи: середовище програмування Visual Studio Code, мова програмування HTML, мова програмування Javascript, мова програмування CSS, віртуальна лабораторія з архітектури комп'ютера\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *вступ, 1 техніко-економічне обґрунтування кваліфікаційної роботи, 2 розробка віртуальної лабораторії для вивчення архітектури комп’ютера, 2.1 проєктування архітектури віртуальної лабораторії, 2.2 структура коду та його оптимізація, 2.3 використання framework vue для візуального програмування, 2.4 тестування та налагодження віртуальної лабораторії, 3 інструкція по роботі з програмою, 4 економічна частина, охорона праці, висновoк, перелік джерел інформації, додатки*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
| *6. Економічна частина* | Крупська Т.М., викладач |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання *«5» травня 2025  рокую*

#### КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів кваліфікаційної  роботи | Строк  виконання етапів роботи | Примітка |
| 1 | Вступ | перший тиждень |  |
| 2 | Техніко-економічне обґрунтування кваліфікаційної роботи | перший тиждень |  |
| 3 | Розробка віртуальної лабораторії для вивчення Архітектури комп’ютера | другий тиждень |  |
| 4 | Проєктування архітектури віртуальної лабораторії | другий тиждень |  |
| 5 | Структура коду та його оптимізація | третій тиждень |  |
| 6 | Використання Framework Vue для візуального програмування | четвертий тиждень |  |
| 7 | Інструкція по роботі з програмою | четвертий тиждень |  |
| 8 | Економічна частина | четвертий тиждень |  |
| 9 | Охорона праці | п’ятий тиждень |  |
| 10 | Висновок | п’ятий тиждень |  |
| 11 | Оформлення пояснювальної записки | п’ятий тиждень |  |
| 12 | Оформлення графічної частини | шостий тиждень |  |
| 13 | Оформлення кваліфікаційної роботи | шостий тиждень |  |
| 14 | Перевірка роботи керівником | шостий тиждень |  |
| 15 | Рецензування та попередній захист | сьомий тиждень |  |

**Здобувач освіти        \_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

( підпис ) (Власне ім`я ПРІЗВИЩЕ)

**Керівник роботи      \_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

( підпис )                    (Власне ім`я ПРІЗВИЩЕ)

ЗМІСТ

|  | ВСТУП……………………………………………………………………… |  |
| --- | --- | --- |
| 1 | ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ…………………………………………………………………….. |  |
| 2 | Розробка віртуальної лабораторії для вивчення Архітектури комп’ютера.……………………………………………………………........ |  |
| 2.1 | Проєктування архітектури віртуальної лабораторії……………………… |  |
| 2.2 | Структура коду та його оптимізація…......................................................... |  |
| 2.3 | Використання Framework Vue для візуального програмування……………………………………………………………... |  |
| 2.4 | Тестування та налагодження віртуальної лабораторії………………………………………………………………….. |  |
| 3 | ІНСТРУКЦІЯ ПО РОБОТІ З ПРОГРАМОЮ……………………………... |  |
| 4 | ЕКОНОМІЧНА ЧАСТИНА……………………………………………….. |  |
| 5 | ОХОРОНА ПРАЦІ………………………………………………………..... |  |
|  | ВИСНОВОК………………………………………………………………... |  |
|  | ПЕРЕЛІК ДЖЕРЕЛ ІНФОРМАЦІЇ………………………………………... |  |

ВСТУП

В сучасних умовах стрімкого розвитку інформаційних технологій все більшого значення набувають інтерактивні методи навчання в галузі комп’ютерної інженерії. Одним із таких методів є віртуальні лабораторії, які дозволяють здобувачам освіти набувати практичних навичок роботи з комп’ютерними системами без потреби в фізичному обладнанні. Віртуальні комп’ютерні лабораторії є необхідним елементом освітнього процесу, оскільки вони гарантують навчання й розвиток здобувачів навіть в умовах дистанційного навчання, використовуючи мінімальний набір доступних гаджетів.

Метою даної кваліфікаційної роботи є розробка віртуальної навчальної лабораторії, призначеної для інтерактивної візуалізації архітектури комп’ютерних систем. Для досягнення поставленої мети в роботі вирішено такі завдання:

1. проаналізувати сучасні підходи до створення віртуальних лабораторій та дослідити архітектуру комп’ютерних систем, включаючи основні компоненти ЕОМ і периферійні пристрої.
2. спроєктувати архітектуру віртуальної лабораторії, обґрунтувати вибір інструментів розробки та реалізувати функціональну частину програмного продукту.
3. провести техніко-економічне обґрунтування розробленого рішення та створити інструкцію користувача для ефективного використання лабораторії.

При виконанні роботи використовувались методи аналізу та синтезу – для визначення вимог до програмного засобу; об’єктно-орієнтоване проєктування – для розробки структури додатка; веб-технології HTML5, CSS, JavaScript  (Vue.js)  – для безпосередньої реалізації інтерактивних модулів;

Практичне значення роботи полягає в створенні віртуальної лабораторії, що може бути використана у навчальному процесі при вивченні дисциплін з архітектури комп’ютерних систем.

1 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Забезпечення якісного засвоєння знань у сфері комп’ютерної інженерії вимагає не лише теоретичної підготовки, а й практичної взаємодії з апаратним забезпеченням. Однак організація традиційних лабораторій, заснованих на використанні реальних комплектуючих, часто супроводжується значними витратами на закупівлю обладнання і його обслуговування, а також потребує наявності спеціально обладнаних приміщень і технічного персоналу. Додатково виникають труднощі, пов’язані з обмеженою кількістю робочих місць, що ускладнює проведення занять для великих груп здобувачів освіти або в умовах дистанційного навчання.

Створення інтерактивного веб-додатка у вигляді віртуальної лабораторії дозволяє вирішити всі вище подані проблеми завдяки імітації основних процесів, які зазвичай відбуваються в реальному середовищі. Користувачі мають можливість досліджувати будову комп’ютера, з’єднувати логічні елементи, досліджувати будова та принципи роботи периферійних пристроїв, працювати з системами числення та виконувати базові операції з машинним кодом. Усе це реалізується в зручному інтерфейсі за допомогою браузера, без потреби в фізичному обладнанні.

Переваги такого підходу є суттєвими як з технічної, так і з організаційної точки зору. Перш за все, варто відзначити економічну ефективність. Віртуальна лабораторія не потребує закупівлі фізичних комплектуючих, інструментів чи навчальних стендів, що особливо важливо для навчальних закладів з обмеженим бюджетом. Витрати на обслуговування також зведені до мінімуму: немає потреби ремонтувати чи оновлювати обладнання, яке з часом застаріває або виходить з ладу. Програмний продукт після створення може багаторазово використовуватися без додаткових витрат.

Другою важливою перевагою є універсальність і масштабованість рішення. Один і той самий веб-додаток може використовуватись багатьма студентами одночасно та необмежену кількість раз. Його можна запускати на будь-якому пристрої, що має браузер, незалежно від операційної системи чи місця розташування користувача. Це дозволяє проводити навчання у зручному для студента режимі — як у класі, так і вдома. Додаток не потребує встановлення і працює навіть локально, що значно спрощує його використання.

Окрему увагу варто приділити питанню безпеки. При використанні реального обладнання завжди існує ризик його пошкодження або отримання травми студентом через необережність чи помилки в підключенні. У віртуальному середовищі ці ризики повністю виключені. Студент може багаторазово виконувати завдання, експериментувати з конфігураціями, пробувати різні варіанти без жодних наслідків для обладнання чи власного здоров’я. Це створює комфортні умови для навчання, в яких користувач не боїться помилитися і може повністю зосередитись на засвоєнні матеріалу.

