UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS

Programa de Ingeniería de Sistemas y Computación

Extensión para Visual Studio Code como simulador para procesador RISC-V

Autor:
Amir Evelio Hurtado Mena
Código: 1077997025
Email: a.hurtado@utp.edu.co

Director:
(Nombre del Director)

Área Temática: Ingeniería de Software
Línea de Investigación: Desarrollo de Software Aplicado

Modalidad: Proyecto de Aplicación

Pereira, Colombia
17 de octubre de 2025

Índice

1. Introducción

La Arquitectura de Computadores es una materia fundamental en la carrera de Ingeniería de Sistemas y Computación, ya que permite entender cómo funciona el hardware que ejecuta todo el software que creamos. Sin embargo, los conceptos sobre el funcionamiento interno de un procesador, como los registros, las unidades de control y los ciclos de instrucción, suelen ser muy abstractos y dificiles de asimilar para los estudiantes que cursan la asignatura en la Universidad Tecnológica de Pereira.

Esta dificultad a menudo genera un obstáculo en el proceso de aprendizaje, ralentizando el avance de las clases y dejando vacíos conceptuales en los futuros profesionales. Para abordar este problema, este proyecto propone el desarrollo de una herramienta de software: una extensión para el editor de código Visual Studio Code que funciona como un simulador gráfico e interactivo de un procesador con arquitectura RISC-V, tanto en su versión monociclo como segmentada.

El objetivo es ofrecer a estudiantes y docentes un recurso que traduzca las operaciones complejas del procesador en visualizaciones claras y fáciles de seguir. De esta manera, se busca fortalecer la comprensión de los temas clave de la materia, haciendo el aprendizaje más práctico, intuitivo y efectivo, y mejorando así la calidad de la formación de los ingenieros de la universidad.

2. Planteamiento del problema

2.1. Antecedentes (Contextualización del problema)

La enseñanza de la Arquitectura de Computadores presenta desafios pedagógicos bien documentados en la literatura académica. Uno de los principales retos identificados es la dificultad para conectar la teoría con la práctica (Bridging Theory and Practice). Los estudiantes a menudo luchan por trasladar el conocimiento teórico a una aplicación tangible, lo que dificulta la comprensión de cómo se ejecutan los programas a nivel de sistema. Para abordar esto, varios autores proponen el uso de sistemas interactivos y laboratorios de aprendizaje asistido por computador (CAL) que permiten a los estudiantes visualizar la ejecución de programas e inspeccionar los detalles de implementación hasta el nivel de transferencia de registros ([1], [6]).

Otro desafio clave es la comprensión conceptual (Conceptual Understanding) de sistemas que son complejos e intangibles. Investigaciones señalan que muchos estudiantes, incluso de ciencias de la computación, tienen ideas erróneas fundamentales sobre cómo funcionan realmente los programas, un problema que requiere atención desde el diseño del currículo ([8]). En este contexto, el uso de herramientas de visualización de la ejecución se ha propuesto como una solución efectiva para ayudar a los estudiantes a comprender estos sistemas ([6]).

Por lo tanto, la literatura académica no solo confirma la existencia de estas barreras de aprendizaje, sino que también respalda el desarrollo de herramientas interactivas y visuales, como la propuesta en este proyecto, como un camino viable para mejorar significativamente la experiencia educativa en esta área fundamental.

2.2. Causas (qué está causando el problema)

La causa principal del problema en la asignatura de Arquitectura de Computadores de la UTP es la brecha existente entre la teoría abstracta enseñada en clase y la falta de herramientas prácticas que permitan a los estudiantes visualizar y experimentar con dichos conceptos. Esta situación fue validada a través de una encuesta realizada a 57 estudiantes del programa que ya habían cursado la materia. Los resultados revelan la magnitud del desafío: un contundente 86 % de los estudiantes calificó el proceso de comprensión de la arquitectura de un computador y su implementación en hardware como Çomplejo.º "Muy complejo", lo que demuestra que no se trata de una percepción aislada, sino de una dificultad generalizada en el alumnado (Ver Gráfico 1).

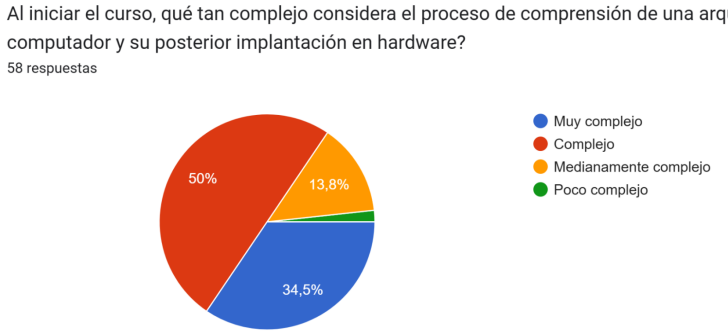

Figura 1: Resultados sobre la complejidad percibida.

