

Project Proposal

1 Group Information

Group Number: 1

Member List: Nikita Aleksii, Robby Votta, Yurdanur Yolcu, and Maria Fajardo

2 Introduction

As a team, we aim to develop AssemblyViz, a web application designed to modernize the interface of the assembly simulator currently used in Davidson College's Computer Organization (CSC 250). This project originated from a need within our environment. Dr. Mendes, currently teaching CSC 250, where Assembly is actively used, believes that the current application's visibly old look creates a bias or hesitation from students to use it. Additionally, the current application doesn't contain a graphical component, which creates a major hardship for the students' learning process. He is motivated by guaranteeing clarity and motivation for his students to succeed in the course and with Assembly code.

The application will replicate the simulator's core functionality while allowing users to write various types of assembly code and visualize the execution of a virtual machine through a graphical interface. The simulator will support self-modifying code, which is a feature not commonly available in modern assembly environments. Students typically begin with this system before transitioning to contemporary tools, which creates a gap between the educational model and modern assembly development practices.

2 Novelty

While the existing software, SimHYMN, allows assembly code visualization, it lacks clarity and functionality (<https://cburch.com/socs/hymn/index.html>). Mac users complained that it was hard to install the application because the system always asked for permission. Moreover, almost every student in CSC 250 complained about the eye-searing design that created a mental bias against programming in assembly. Also, the software doesn't track memory history and does not provide debugging functionality, which makes students' lives harder since they can't pinpoint the problem.

Additionally, Dr. Mendes showed a great interest in teaching CSC 250 using a novel instruction set architecture (ISA), called RISC-V. There's some existing software (<https://peterhigginson.co.uk/RISC/>) that lacks the same features—clarity and functionality.

For our project, we will develop a web application that incorporates the functionality of existing software and adds a history of memory cells and a debugger. We will also implement RISC-V visualization and keep it in the same web application, so that students and professors have a choice of which ISA they want to use.

3 Customer Needs

Our primary customer is Dr. Mendes from the Davidson College Computer Science Department. He came up with the idea to redesign SimHYMN for his students. He is one of the drivers of the project and dictates the direction we will take. He is a primary stakeholder because over the next few weeks, he will be involved in the process and execution of the project, and he wants this project to succeed in order to reach his goals. Additionally, our AssemblyViz team is also an internal stakeholder because we are responsible for building, testing, and ensuring the success of our project.

Our secondary stakeholders are all those who are impacted, benefited, or who have an indirect influence on the project. In this case, we have the users as one of our external stakeholders because they will directly use the software and benefit from its redesign, but are not involved in the building process (other than testing). They drive our end goal. For us, the overall experience should feel clear and intuitive. A start page with all the aspects laid out, but nothing is yet filled in or predetermined; the user chooses. Clear instructions and help are laid out. An option for a walkthrough is given to the user. Clear feedback when the program is running. The overall experience should be of the user always being situated and knowing what is happening.

3.1 User Requirements

Write at least 5 SMART user stories based on the stakeholders' needs and wants:

1. **As a CS Professor I want** to clearly step through students' code one instruction at a time and visualize said steps in the memory and register **so that** I can efficiently identify errors during office hours.
2. **As a CSC 250 student I want** to be able to run through my code in AssemblyViz one instruction at a time **so that** I can understand and visualize how my code works step by step.
3. **As a CSC 250 student I want** the simulator to give me an instant error message **so that** I can correct the mistake while I am editing.

4. As a CSC 250 student I want to interact with a clean and divided interface so that I can locate the editor, the memory, and the register and use each of them accordingly.
5. As a Developer I want the three-view simulator to display and update independently so that I can still edit my program with conformity, and simulate it when I need to.

3.2 Acceptance Tests

1. Given a student's assembly program is loaded into the simulator and the program has not yet been executed when the professor steps through the program one instruction at a time then the current instruction is clearly shown and the outputs of each instruction are visible.
2. Given a student has written assembly code in the simulator when the student runs the program step by step then the program executes one instruction at a time and the student can see how the program state changes at each step.
3. Given a student is running their assembly code when there is an error in
4. Given a student opens AssemblyViz when the main interface is displayed then the 3-part layout is clear and easy to understand and the student can identify where to write code and run the program.
5. Given the developer is working on one view of the simulator when changes are made to that view then the other views continue to function as expected.

4 Project Goals

4.1 Customer Problems and Benefits

What customer problem have you chosen to address?

- In implementation-free terms, what user benefit will the system provide?
- How will the benefit support the customer's desired overall experience?