Додатковою перевагою є використання сучасних технологій з відкритим кодом (HTML5, CSS, JavaScript, Vue.js), які не потребують оплати ліцензій та доступні для вільного розповсюдження. Це дозволяє швидко впроваджувати, оновлювати й адаптувати програму під конкретні потреби навчального процесу.

Таким чином, розробка віртуальної лабораторії є ефективним рішенням для створення доступного, інтерактивного та безпечного середовища для навчання студентів.

2 РОЗРОБКА ВІРТУАЛЬНОЇ ЛАБОРАТОРІЇ ДЛЯ ВИВЧЕННЯ АРХІТЕКТУРИ КОМП’ЮТЕРА

2.1 Проєктування архітектури віртуальної лабораторії

Розроблена віртуальна лабораторія реалізована як сучасний веб-додаток, що працює без потреби встановлення — достатньо лише браузер. Вся функціональність зосереджена на клієнтській стороні: обробка взаємодій, логіка і візуалізація відбуваються безпосередньо у браузері за допомогою JavaScript. У найпростішому варіанті лабораторія може запускатися навіть із локального диска, без інтернету.

Застосунок реалізований як набір окремих HTML-сторінок, кожна з яких відповідає окремій лабораторії. Кожна сторінка містить інтерактивний інтерфейс, побудований з використанням фреймворку Vue.js. Після відкриття сторінки завантажується відповідний JavaScript-код і починає працювати Vue - додаток, який керує всією взаємодією користувача з інтерфейсом. Навігація між лабораторіями здійснюється через перехід між HTML-файлами, але всередині кожної сторінки усе працює динамічно: перетягування елементів, відображення опису, оновлення стану — усе виконується миттєво, без перезавантаження. Завдяки фреймворку Vue дані, елементи та події зручно зв’язані, а зміни на екрані відбуваються плавно і без затримок. Таким чином це дозволяє нам створити гнучкий та зрозумілий інтерфейс, який добре підходить для навчання через взаємодію з віртуальними об’єктами.

Подібна архітектура має низку переваг: висока швидкість, можливість офлайн-доступу, легке масштабування. Компонентна структура Vue дозволяє просто додавати нові лабораторії або змінювати функціональність без повної перебудови проєкту. Це не лише технічно виправдано, а й зручно для користувача, бо дозволяє сконцентруватися на змісті, не відволікаючись на навігацію чи очікування завантаження.

Структурно веб-додаток поділено на чотири модулі, кожен з яких відповідає одній навчальній лабораторії. Кожна лабораторія оформлена як незалежна частина — зі своїм власним кодом, зовнішнім виглядом і логікою. Це дозволяє легко змінювати або допрацьовувати один модуль, не зачіпаючи інші. Наприклад, створено компонент для лабораторії “Компоненти комп’ютера”, який відображає графічні зображення компонентів та їх роз’ємів ПК, таких як материнська плата, CPU, RAM, GPU, накопичувач тощо.

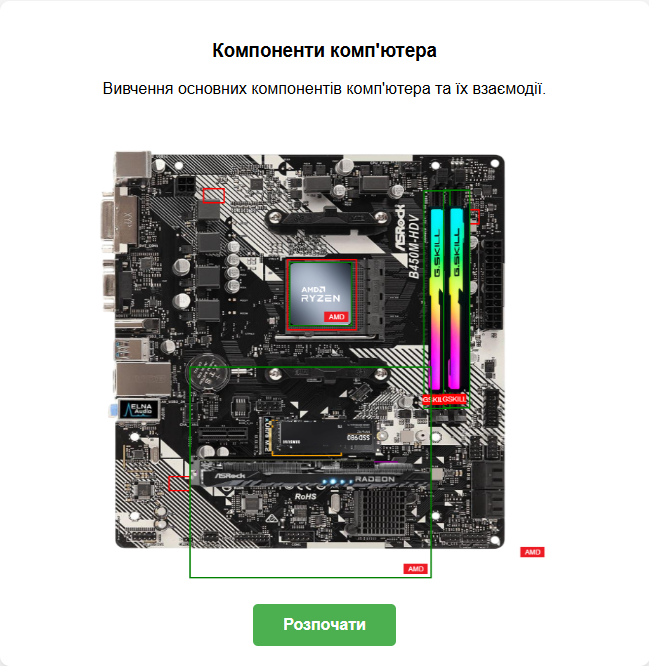


Рисунок 2.1.1 – Інтерфейс лабораторії «Компоненти комп’ютера»

Інший компонент відповідає за лабораторію “Периферійні пристрої” та містить зображення периферійних пристроїв (комп'ютерна миша, геймпад, клавіатура, монітор) з поясненням їх принципу роботи та з поясненювальними відео.

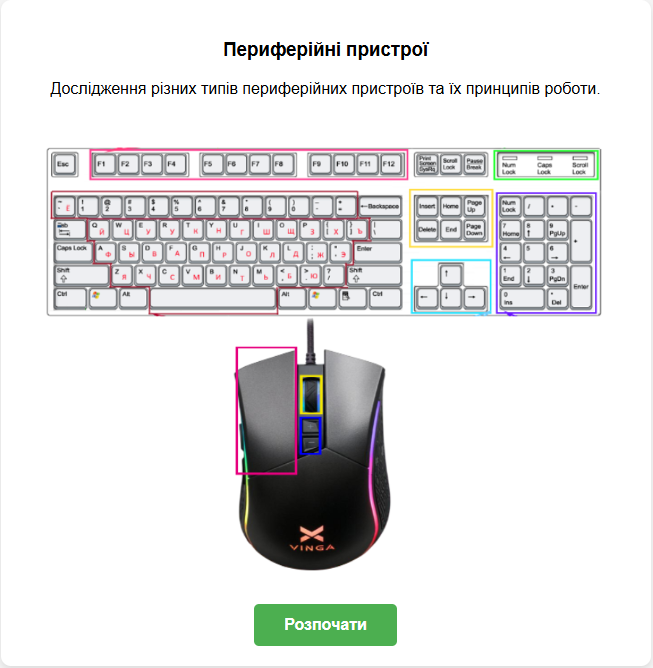


Рисунок 2.1.2 – Інтерфейс лабораторії «Периферійні пристрої»

Компоненти для конвертації чисел та машинного коду реалізують відповідні форми вводу/виводу даних.