Más importante aún, la encuesta también indagó sobre la solución a esta dificultad. Al preguntarles sobre la necesidad de una nueva herramienta, la respuesta fue abrumadora: un 91.1 % de los estudiantes (sumando las categorías "Muy necesarioz "Necesario") afirmó que la inclusión de una herramienta tecnológica para visualizar los conceptos sería clave para un mejor aprendizaje (Ver Gráfico 2). Estos datos demuestran que la causa del problema no es una falta de interés por parte del alumnado, sino una necesidad directa y explícita de un recurso didáctico interactivo que les permita conectar los conceptos abstractos con una representación visual y práctica. La herramienta propuesta en este proyecto busca, por lo tanto, atacar directamente esta causa raíz.

Figura 2: Necesidad de una herramienta tecnológica según los estudiantes.

2.3. Definición del problema

Los estudiantes de la asignatura de Arquitectura de Computadores del programa de Ingeniería de Sistemas y Computación de la UTP presentan dificultades significativas para comprender los procesos internos de un procesador (monociclo y segmentado).

2.4. Consecuencias (de no resolver el problema)

De no abordarse este problema, las consecuencias afectan a múltiples niveles. Para los estudiantes, implica una base conceptual débil que puede perjudicar su desempeño en materias posteriores que dependen de estos conocimientos, además de generar frustración y desinterés por el área del hardware. Para los docentes, significa tener que invertir más tiempo en reforzar conceptos básicos, ralentizando el avance del temario y limitando la posibilidad de profundizar en temas más avanzados. A largo plazo, para el programa académico, esto representa un riesgo en la calidad de la formación, ya que sus egresados podrían tener vacíos en un área fundamental de la ingeniería de sistemas, lo que podría impactar negativamente su perfil profesional en el mercado laboral.

3. Justificación

La realización de este proyecto se justifica plenamente por su alto impacto en el proceso formativo de los estudiantes de Ingeniería de Sistemas y Computación de la UTP, así como por los aportes que ofrece tanto a los docentes como al programa académico. A continuación, se detallan los beneficios desde distintas perspectivas:

3.1. Justificación Pedagógica y Académica

Este proyecto ataca directamente una barrera de aprendizaje validada. Como lo demostró la encuesta, el 86 % de los estudiantes considera complejo el proceso de comprensión de la materia, y un 91.1 % reclama una herramienta tecnológica para facilitar este proceso. Por lo tanto, el principal beneficio es para el estudiante, quien contará con un recurso interactivo diseñado para:

- Transformar lo abstracto en tangible: Permitirá visualizar el flujo de datos, el estado de los registros y la ejecución de instrucciones paso a paso, haciendo que los conceptos teóricos cobren vida.
- Fomentar el aprendizaje autónomo: Los estudiantes podrán experimentar y probar código a su propio ritmo, reforzando los conocimientos vistos en clase y resolviendo dudas de forma práctica.
- Aumentar la motivación y reducir la frustración: Al facilitar la comprensión, se espera un mayor interés por una de las áreas más fundamentales y áridas de la carrera.

Para los docentes, la herramienta se convertirá en un valioso recurso que les permitirá explicar temas complejos de manera más eficiente y gráfica, optimizando el tiempo en el aula y asegurando un nivel de comprensión más homogéneo en el grupo.

3.2. Justificación Tecnológica

Desde el punto de vista tecnológico, el proyecto es relevante por el desarrollo de una solución de software específica y moderna. La creación de una extensión para Visual Studio Code, uno de los editores de código más utilizados en el mundo profesional, asegura que la herramienta se integre de forma natural en el entorno de trabajo de los estudiantes. Además, el enfoque en la arquitectura RISC-V, un estándar abierto y en pleno auge, garantiza que el conocimiento adquirido a través del simulador sea actual y pertinente para el futuro de la industria del hardware.

3.3. Justificación Institucional

Para el programa de Ingeniería de Sistemas y Computación de la UTP, este proyecto representa una oportunidad para fortalecer sus métodos de enseñanza. La implementación de esta herramienta demuestra un compromiso con la mejora continua de la calidad educativa y con la atención a las necesidades de sus estudiantes. En resumen, este trabajo no solo resuelve un problema académico concreto y medido, sino que también aporta una solución tecnológica moderna y fortalece el ecosistema de aprendizaje del programa, generando un beneficio claro para toda la comunidad universitaria involucrada.