Our primary goal is to provide students with software that is easier to work with and visualize, accelerating the learning process of assembly code. Currently, students spend valuable time wrestling with confusing and finicky software when they should be focused on solving problems with assembly code itself. To address this, we are developing a more accessible web-based platform that eliminates the need for downloads and provides clear visual feedback on

code execution. Our tool will also aid in the critical transition between using HYMN assembly and RISC-V, bridging the gap between these two important learning stages.

4.2 Measure of Success

- Who outside the team have you tested the idea on?
- How will you know whether the customer got their desired benefits?
- What are your customer-centric measures of success?

We will test AssemblyViz with our client, Dr. Mendes, our class professor, Dr. Lim, current CSC 250 students, past CSC 250 students, and ETs to ensure the tool meets its educational requirements. Our goal is that students can write their assembly code, press the run button, and watch it execute line by line through the instructions. If it works: great, if it doesn't, they can clear the memory and try again. We want a tool that is easy to navigate and easy to visualize data. We want a clear improvement over HYMN.

To measure success through customer-centric criteria, we will do a demo or test with current CSC250 students and run a Qualtrics survey to look at quantitative data (scores, accuracy, etc.) and qualitative data (feelings regarding the website use and user sentiment). Some questions to ask on the survey are:

- Students can predict the next state before stepping
- Whether they complete assignments faster
- Whether they report a better understanding
- Whether there are fewer office hours, questions about basic concepts of the software itself, or questions about the assignment.
- Preference metrics (X% prefer AssemblyViz over HYMN),
- Usage metrics (students use the tool for an average of X minutes per assignment)
- Understanding metrics (X% reduction in "I don't understand what's happening" questions).

We will survey students to capture the overall opinion on AssemblyViz compared to HYMN in class. We can do two demos. One at the beginning, and provide students with the website so they can complete the survey after class, but also have access to it for their own purposes. Then, later on, another survey will be conducted after they've had the time and space to use it, so we can measure which one they prefer more and if our tool is meeting all expectations.

5 System Description

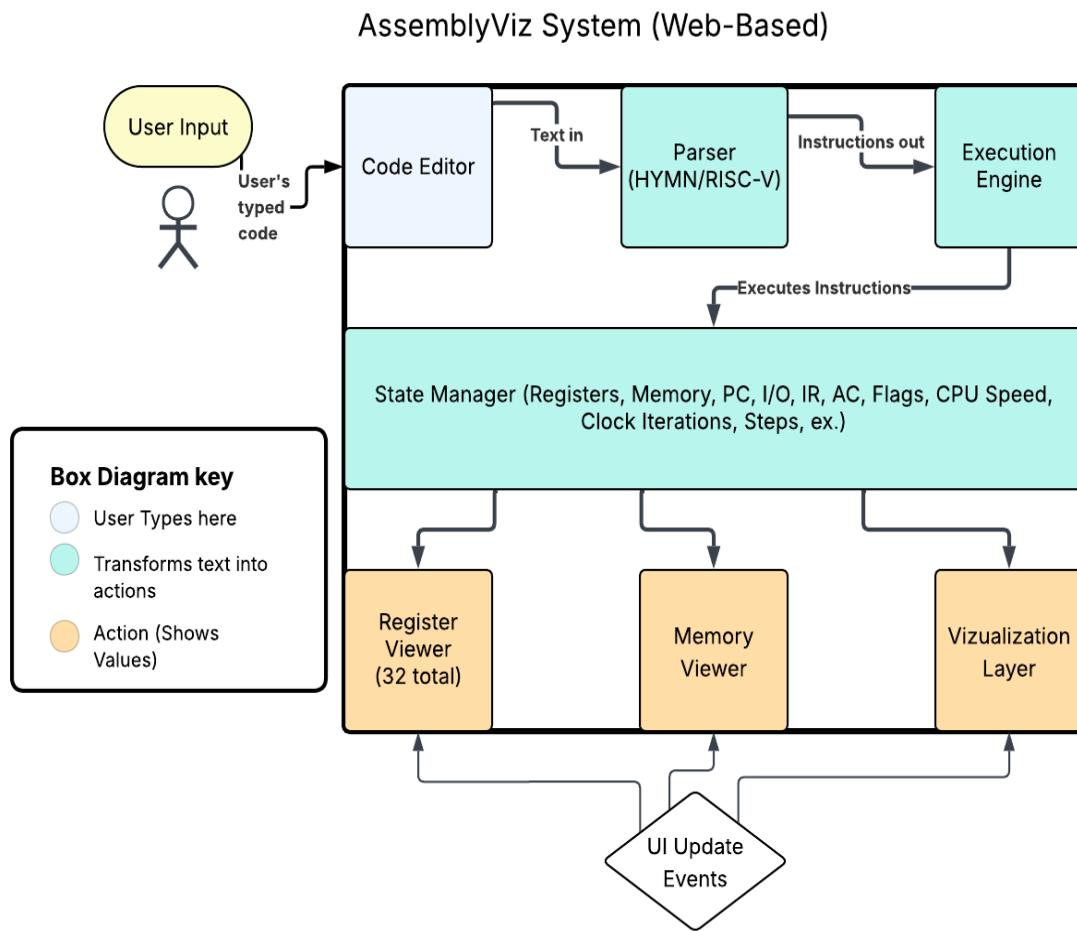


Figure 1: Block Diagram of the General System Design we plan to implement for AssemblyViz.