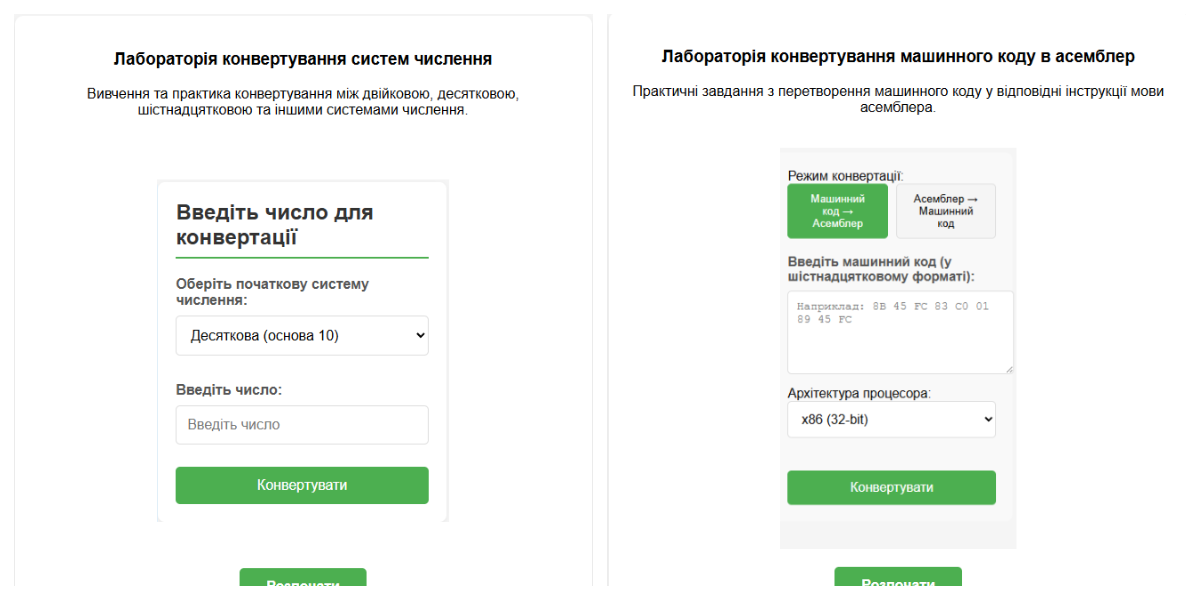


Рисунок 2.1.3 – Інтерфейс лабораторій з конвертування

При проєктуванні інтерфейсу використано принципи *drag-and-drop* для інтерактивних лабораторних робіт. HTML5 надає стандартні інструменти для реалізації перетягування елементів, коли користувач може перетягнути за допомогою зажатої лівої клавіші миші об’єкт в інше місце на екрані. У модулях ця можливість використовується для моделювання процесу складання комп’ютера та підключення пристроїв. На макеті комп’ютера визначено зони, куди дозволено відпускати певні об’єкти (наприклад, область слоту оперативної пам’яті на материнській платі).

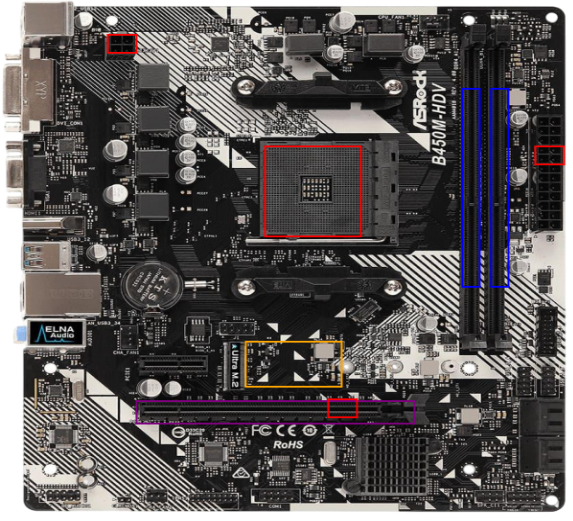


Рисунок 2.1.4 – Зони на материнській платі

Користувач перетягує компонент до відповідної зони й коли компонент знаходиться на правильному місці на материнці (наприклад процесор в сокеті материнки), спрацьовує подія, яка показує що зроблено все правильно.

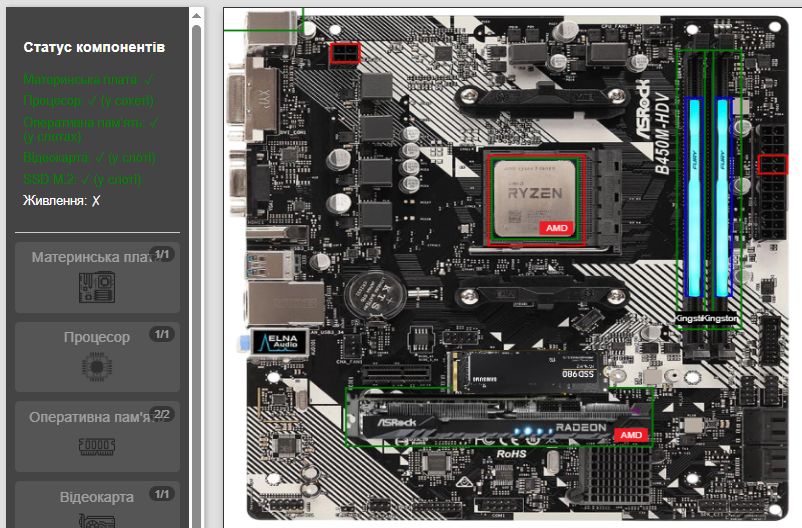


Рисунок 2.1.5 – Встановлені компонентів на материнську плату

Таким чином, архітектура інтерфейсу побудована так, щоб спрямовувати студента навчити правильно підключати елементи комп’ютера і паралельно навчати принципу їх роботи.

Архітектура модулів конвертації чисел і коду побудована навколо форм вводу/виводу. Компонент “Конвертування систем числення” містить поля вводу для числа та вибору початкової системи числення (наприклад, десяткова, двійкова, шістнадцяткова). В цьому модулі не використовується *drag-and-drop*, проте акцент зроблено на миттєвому зворотному зв’язку: студент вводить число та обирає системи числення, після чого може натиснути кнопку “Конвертувати” – і результат з’явиться одразу на екрані користувача.

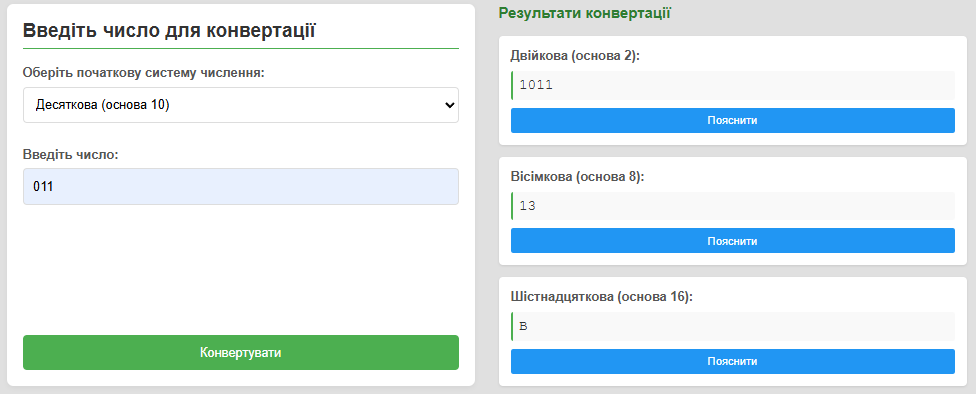


Рисунок 2.1.6 - Конвертація чисел у різні системи

Лабораторія “Конвертування машинного коду в асемблер” спроєктована за схожим принципом: є текстове поле для вводу машинного коду (16-кового коду) та поле для відображення отриманої асемблерної інструкції.