4. Objetivos

4.1. Objetivo General

Desarrollar una extensión de software para Visual Studio Code que simule de forma gráfica y textual el funcionamiento de un procesador RISC-V con arquitecturas monociclo y segmentada, para facilitar la comprensión de los conceptos fundamentales de la asignatura Arquitectura de Computadores.

4.2. Objetivos específicos

1. Elaborar el documento de especificación de requisitos funcionales y no funcionales del simulador, detallando las características de visualización necesarias para las arquitecturas RISC-V monociclo y segmentada. **Entregable:** Documento de Especificación de Requisitos.

2. Diseñar la arquitectura de software de la extensión para Visual Studio Code y los prototipos de la interfaz de usuario (mockups), garantizando que la interacción sea intuitiva y responda a los requisitos definidos. **Entregable:** Documentos de diseño de arquitectura y prototipos/mockups de la interfaz.

3. Construir los módulos funcionales del simulador según el diseño establecido, implementando la carga de código, la ejecución paso a paso y la visualización en tiempo real de los componentes del procesador. **Entregable:** Código fuente de la extensión (versión funcional).

4. Ejecutar un plan de pruebas funcionales para verificar el correcto funcionamiento del simulador y realizar pruebas piloto con un grupo de usuarios (estudiantes y docentes) para evaluar la usabilidad y pertinencia pedagógica de la herramienta. **Entregable:** Informe de resultados de pruebas funcionales y de las pruebas piloto.

5. Metodología

5.1. Marco Metodológico

Para el desarrollo de este proyecto de aplicación, se empleará una Metodología de Desarrollo de Software Basada en Prototipos. Esta metodología es ideal para proyectos donde la interfaz de usuario y la experiencia de interacción son críticas para el éxito, como es el caso de esta herramienta educativa. El enfoque se centra en construir versiones funcionales tempranas (prototipos) que pueden ser evaluadas por los usuarios finales (estudiantes y docentes) para validar y refinar los requisitos desde el inicio del proceso, evitando así errores y malentendidos en fases posteriores del desarrollo ([10]).

El proceso se dividirá en cuatro fases principales, que se alinean directamente con los objetivos específicos del proyecto:

5.1.1. Fase 1: Definición y Especificación

En esta etapa inicial, se realizará el levantamiento y la documentación de los requisitos. Las actividades incluyen:

- **Estudio conceptual:** Se analizarán los temas clave de las arquitecturas RISC-V monociclo y segmentada que presentan mayor dificultad, basándose en la bibliografia de la asignatura y los resultados de la encuesta inicial.
- **Documentación:** Se elaborará el documento de especificación de requisitos, detallando las funcionalidades (ej. carga de código, ejecución paso a paso) y los requisitos no funcionales (ej. rendimiento, diseño responsivo).

5.1.2. Fase 2: Diseño y Prototipado

Con los requisitos claros, se procederá a diseñar la solución.

- **Diseño de Arquitectura:** Se definirá la estructura interna de la extensión, los componentes de software y cómo se comunicarán entre sí.
- **Diseño de Interfaz (UI/UX):** Se crearán los prototipos visuales y de interacción (mockups) de la herramienta utilizando software de diseño como Figma. Estos prototipos se centrarán en intentar tener una experiencia intuitiva.

5.1.3. Fase 3: Construcción e Implementación

Esta es la fase de codificación.

- **Configuración del entorno:** Se preparará el ambiente de desarrollo para extensiones de Visual Studio Code.
- **Desarrollo incremental:** Se construirán las funcionalidades del simulador de forma modular, empezando por el núcleo (la simulación del procesador) y luego la capa de visualización.

5.1.4. Fase 4: Validación y Evaluación

En esta última fase se verificará que la herramienta cumple con lo esperado.

- **Descripción de la unidad de muestreo:** La ünidad de muestreo"serán los participantes de las pruebas piloto. Se seleccionará un grupo de estudiantes que estén cursando, y al menos un docente del área.
- **Descripción del "tratamientoz su aplicación:** El "tratamiento"será el uso guiado de la extensión de Visual Studio Code. A los participantes se les entregarán ejercicios prácticos y se les pedirá que utilicen el simulador para resolverlos.

- **Información a registrar:** Se recolectará información cualitativa y cuantitativa a través de observación directa, encuestas de satisfacción y reporte de errores.
- **Análisis de la información:** Los resultados de las pruebas se analizarán para identificar puntos de mejora en la herramienta y para redactar el informe final que valida el cumplimiento del proyecto.