Our system will need to take typed-out code from students and translate it into the proper visualization based on the assembly language code instructions written. The Code Editor is where students type out their assembly instructions. It will include line numbers, Basic editing features (copy/paste, undo/redo), and Error indicators. This is the first point of interaction with the software. This will be integrated on the front end via React. The parser component converts the assembly text into a tokenized, structured format for the execution engine to understand. It will take a string and turn it into an array of instructions understood by the HYMN/RISC-V instruction set architectures. The execution engine is the CPU simulator that executes the assembly instructions virtually. It will fetch an instruction token, decode it, perform the operation

it is associated with, then move on to the next token. This part MUST be accurate in order for the project to see any sort of success. The State manager will coordinate updates between each of the visual components of the website. It will notify the UI of which components change from registers to memory to PC, and so on. This will make debugging much easier. The display components will show the actions from the execution machine to the user. It will show the data moving from the registers to memory for each line of code. Each of the three components (register viewer, memory viewer, and the visual layer) will update the state manager after they have completed a line of code.

6 Solution Approach

Our project will be used as a web-based assembly simulator emphasizing clarity and visualization. As detailed by Dr. Mendes, he would like the experience of the system to be as follows: when the student opens the webpage, the memory starts with all 0s. The student will write code on the code panel, they will declare data, the positions in the 32-bit spaces that will have each number and its initial position. When they are done, they can press the run button. The visualizer will go line by line, clearly indicating its position, and go through each instruction. There is the option to clear memory and retry.

To implement the back-end logic, we will be using TypeScript to code an assembler. For the front end, we will be using HTML, CSS, and TypeScript with React, which helps with the responsiveness of the interface. We will use Figma to create the prototypes. We will test and measure how users are responding to the project. Some tasks we might test include typing and running the code step by step, being able to observe changes in the memory, and restarting the simulation. We also want users to comment on the look and interactivity of the product. We want them to tell us how easy or hard it was to navigate by themselves. Since we will be using an iterative approach, testing will allow us to look at the feedback and edit our code and program to better adjust each time to positive and negative responses.

7 Project Management

For this project, we will use the Agile approach to the Scrum framework. We will always be aware of the 4 main values of Agile. We are dedicated to putting out an interface that will be specifically used by Davidson Computer Science students. So, the interface redesign and visualization features will likely change based on testing and feedback. Scrum supports short development cycles (sprints) that allow continuous refinement.

Sprint 1: Foundation & Architecture (Proposal – Report 1)

Objective: Establish the core application infrastructure for AssemblyViz

Deliverables:

- Complete project setup

- Front-end framework with basic UI shell and navigation structure

- Data models for assembler components (instructions, registers, memory)

- Technical architecture documentation

Out of scope: No assembler functionality implemented yet

Sprint 2: HYMN Assembler Implementation (Report 1 – Report 2)

Objective: Build a fully functional HYMN assembler with visualization capabilities

Deliverables:

- HYMN instruction parser and assembler

- Memory visualization with execution history

- Register display and tracking

- Interactive debugger

Out of scope: RISC-V support

Sprint 3: RISC-V Assembler Implementation (Report 2 – Final)

Objective: Support RISC-V assembly language

Deliverables:

- RISC-V instruction parser and assembler (RV32I base instruction set)

- Memory visualization, register display, and debugger matching HYMN functionality

- Architecture selection toggle (switch between HYMN and RISC-V modes)

- Final documentation and testing

Out of scope: Advanced RISC-V extensions beyond base integer instructions

8 Team Management

8.1 Roles

As Scrum proposes, we will have one Scrum Master and one Product Owner in our team to create two opposite forces to propel the project while protecting the team's well-being. Nikita will be the scrum master, and Maria will be the product owner. Yurdanur, Robby, Maria, and Nikita will be developers. As a small team in a non-corporate environment, we find it important to assess roles and responsibilities suitable for our team members' characters and schedule.