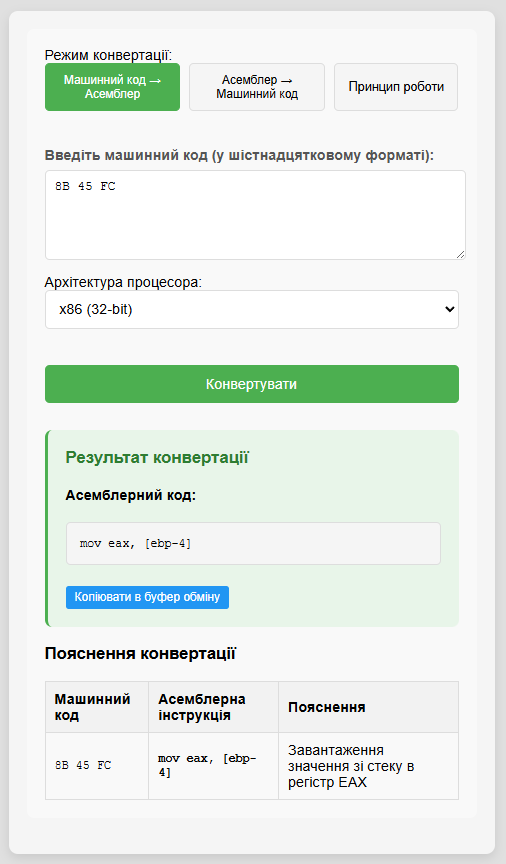


Рисунок 2.1.7 - Конвертація машинного коду в асемблер

У цього модуля закладено таблицю відповідностей кодів і мнемонік асемблера. Фактично реалізовано простий дизасемблер, оскільки програмно виконується перетворення машинних команд на асемблерні мнемоніки. Наприклад, якщо студент вводить умовний код 1010, програма знаходить у таблиці, що бінарному коду 1010 відповідає інструкція ADD – і виводить “ADD” у полі результату. Набір команд, підтримуваних конвертором, було визначено під час проєктування (це спрощена модель процесора з обмеженим списком команд для навчальних цілей). Архітектурно цей компонент побудований так: при ініціалізації завантажується структура даних, що зберігає відповідності між машинними кодами та асемблером; при кожній зміні вводу викликається функція пошуку відповідного мнемоніка.

Загалом, архітектура віртуальної лабораторії спроєктована модульно та масштабовано. Компонентний підхід Vue дозволив ізолювати код різних лабораторій, що полегшує подальшу підтримку та розширення системи. Наприклад, за потреби можна додати новий модуль (наприклад, симуляцію роботи арифметико-логічного пристрою або візуалізацію роботи пам’яті) як окремий Vue-компонент, не зачіпаючи існуючі частини. Комунікація між компонентами мінімальна – фактично кожна лабораторія працює автономно, а загальним є лише каркас додатка (меню навігації та стилі оформлення сайту).

2.2. Структура коду та його оптимізація

Після визначення архітектури віртуальної лабораторії, наступним кроком є розробка структури коду, що відповідає цій архітектурі, та його оптимізація для забезпечення ефективної роботи та відповідності технічним вимогам.

Код віртуальної лабораторії «Компоненти комп’ютера» побудований так, щоб був простий для розуміння, легко змінюваний та швидко працював у браузері. Програма складається з HTML-сторінки, CSS-файлу, зі стилями та інтерактивної частини, написаної на JavaScript з використанням бібліотеки Vue.js. Основна розмітка та логіка розміщені у файлі virtualka.html, а стилі оформлення — в окремому файлі style1.css.

У програмі використовується підхід реактивності, тобто дані, які зберігаються у змінних, автоматично оновлюють вигляд на екрані. Наприклад, якщо користувач перетягнув процесор і встановив його на материнську плату, то Vue сам оновлює відображення без перезавантаження сторінки. Усі компоненти (процесор, пам’ять, відеокарта та інші) перевіряються через спільну функцію isComponentPlaced, яка повертає true або false:

isComponentPlaced(name) {

return this.components.some(component => component.name === name && component.placed)

}

Замість того щоб писати окрему перевірку для кожного компонента, ця функція приймає ім’я компонента і перевіряє, чи він уже встановлений. Такий підхід дозволяє використовувати один і той самий код багато разів — це і є **оптимізація**. Наприклад, щоб перевірити, чи встановлено процесор, використовується такий HTML-код:

<li :class="{ completed: isComponentPlaced('cpu') && cpuInSocket }">

Процесор: {{ isComponentPlaced('cpu') ? (cpuInSocket ? '✓ (у сокеті)' : '✗ (не в сокеті)') : '✗' }}

</li>

Ще один приклад оптимізації — це використання спільних CSS-класів. Для всіх зображень компонентів використовується клас .component, щоб не дублювати стилі:

.component {

width: 60px;

height: 60px;

margin: 10px;

cursor: grab;

}

Це дозволяє легко змінити розмір усіх компонентів одночасно, просто змінивши одне місце в CSS-файлі. Для зон, куди перетягуються елементи, використовується клас .drop-zone. Всі події перетягування обробляються через одну функцію handleDrop, яка визначає, чи можна вставити компонент у певне місце:

handleDrop(slot) {

if (this.draggingComponent === slot && !this[slot + 'Installed']) {

this[slot + 'Installed'] = true;

} else {

alert("Компонент не підходить або вже встановлений");

}

}

Замість того щоб створювати окрему функцію для кожного типу (процесор, відеокарта, пам’ять), тут використовується змінна slot, яка передається у функцію. Таким чином, код працює з будь-яким компонентом за однаковими правилами. Це сильно зменшує кількість коду та підвищує його зручність для подальших змін.

Вся логіка відображення побудована на простих умовах: якщо компонент встановлено то програма показує іконку або позначку ✓, якщо ні — показує ✗. Наприклад, у спискі статуса компонентів зліва біля кожного пристрою автоматично з’являється правильний значок, що дуже зручно для користувача.

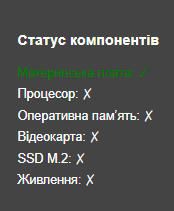


Рисунок 2.2.1 – Список статусів компонентів

Усе ,це дозволило зробити код більш гнучким, чистим та легким у підтримці. Такий підхід допомагає уникнути помилок, легко знаходити частини, які потрібно змінити, і при цьому забезпечує високу швидкість роботи навіть на слабких комп’ютерах. Важливо також те, що код добре коментований і всі змінні мають зрозумілі назви — це важливо не лише для розробника, а й для викладачів чи інших студентів, які будуть працювати з цим проєктом у майбутньому.

2.3. Використання Javascript framework Vue для візуального програмування

Одним із ключових рішень при створенні віртуальної лабораторії було використання фреймворку Vue.js. Це сучасний інструмент для створення веб-інтерфейсів, який дозволив зробити нашу програму зручною в розробці, зрозумілою у структурі й легкою в підтримці. Vue.js сфокусований на роботі з інтерфейсом (тобто тим, що бачить користувач) і не потребує складного налаштування для початку роботи.

У нашій програмі ми поділили інтерфейс на окремі частини — компоненти. Наприклад, окрема частина відповідає за лабораторію з комплектуючими ПК, інша — за периферійні пристрої або за конвертацію чисел. Це зробило код структурованим і дозволило працювати над кожним модулем незалежно. Деякі елементи, наприклад спливаюче вікно з описом компоненту, ми використали в кількох лабораторіях повторно — це зменшило обсяг коду і спростило розробку.

Vue автоматично слідкує за змінами в даних. Якщо користувач змінює вхідне число або вибирає іншу систему числення, результат оновлюється миттєво — не потрібно писати окремі функції для обробки подій, як це робиться у класичному JavaScript. У лабораторії з конвертацією чисел це виглядає так: ми пов'язали змінні inputValue, inputBase, outputBase та result, і Vue сам оновлює result, коли щось змінюється. Код для цього простий і зрозумілий:

convertNumber() {

const input = parseInt(this.inputValue, this.inputBase);

if (!isNaN(input)) {

this.result = input.toString(this.outputBase).toUpperCase();

} else {

this.result = "Помилка вводу";

}

}

Інтерфейс у Vue ми описували як HTML-шаблони з додатковими вказівками. Наприклад, у лабораторії з компонентами комп’ютера список доступних елементів показується за допомогою конструкції типу:

<div v-for="(item, index) in components" :key="index" class="item-block">

<img :src="item.image" :alt="item.name">

<p>{{ item.name }}</p>

</div>

Цей код бере дані з масиву components і створює для кожного елементу свою частину інтерфейсу. Якщо масив зміниться — Vue автоматично оновить відповідну частину сторінки. Це дуже зручно, бо не потрібно вручну змінювати HTML.

