ES 215 - Computer Organization & Architecture

FINAL PROJECT REPORT

Bhavik Patel - 22110047 Hitesh Kumar - 22110098

Ruchit Jagodara - 22110102 Jinil Patel - 22110184

COA EDUCATIONAL GUI TOOL

GitHub Repository Link: https://github.com/Hit2737/COA_GUI_Tool_Project/

PROBLEM STATEMENT

Understanding and visualising complex concepts in computer organisation and architecture is challenging for students and educators due to the abstract nature of components like data structures, performance metrics, cache memory, and floating-point representation. Traditional teaching methods often lack interactive elements, making it difficult for learners to grasp the impacts of varying parameters on system performance and the behaviour of algorithms.

WHAT WAS OUR TARGET?

This project addressed these challenges by developing a comprehensive Graphical User Interface (GUI) tool that simulates and visualises key computer organisation and architecture concepts. The tool should include interactive components that allow users to visualise and analyse performance metrics based on user inputs, cache memory behaviour under different replacement policies, IEEE 754 floating-point representation, and number conversions between different bases. By providing a hands-on experience with these components, this tool will enhance understanding, foster experimentation, and support learning by making abstract concepts more tangible and accessible.

WHAT WE ACHIEVED?

1. PERFORMANCE METRICS ANALYSER (PMA):

Different Metrics Calculators: Implemented components that provide real-time calculated values of given metrics using the inputs provided by the user, either by using a slider or typing it manually.

Graph Visualization: Displayed graphical representation of some metrics to represent their relationship with the input parameters and understand the tradeoffs between different metrics.

Interactive Elements: Implemented dropdowns for users to select the metric to view its detailed information and calculation. Also, real-time updates are shown in the calculations and the graph representation to show the behaviour of the metric with the inputs used. Used the Chart.js library for a user-friendly interface to render graphs.

----

2. CACHE SIMULATOR (CS):

Replacement Policies: Implemented different Cache Replacement Policies to visualise how data replacement happens in Cache Memory under different policies. Given policies were used in the web application:

⇒ <u>Least Recently Used (LRU)</u>: Discards the least recently used data, assuming recently accessed data will be reused soon.

⇒ <u>Least Frequently Used (LFU)</u>: Removes the least frequently accessed data, focusing on discarding data with low usage.

⇒ <u>First In First Out (FiFo)</u>: Replaces the earliest added data without considering access frequency or recency.

⇒ <u>Last In First Out (LiFo)</u>: Removes the most recently added data, often used in stack-based contexts.

----

3. IEEE 754 FLOATING-POINT CONVERSION (IEEE):

IEEE Conversion Component: Allows users to input a decimal value and see its representation in IEEE 754 single and double-precision formats.

Precision Switching: Users can toggle between single and double precision, allowing comparison between the two formats.

Step-By-Step Conversion: The user can look at different conversion stages, including calculations for the sign bit, exponent, and mantissa, along with the exact value stored and the error due to conversion.

Special Case Handling: Select special cases like $\pm\infty$ NaN and see their representation in IEEE 754 floating point representation.

----

4. NUMBER CONVERSION (NC):

Base Conversion: Convert between any base system from 2 to 16 with a step-by-step explanation.

Conversion Process: The standard conversion process for converting from one base to another is explained, including intermediate steps to enhance understanding of the conversion method.

----

5. USER INTERACTION (UI & UX):

Interactive Elements: Incorporated sliders, dropdowns, and real-time parameter adjustment to allow users to interact with the tool dynamically.

Intuitive Interface: Designed the user interface (UI) to be intuitive and responsive, making it easy to navigate without a guide.

Validation and Error Handling: Ensured inputs are validated, providing feedback in case of errors to enhance the user experience (UX).

KEY LEARNINGS

Performance Metrics:

Key performance metrics—Cycles Per Instruction (CPI), Clock Rate, Overhead Time, Execution Time, Speed-Up, Throughput, and Efficiency—offer a structured approach to evaluating system performance. Each metric provides insights into processing speed, capacity, and resource utilisation, collectively enabling a comprehensive system efficiency assessment.

Parallelism:

The impact of parallel and sequential instruction execution was analysed to assess their effects on performance. Parallel processing can significantly enhance speed and efficiency by allowing concurrent operations, though it introduces additional complexity in dependency and synchronisation management.

Cache Memory:

Cache replacement policies, including Least Recently Used (LRU) and First-In-First-Out (FIFO), are critical in optimising data retrieval and minimising latency. By managing data efficiently, these policies improve cache performance and reduce execution time, which is essential for high-speed processing.

Floating-Point Representation:

The IEEE 754 standard for floating-point representation provides a structured format for storing real numbers, which is essential for accurate numerical computation. This standard divides values into sign, exponent, and mantissa, helping minimise precision errors while ensuring consistency across complex calculations.

TECH STACK:

- ➢ React.js ⟶ Complete Application
- ➢ Reactflow.js ⟶ Cache Simulator
- ➢ Bootstrap CSS ⟶ Styling of the Application
- ➢ Chart.js ⟶ Graphical Representation of Performance Metrics

---

INDIVIDUAL CONTRIBUTION

Bhavik Patel:

⟹ Implemented Number Convertor and worked on IEEE Floating Convertor.

Hitesh Kumar:

⟹ Implemented Performance Metric Analyser and worked on Cache Simulator.

Jinil Patel:

⟹ Implemented Cache Simulator.

Ruchit Jagodara:

⟹ Implemented IEEE Floating Convertor.

OTHER KEY TAKEAWAYS

ERROR HANDLING AND COMPONENT REUSABILITY IN REACT:

Resolved common errors in React Flow, ensuring valid node IDs and proper provider usage. This experience was valuable for enhancing the robustness of the application. Focused on reusable components for easier maintainability and scalability, a crucial practice in software development.

EXPERIMENTATION WITH PARALLELISM AND SEQUENTIAL EXECUTION:

The application enabled side-by-side comparisons of parallel versus sequential instruction execution, helping users grasp the impact of parallelism on system performance. Interactive models show how parallel processing can enhance efficiency and highlight challenges such as dependency management in multi-threaded systems.

FUTURE PROSPECTS

ADVANCED PERFORMANCE ANALYSIS:

Aiming to incorporate more detailed tools for performance analysis and visualisation with the end goal of providing in-depth knowledge to the user in the field of Computer Organization and Architecture.

EXPANDING CACHE SIMULATION TO L2 AND L3 LEVELS:

The goal is to enhance the Cache Simulator by adding simulations for L2 Cache and, if possible, L3 Cache. Users can explore caching at a deeper level within the memory hierarchy by including multiple levels of cache. This extension will help illustrate how each cache layer impacts data access speed and overall system performance, giving users a more comprehensive understanding of memory optimisation.

EXTENSIBILITY:

Enable tool customisation, allowing users to add their own Performance Metric formulas and show the results. This will result in a better user experience by providing a platform to analyse new metrics quickly.

You can access the site here. (Best Experience on Laptop)