This role distribution will allow us to maintain effective communication and accountability while remaining flexible to individual availability. Regular Scrum ceremonies and a shared task board will be used to track progress, manage priorities, and

adapt to challenges throughout the project lifecycle. We hope to finish this project productively and efficiently for our user Students of CSC250 – Comp. Organization.

8.2 Scheduling

As a team, we decided to meet approximately once a week in person, while giving some flexibility according to our schedule. We believe that since all of us are college students with different schedules, leaving room for little changes will protect our well-being. We will meet with Dr. Mendes biweekly in person. We are aware his schedule is packed and ready to make moderate changes throughout the project. We communicated it with Dr. Mendes, and he is willing meet with our team regularly.

8.3 Background

Yurdanur Yolcu: is proficient in HTML, CSS, JavaScript, Java, and Python. She has experience in Data Visualization and hopes to implement those skills for the design of this project. She has not taken CSC 250 - Computer Organization class, so she will provide an outside point of view to the project.

Nikita Aleksii: has experience in using Python, HTML, CSS, and JavaScript–created an interactive website for a non-government organization. He took the Computer Organization course last semester and is taking the Computer Architecture course, which goes through the RISC-V architecture. Interested in the fundamentals of computing, including but not limited to SoC devices.

Maria Fajardo: has experience with Python and Java, but is mostly interested in Human-Computer Interaction and using HTML, CSS, and JavaScript to create a clear and intuitive user experience. She's taken web design classes that use different tools such as Node, React, and Express in order to help with the Front-End aspect. She is currently enrolled in CSC 250 and is hoping to learn about the current program being used and how this one can help more students.

Robby Votta: languages - Python, Java, C, Solidity. Classes: Data structures, Comp Org, Image Processing. Experiences - Public data wrangle Hack-a-thon in R 2025, MBC Auto-Pilot Hack-a-thon. I have only a little experience using HTML, CSS, or React, but it will be helpful on the back-end, implementing HYMN assembly software onto our web-based software in Python.

9 Constraints and Risks

This project does not represent any social, ethical, policy, or legal constraints. Given that the nature of AssemblyViz is strictly educational, it is completely safe and does not collect data or any other sensitive information from its users. We will have all the necessary information and resources to successfully complete the project. We might require faculty assistance when approaching this challenge, and also learn how to use some more advanced web development techniques. In that case, Professor Owen Mundy from the Digital Studies department has some tools that can help us. We can host our project using GitHub Pages for now, but when we are ready to make it public, Davidson Domains can help us with hosting it. We will be using the documents provided to the CSC 250 Computer Organization class to be able to run our tests and base our logic. We are confident we will have access to students currently in the class to be able to test and gain feedback on it. Our user tests will not collect any important data on the students, so there is no risk there.

10 Values

- What are the important principles (≥ 3) that will guide your team?
 - For each principle, write a short description of what the principle means. For example,
Simplicity over complexity: We value simple and understandable solutions. When multiple approaches are possible, we prefer the one that is easier to maintain, test, and explain.
 - Justify why each principle makes sense for your team and project.
1. **Working application over extensive documentation:** we value the usability and applicability of our product in real-world applications, particularly in teaching. Thus, we focus more on developing the application itself rather than extensively documenting our steps. This principle is particularly important for our team since we cannot afford bugs or glitches in a teaching tool for students.
 2. **User-friendliness over complexity:** We value user-friendliness since it is important for college students to understand how to use an app without spending too much time. The existing software with complex solutions provides confusing instructions and design, while we aim to create a simple tool that is easy to understand. Therefore, user-friendliness is essential for the success of students.
 3. **Quality over quantity:** we believe maintaining the highest quality of the product is more important than including every feature. Otherwise, unanticipated problems might occur, which will downgrade the quality of the student experience. Thus, the team must prioritize high standards of quality over quantity.

11 GitHub

Follow the instructions below:

1. One person creates a GitHub repository.
2. Add other members to the repository as collaborators.
3. Update your README.md file with the title of your project, your group number, and the group members with their roles. Example below:
4. Add me (username: hlim1) as a collaborator to the repository.
5. Create a folder called “Reports” in the main branch.
6. In the folder, add this proposal. You will add all the reports to this folder, in addition to submitting them to Moodle.
7. Take a screenshot of the ‘Collaborators’ page and add it to this section.

The screenshot shows the 'Collaborators' page on GitHub. At the top, there is a 'Select all' checkbox, a 'Type' dropdown, and a search bar with placeholder text 'Find a collaborator...'. Below this, a table lists four users:

User	Status	Action
heyitsyurda	Collaborator	
HeuiChan (Terrence) Lim	Pending Invite	
Maria A. Fajardo	mafajardo23 · Collaborator	
Robby Votta	rovotta · Collaborator	