Початкове підключення Vue було дуже простим. Ми просто підключили скрипт Vue через Content Delivery Network у <script> і оголосили новий екземпляр Vue, що дозволило нам підключити фреймворк напряму з інтернету, не зберігаючи її у своєму проєкті:

const app = Vue.createApp({

data() {

return {

inputValue: '',

inputBase: 10,

outputBase: 2,

result: ''

};

},

methods: {

convertNumber

}

});

app.mount('#app');

Це дозволило нам швидко почати працювати без складних налаштувань. Пізніше ми змогли розширити програму, не переписуючи її з нуля.

Перевага Vue в тому, що його можна поступово ускладнювати: додавати нові функції, підключати інші бібліотеки. У нашому випадку базового функціоналу Vue вистачило повністю. Але якщо колись ми захочемо додати авторизацію студентів або збереження результатів — Vue легко підтримує такі розширення.

Отже, використання Vue значно полегшило розробку: ми змогли зосередитися на освітньому змісті лабораторії, а не на технічних труднощах. В результаті ми створили зрозумілий, швидкий та зручний у підтримці вебдодаток.

2.4. Тестування та налагодження віртуальної лабораторії

Тестування є важливою частиною розробки будь якого комплексного проєкту, оскільки від правильності роботи залежить, наскільки ефективно користувач зможе користуватись програмою. Було проведено тестування окремо для кожної з лабораторій, щоб переконатися, що всі функції працюють як слід.

У лабораторії "Компоненти комп'ютера" було перевірено, чи всі елементи можна перетягнути в потрібні місця. Наприклад, було додано материнську плату, процесор, модулі пам’яті, відеокарту, SSD і блок живлення. Було протестовано, чи можна кожен із цих компонентів вставити в правильний слот, і чи з’являється опис після встановлення. Також було перевірено, що стан схеми правильно оновлюється — наприклад, якщо зібрати все правильно і додати монітор, то з’являється повідомлення, що комп’ютер зібрано правильно. Помилкові дії, такі як повторне додавання вже вставленого елемента або спроба вставити відеокарту у роз’єм оперативної пам’яті, також було протестовано. У таких випадках система видавала підказки або ігнорувала дію.

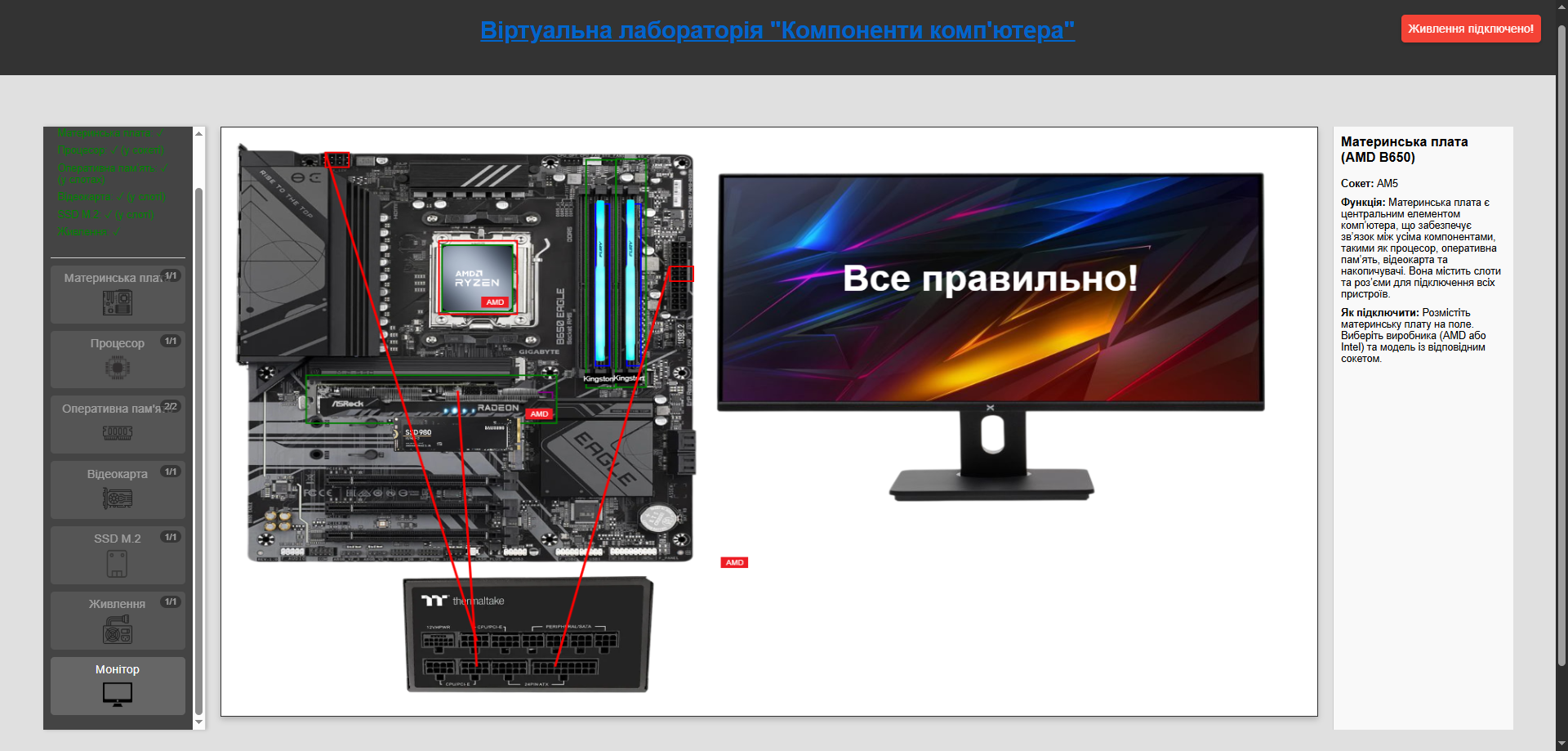


Рисунок 2.4.1 – Тестування лабораторії "Компоненти комп'ютера"

У лабораторії "Периферійні пристрої" було перевірено, як працює додавання клавіатури, мишки, геймпада, джойстика і монітора. Для кожного з пристроїв було перевірено, чи з’являються відповідні описи і чи працюють інтерактивні зони. Наприклад, при натисканні на зону миші з’являвся опис кнопок, а при натисканні правою кнопкою — відкривалося відео про обслуговування пристрою. Також було протестовано, що відбувається при додаванні кількох однакових пристроїв або при неправильних діях. Усі ці випадки оброблялися коректно.

,

Рисунок 2.4.2 – Тестування лабораторії "Периферійні пристрої"

У лабораторії "Конвертація систем числення" було протестовано обчислення. Наприклад, було введено десяткове число 255 і переконанося, що результат у двійковій системі — 11111111, а у шістнадцятковій — FF. Також було перевірено зворотні перетворення — наприклад, що двійкове 1010 правильно перетворюється у 10. Було протестовано нуль і довгі числа, а також випадки некоректного вводу — наприклад, якщо ввести літеру замість числа, система показувала повідомлення про помилку.