5.2. Actividades

A continuación, se desglosan las actividades principales derivadas de cada objetivo específico:

- **Objetivo 1: Elaborar el documento de especificación de requisitos...**
  - Actividad 1.1: Investigar y sintetizar los conceptos de RISC-V monociclo y segmentado más relevantes para la simulación.
  - Actividad 1.2: Redactar la primera versión del documento de requisitos funcionales y no funcionales.
  - Actividad 1.3: Validar el documento de requisitos con el director del proyecto.
- **Objetivo 2: Diseñar la arquitectura de software...**
  - Actividad 2.1: Crear el diagrama de la arquitectura de la extensión.
  - Actividad 2.2: Diseñar los mockups y el flujo de usuario de la interfaz gráfica.
  - Actividad 2.3: Documentar las decisiones de diseño.
- **Objetivo 3: Construir los módulos funcionales...**
  - Actividad 3.1: Desarrollar el motor de simulación para la arquitectura monociclo.
  - Actividad 3.2: Desarrollar el motor de simulación para la arquitectura segmentada.
  - Actividad 3.3: Implementar la interfaz gráfica y su conexión con el motor de simulación.
- **Objetivo 4: Ejecutar un plan de pruebas funcionales...**
  - Actividad 4.1: Elaborar el plan de pruebas y los casos de uso para la prueba piloto.
  - Actividad 4.2: Seleccionar y convocar a los participantes para la prueba.
  - Actividad 4.3: Ejecutar las sesiones de la prueba piloto.
  - Actividad 4.4: Recolectar y analizar los resultados de las pruebas.
  - Actividad 4.5: Redactar el informe final de validación.

6. Aportes del Proyecto

- **Software Funcional:** El principal aporte es la extensión para Visual Studio Code, una herramienta de software completamente funcional para la simulación gráfica e interactiva de un procesador RISC-V en sus arquitecturas monociclo y segmentada. Este software quedará a disposición del programa de Ingeniería de Sistemas y Computación para su uso libre en la asignatura de Arquitectura de Computadores.
- **Documentación Técnica del Proyecto:** Se entregarán todos los documentos que respaldan el proceso de ingeniería de software realizado, los cuales incluyen: el Documento de Especificación de Requisitos, los Documentos de Diseño y el Informe de Pruebas y Validación.
- **Manual de Usuario:** Se elaborará un documento guía que explique de manera clara y sencilla cómo instalar, configurar y utilizar la extensión de software, dirigido tanto a estudiantes como a docentes.

7. Marco Referencial

7.1. Marco Teórico

En este capítulo se definen los conceptos técnicos fundamentales que sustentan el desarrollo del proyecto. Se explican las tecnologías y arquitecturas clave necesarias para comprender tanto el problema abordado como la solución propuesta.

7.1.1. Arquitectura RISC-V

RISC-V es una arquitectura de conjunto de instrucciones (ISA) basada en los principios de un computador con un conjunto reducido de instrucciones (RISC). A diferencia de otras arquitecturas comerciales, RISC-V es completamente gratuita y de código abierto, lo que permite una amplia adopción y personalización ([7]). Su diseño es inherentemente modular, lo que significa que consiste en un conjunto base de instrucciones simples al que se le pueden añadir extensiones para funcionalidades específicas. Esta flexibilidad la hace ideal para una gran variedad de aplicaciones, desde pequeños sistemas embebidos hasta computadores de alto rendimiento ([5]).

7.1.2. Arquitecturas de Procesador

Para ejecutar las instrucciones de una arquitectura como RISC-V, un procesador debe seguir un modelo de implementación. Los dos modelos fundamentales para la enseñanza de la arquitectura de computadores son el monociclo y el segmentado.

Procesador Monociclo (Single-Cycle) Este diseño ejecuta cada instrucción en un único y largo ciclo de reloj. Su principal ventaja es la simplicidad, lo que facilita su comprensión inicial. Sin embargo, su rendimiento es bajo, ya que la duración del ciclo de reloj debe ajustarse a la instrucción más lenta, resultando en una menor frecuencia de operación y eficiencia ([3]).

Procesador Segmentado (Pipelined) Esta arquitectura mejora drásticamente el rendimiento al dividir la ejecución de una instrucción en múltiples etapas más cortas que operan en paralelo, de forma similar a una línea de ensamblaje. El modelo clásico para RISC-V se divide en cinco etapas: 1. Búsqueda (Fetch), 2. Decodificación (Decode), 3. Ejecución (Execute), 4. Memoria (Memory) y 5. Escritura (Write Back) ([4]). Al permitir que varias instrucciones se procesen simultáneamente en diferentes etapas, se logra un rendimiento mucho mayor que en el diseño monociclo. No obstante, esta paralelismo introduce complejidades como los riesgos"(hazards), que deben ser gestionados para asegurar el correcto funcionamiento ([3]).