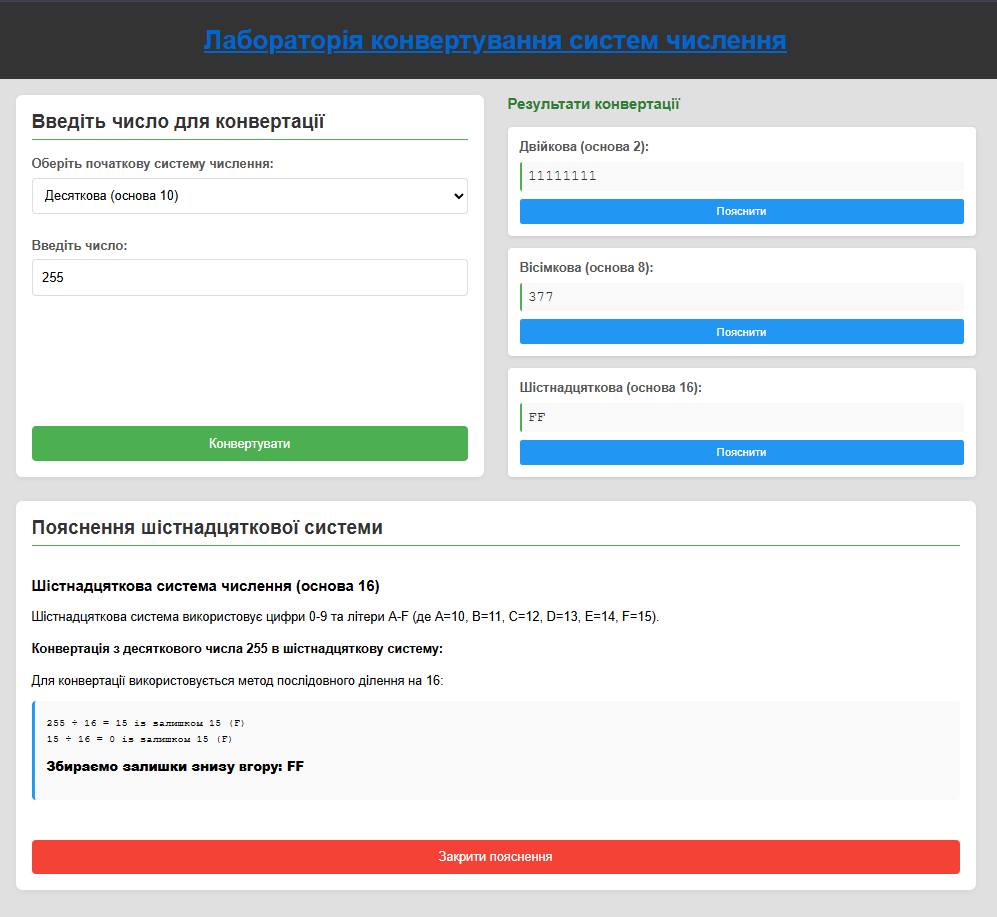


Рисунок 2.4.3 – Тестування лабораторії "Конвертація систем числення"

У лабораторії "Машинний код – Асемблер" було перевірено, чи правильно працює перетворення шістнадцяткових кодів на текстові інструкції. Наприклад, було введено шістнадцяткове значення 8B 45 FC, що відповідає команді 'eax, [ebp-4]' — завантаження значення зі стеку в регістр EAX, і у результаті з’явилася відповідна команда. Також було перевірено реакцію системи на введення неправильного коду — у такому випадку програма видає повідомлення “Невідома команда”. Це підтвердило, що лабораторія обробляє як передбачені сценарії, так і помилкові ситуації.

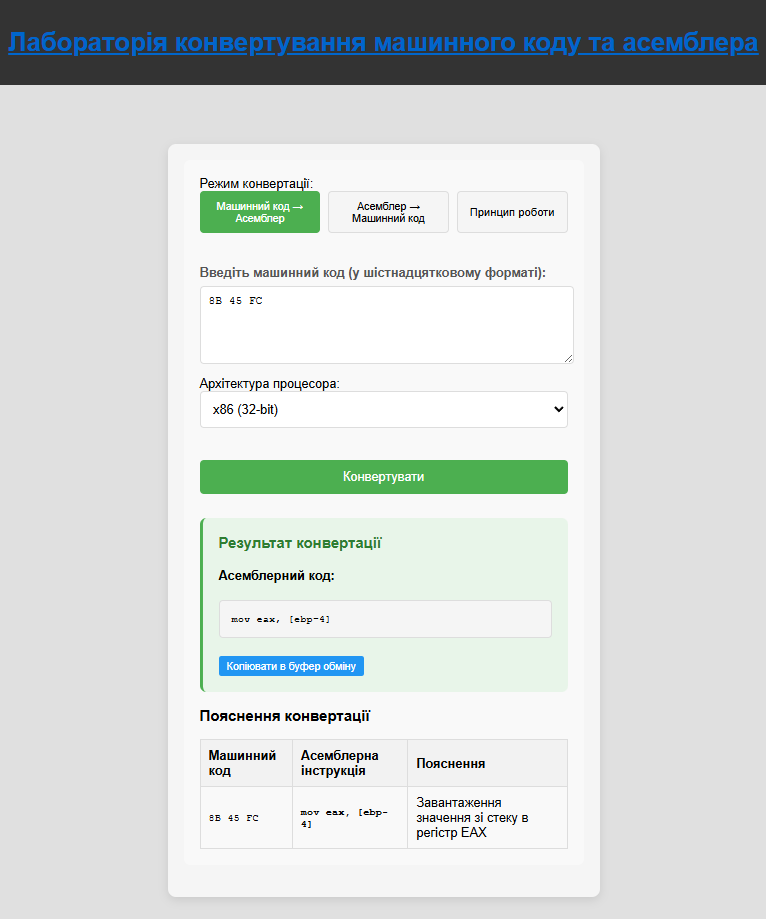


Рисунок 2.4.4 – Тестування лабораторії "Машинний код – Асемблер"

Крім перевірки функціоналу, було проведено юзабіліті-тестування. Студенти і викладачі пробували працювати з лабораторією без сторонньої допомоги. За їхніми відгуками, інтерфейс виявився зрозумілим і зручним. Інструкції перед початком роботи, підказки при наведенні курсора, великі шрифти і контрастні кольори — усе це зробило використання лабораторії комфортним. Було протестовано також адаптивність інтерфейсу — програма коректно працює як на великому моніторі, так і на ноутбуці. На телефоні інтерфейс відкривається, але через специфіку *drag-and-drop* головною платформою лишається ПК.

Додаток було протестовано в різних браузерах: Chrome, Firefox, Edge. Усі основні функції працювали однаково. Лише у Firefox було незначне візуальне відхилення при перетягуванні, яке було виправлено стилем drag-image. Також було перевірено програму на різних екранах — від маленьких ноутбуків до широких моніторів. Адаптивна верстка спрацювала коректно.

Нарешті, було проведено налагодження (debugging). Під час тестування було знайдено кілька помилок — наприклад, підказка про компонент не з’являлася при дуже швидкому перетягуванні. Це було виправлено, додавши затримку через Vue.nextTick, щоб дочекатися оновлення даних. Також при конвертації у шістнадцяткову систему спочатку виводились літери у нижньому регістрі (a, b, c), але це було виправлено методом toUpperCase(). Для налагодження активно використовувалися вкладки Console і Network в інструментах розробника браузера — це допомогло знайти і виправити проблеми із завантаженням ресурсів.

Завдяки такому тестуванню і налагодженню було досягнуто впевненості, що програма стабільна, правильно працює у всіх випадках і готова до використання в навчальному процесі.

3. ІНСТРУКЦІЯ ПО РОБОТІ З ПРОГРАМОЮ

Віртуальна лабораторія надається у вигляді веб-сайту (HTML/CSS/JS-файли) і працює в сучасному браузері (Chrome, Firefox, Safari, Edge). JavaScript має бути увімкнено. Щоб почати роботу, відкрийте файл main.html. З головної сторінки можна вибрати потрібну лабораторію, яка відкриється без перезавантаження сторінки.

Модуль «Компоненти комп’ютера»: дозволяє складати ПК із віртуальних компонентів. Перетягніть елемент до відповідного роз’єму на материнській платі. При правильному розміщенні з’являється інформація про деталь. Якщо обрано неправильне місце — компонент не встановлюється. У будь-який момент його можна перемістити або видалити.

Модуль «Периферійні пристрої»: показує, як зовнішні пристрої підключаються до комп’ютера. Перетягніть іконку до відповідного порту. Після підключення з’явиться опис пристрою. Деякі пристрої реагують на клік замість перетягування. Вивчення інтерфейсів допомагає зрозуміти роль кожного пристрою в системі.

Модуль «Конвертування систем числення»: дозволяє переводити числа між системами (2, 8, 10, 16). Введіть число, виберіть системи й натисніть “Конвертувати”. Невірні символи видають попередження. Можна швидко змінювати напрямок перетворення.

Модуль «Конвертування машинного коду»: перетворює шістнадцяткові коди в команди асемблера. Наприклад, 8B 45 FC = eax, [ebp-4]. Якщо код не розпізнано, з’являється повідомлення про помилку. Дотримуйтесь правильного формату вводу (байти через пробіл).

Програма допускає багаторазове використання. Якщо інтерфейс “завис” — перезавантажте сторінку. Вивчайте теорію перед практикою, а результати виконуйте згідно з методичними вказівками. У разі навчання онлайн – надайте результати викладачу у зручній формі (наприклад, скриншотом).

4 ЕКОНОМІЧНА ЧАСТИНА

Завданням економічної частини кваліфікаційної роботи є розрахунок собівартості та відпускної ціни віртуальної лабораторії для візуалізації архітектури комп’ютерних систем. У цьому розділі розраховано вартість матеріальних витрат на створення віртуальної лабораторії, витрат на електроенергію, а також на оплату праці та витрати по непрямих статях калькуляції. Всі розрахунки були проведені за статтями калькуляції, які наведено у підручнику «Економіка підприємства» за редакцією С.Ф.Покропивного. Результати обчислень представлені у вигляді таблиць.

Таблиця 4.1 – Вихідні дані

|  |  |
| --- | --- |
| Найменування показників | Значення |
| Кількість аркушів паперу, шт. | 20 |
| Ціна за одиницю, грн. | 0,35 |
| Кількість флеш накопичувачів, шт, | 1 |
| Ціна флеш накопичувача, грн. | 210 |
| Потужність ЕОМ, кВт | 0,25 |
| Тривалість роботи ЕОМ, год. | 76 |
| Ціна 1 кВт/ год. електроенергії, грн. | 10,66 |
| Фонд оплати праці за місяць, грн. | 10920,00 |
| Плановий фонд робочого часу за місяць, днів | 22 |
| Додаткова заробітна плата, % | 12 |
| Єдиний соціальний внесок, % | 22 |
| Постанова задачі, люд.год. | 6 |
| Розробка дизайну сайту, люд.год. | 22 |
| Створення вебсайту, люд.год. | 32 |
| Відладка вебсайту, люд.год. | 16 |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| Найменування показників | Значення |
| Витрати на утримання та експлуатацію машин і устаткування, % | 15 |
| Загальногосподарські витрати, % | 18 |
| Позавиробничі витрати, % | 0,5 |
| Рівень рентабельності, % | 20 |
| Ставка ПДВ, % | 20 |

Таблиця 4.2 – Розрахунок матеріальних витрат на створення вебсайту

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Кількість аркушів паперу, шт. | Ціна одного аркушу паперу, грн. | Вартість аркушів, грн. | Кількість флеш накопичувачів, шт, | Ціна флеш накопичувача, грн. | Матеріа-льні витрати, грн. |
| 20 | 0,35 | 7 | 1 | 210 | 217 |

Таблиця 4.3 – Розрахунок витрат на електроенергію

|  |  |  |  |
| --- | --- | --- | --- |
| Час роботи ЕОМ, год. | Потужність ЕОМ, кВт | Ціна 1 кВт/год. електроенергії, грн. | Вартість на електроенергію, грн. |
| 76 | 0,25 | 10,66 | 202,54 |

Таблиця 4.4 – Розрахунок витрат на оплату праці

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Загальні трудовитрати, люд.год. | Фонд оплати праці, грн. | Фонд робочого часу, днів | Основна заробітна плата, грн. | Додаткова заробітна плата, грн. |
| 76 | 10920 | 22 | 4715,80 | 565,90 |

Таблиця 4.5 – Розрахунок суми єдиного соціального внеску

|  |  |  |
| --- | --- | --- |
| Сума основної і додаткової заробітної плати, грн. | Єдиний соціальний внесок, % | Єдиний соціальний внесок, грн. |
| 5281,70 | 22 | 1161,97 |

Таблиця 6.6 – Розрахунок витрат по непрямих статтях калькуляції

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Основна заробітна плата, грн. | Витрати на утримання обладнання, грн. | Загальногосподарські витрати, грн. | Виробнича собівартість, грн. | Позавиробничі витрати, грн. |
| 4715,80 | 707,37 | 976,17 | 8546,75 | 42,7 |

Таблиця 6.7 – Розрахунок витрат на створення веб-лабораторії

|  |  |  |
| --- | --- | --- |
| Найменування статей калькуляції | Витрати, грн. | Структура, % |
| Матеріальні витрати | 217 | 2,53 |
| Електроенергія | 202,54 | 2,36 |
| Основна заробітна плата | 4715,80 | 54,9 |
| Додаткова заробітна плата | 565,90 | 6,59 |
| Єдиний соціальний внесок | 1161,97 | 13,53 |
| Утримання та експлутація машин і устаткування | 707,37 | 8,24 |
| Загальногосподарські витрати | 976,17 | 11,36 |
| Виробнича собівартість | 8546,75 | 99,5 |
| Позавиробничі витрати | 42,7 | 0,5 |
| Повна собівартість | 8589,45 | 100 |

Таблиця 6.8 – Розрахунок витрат на створення веб-лабораторії

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Повна собівартість, грн | Прибуток, грн. | Оптова ціна, грн. | ПДВ, грн. | Відпускна ціна, грн. |
| 8589,45 | 1717,89 | 10307,34 | 2061,47 | 12368,81 |

Отже, у ході виконання економічної частини було визначено витрати, пов’язані з виготовленням та збутом сайту, а також обґрунтовано його ціну; обчислено суму прибутку та суму ПДВ; оптову і відпускну ціну вебсайту. В результаті отримано повну собівартість програмного продукту, що становить 8589,45 грн., і відпускну ціну, що становить 12368,81 грн.

Також було з’ясовано, що на повну собівартість сайту впливають такі витрати:

1. матеріальні витрати на створення вебсайту;
2. витрати на електроенергію;
3. повна і додаткова заробітна плата працівника, який задіяний в процесі розробки;
4. єдиний соціальний внесок;
5. утримання та експлуатація обладнання, яке безпосередньо використовується для розробки;
6. загальногосподарські та позавиробничі витрати.

В свою чергу на відпускну ціну вебсайту впливають всі вище названі чинники, а також декілька інших, таких як:

1. оптова ціна (на неї має вплив прибуток, який обчислюється враховуючи процент рентабельності);
2. податок на додану вартість.

5 ОХОРОНА ПРАЦІ

У процесі реалізації кваліфікаційної роботи важливою складовою стала організація безпечного середовища праці. Незважаючи на те, що розробка вебдодатка здійснювалась у домашніх умовах, характер використаного обладнання, тривалість роботи за комп’ютером, а також зовнішні ризики, пов’язані з воєнним станом, вимагали ретельного дотримання нормативних вимог з охорони праці, електробезпеки, пожежної безпеки, а також санітарно-гігієнічних та ергономічних стандартів. При цьому було враховано вимоги ДСанПіН 3.3.2.007-98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», що регламентує умови та режими праці під час тривалої роботи з комп’ютерною технікою, а також положення Закону України «Про охорону праці» №2694-XII.

Розробка програмного забезпечення відбувалась у домашньому середовищі, яке тимчасово виконувало роль робочої зони. У ході реалізації використовувались персональний комп’ютер, два монітори, периферійні пристрої, мережеве обладнання та джерела безперебійного живлення. Усе устаткування живилось від стандартної побутової мережі змінного струму напругою 220 В. При цьому велика увага приділялася дотриманню правил безпечної експлуатації електроустановок відповідно до ПУЕ, НПАОП 40.1-1.32-01, а також використанню сертифікованих подовжувачів і мережевих фільтрів із заземленням.

Основні потенційні ризики при роботі за комп’ютером включають:

* ураження електричним струмом при використанні несправного обладнання або порушенні правил підключення;
* перевантаження зору через тривалу роботу з монітором;
* м’язове та психоемоційне перенапруження;
* ризики пов’язані з недотриманням ергономіки (неправильна поза, незручне робоче місце);
* пожежні ризики внаслідок перегріву обладнання або використання пошкоджених кабелів;
* необхідність оперативного реагування в умовах воєнного стану.

Для мінімізації ризиків було впроваджено ряд заходів. Робоче місце організоване відповідно до гігієнічних норм: монітор розташовано на відстані 60–70 см від очей, верхній край екрана на рівні погляду; робоче крісло має анатомічну спинку, регульовану висоту та підлокітники; використовується розсіяне світлодіодне освітлення, яке не створює відблисків. Згідно з рекомендаціями ДСанПіН, дотримувався режим праці та відпочинку — кожні 50 хвилин роботи за комп’ютером передбачалась 10–15-хвилинна перерва, під час якої виконувалась гімнастика для очей та розминка.

Комп’ютерне обладнання регулярно перевірялось на справність. Усі підключення здійснювались лише через справні розетки із заземленням або через мережеві фільтри з автоматичним вимикачем. Не допускалося використання відкритої проводки, скруток або подовжувачів кустарного виробництва. Згідно з вимогами НПАОП та Закону «Про охорону праці», користувач повинен мати можливість швидкого відключення живлення в разі надзвичайної ситуації. Усі кабелі були організовані так, щоб уникнути перетинання проходів, зменшити ризик спотикання та захистити їх від механічних пошкоджень.

Також були враховані норми пожежної безпеки: робоча зона не містила легкозаймистих матеріалів, джерел відкритого вогню чи надлишкового нагрівання. У приміщенні знаходився порошковий вогнегасник типу ABC, придатний до використання з електрообладнанням. Було організовано швидкий доступ до засобів знеструмлення.

В умовах воєнного стану, згідно із Законом України «Про правовий режим воєнного стану» та рекомендаціями ДСНС, було передбачено алгоритм дій на випадок повітряної тривоги або обстрілу. У разі сигналу повітряної тривоги передбачалося припинення роботи, відключення всього обладнання, зачинення вікон, збирання найнеобхідніших речей та евакуація до найближчого укриття. Була підготовлена тривожна валіза з документами, аптечкою, водою та ліхтариком. У разі неможливості дістатися укриття — визначено захищені зони у приміщенні (ванна кімната, коридор без вікон), що відповідають вимогам щодо укриття.

Крім того, з метою безпеки було вивчено інструкції з першої домедичної допомоги, номери екстрених служб, адреси найближчих пунктів укриття та медичних установ. Важливу роль відіграє також збереження психоемоційної стійкості в умовах ризику — під час повітряних тривог користувач уникав перебування біля вікон, не користувався електроприладами і не наражався на небезпеку.

Таким чином, під час виконання кваліфікаційної роботи були забезпечені всі основні вимоги щодо безпеки праці, з урахуванням специфіки домашнього середовища, роботи за комп’ютером та актуального стану в країні.

ВИСНОВОК

У процесі виконання кваліфікаційної роботи було створено віртуальну навчальну лабораторію для візуалізації архітектури комп’ютерних систем. Розроблений веб-додаток забезпечує інтерактивне середовище, де користувачі можуть ознайомитися з будовою комп’ютера, його компонентами, периферією, а також потренуватись у конвертації чисел і коду.

Проведено аналіз вимог до програмного продукту та доведено доцільність створення саме веб-орієнтованого рішення. Обрано технологію Vue.js як основу для реалізації SPA-застосунку. Проєкт поділений на модулі, що відповідають окремим лабораторіям, що робить систему масштабованою й гнучкою.

Створено простий та інтуїтивний інтерфейс з елементами взаємодії (перетягування, підказки), що полегшує навчання. Код реалізовано відповідно до сучасних вимог до оптимізації, з використанням надійних вбудованих функцій та обробкою типових помилок.

Проведено тестування функціональності та зручності використання. Всі модулі працюють стабільно у популярних браузерах. Після тестування було внесено покращення на основі зворотного зв’язку.

Підготовлено інструкцію для користувачів, яка пояснює роботу кожного модуля. В економічній частині доведено, що такий підхід дозволяє заощадити на обладнанні, забезпечуючи доступність для студентів у будь-якому місці.

У розділі охорони праці розглянуто основні ризики, пов’язані з роботою за комп’ютером, та наведено рекомендації щодо безпечної організації робочого місця. Програма не містить небезпечних факторів при дотриманні базових правил.

Практичне значення проєкту полягає у використанні сучасних веб-технологій у навчанні. Лабораторія може бути адаптована для інших предметів і розширена за рахунок нових модулів, інтерактивних тестів, а в перспективі — 3D-візуалізацій. Результат повністю відповідає поставленим завданням і рекомендований до використання в навчальному процесі.

ПЕРЕЛІК ДЖЕРЕЛ ІНФОРМАЦІЇ

* 1. Методичні вказівки до виконання кваліфікаційної роботи зі спеціальності 123 «Комп’ютерна інженерія». ВСП "СФК НУХТ". URL: https://docs.google.com/document/d/1uyAdVL3t0hGgWALOsw6IQgp016cytyEa/edit?tab=t.0 (дата звернення: 01.04.2025).
  2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення: 02.04.2025).
  3. Віртуальні лабораторії PhET. URL: https://phet.colorado.edu/uk/ (дата звернення: 02.04.2025).
  4. The Importance of Virtual Labs and Simulation Systems for Educational Institutions. URL: https://praxilabs.com/en/blog/2018/03/15/importance-virtual-labs-simulation-systems-educational-institutions/#:~:text=Among%20the%20tools%20used%20in,getting%20results%20using%20the%20computer (дата звернення: 06.04.2025).
  5. Virtual Labs are making state-of-the-art lab experiences accessible to a new generation of students. URL: https://www.labster.com/blog/cost-savings#:~:text=,wasted%20materials%20or%20costly%20breaks (дата звернення: 11.04.2025).
  6. The Progressive JavaScript Framework. URL: https://vuejs.org/ (дата звернення: 15.04.2025).
  7. Про правовий режим воєнного стану: Закон України від 12.05.2015   
     № 389‑VIII: станом на 14 трав. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/389-19#Text (дата звернення: 02.04.2025).
  8. Про охорону праці: Закон України від 14.10.1992 № 2694-XII: станом на 4 квіт. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 02.04.2025).