7.1.3. Extensiones de Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft que se ha convertido en un estándar en la industria. Su poder reside en su ecosistema de extensiones. Estas son aplicaciones que se instalan dentro del editor para añadir nuevas funcionalidades, como soporte para lenguajes, herramientas de depuración o, como en este proyecto, paneles visuales e interactivos. Esta capacidad de extensión permite crear herramientas educativas que se integran directamente en el entorno de trabajo que los estudiantes ya utilizan.

7.2. Estado del Arte

A continuación, se revisan otras herramientas de software que existen para enseñar arquitectura de computadores. Esto ayuda a entender por qué este proyecto es diferente y necesario.

Muchos de los simuladores que existen hoy en día son muy completos, pero asumen que el estudiante ya sabe bastante sobre el tema. Por ejemplo, herramientas como MARS (MIPS Assembler and Runtime Simulator), que es muy usada para enseñar la arquitectura MIPS, tienen una interfaz llena de información que puede confundir a un principiante ([9]). Igualmente, programas más nuevos para RISC-V como VRV o la página web WebRISC-V, son muy potentes, pero para usarlos bien, el estudiante ya necesita entender qué significa cada cosa que ve en la pantalla ([2]).

El problema principal de estas herramientas no es lo que hacen, sino lo difícil que es aprender a usarlas. Están hechas para que alguien pueda revisar y encontrar errores, no para que alguien que no tiene buenas bases pueda entender las ideas desde cero. Así, el estudiante tiene dos trabajos: entender los conceptos dificiles de la materia y, además, aprender a manejar un programa complicado.

Este proyecto fue creado para solucionar ese problema. Nuestra propuesta es diferente porque le da más importancia a que sea fácil de usar y muy visual, en lugar de llenarla de funciones complejas. El objetivo no es hacer otro simulador para expertos, sino una herramienta para empezar, donde la interacción te guíe para aprender. Con una interfaz gráfica interactiva y pensada para principiantes, un estudiante podrá, sin mucha explicación previa, cargar un programa y ver de forma clara cómo se mueven los datos, qué registros cambian y qué partes del procesador se activan en cada paso.

En resumen, mientras las herramientas que ya existen responden a la pregunta "¿Mi código está bien?", esta propuesta busca responder la pregunta que realmente tiene el estudiante: "¿Cómo funciona un procesador internamente?".

7.3. Marco Conceptual

A continuación, se definen los términos clave utilizados a lo largo de este documento para establecer un entendimiento común del dominio del problema y la solución propuesta.

- **Simulador:** Software que imita el comportamiento y el funcionamiento interno de un sistema real, en este caso, un procesador de computador.
- **Extensión de VS Code:** Un pequeño programa que se instala dentro del editor de código Visual Studio Code para añadirle nuevas herramientas o funcionalidades, como el simulador propuesto.
- **RISC-V:** Una arquitectura de conjunto de instrucciones (ISA) de código abierto. Es el "manual de reglas"que le dice al procesador qué operaciones puede realizar y cómo están codificadas.
- **Arquitectura Monociclo:** Un diseño de procesador en el que cada instrucción se ejecuta por completo en un único ciclo de reloj. Es un modelo simple, ideal para fines educativos.
- **Arquitectura Segmentada (Pipeline):** Un diseño de procesador que divide la ejecución de una instrucción en varias etapas. Permite que múltiples instrucciones se procesen al mismo tiempo, mejorando significativamente el rendimiento.
- **Ruta de Datos (Datapath):** Conjunto de componentes de hardware dentro del procesador (como la ALU y los registros) que realizan las operaciones y procesan los datos.
- **Registros:** Pequeñas unidades de memoria de alta velocidad ubicadas dentro del procesador que se utilizan para almacenar temporalmente datos e instrucciones durante la ejecución.
- **Unidad de Control:** El componente del procesador que lee las instrucciones del programa y genera las señales de control para dirigir las operaciones de la Ruta de Datos.

- **Visualización Interactiva:** La representación gráfica del estado interno del procesador (registros, memoria, etc.) que se actualiza en tiempo real y que permite al usuario controlar la simulación paso a paso.

8. Cronograma

(FALTA)

9. Presupuesto y Fuentes de Financiación

Este proyecto no requiere financiación externa. Los recursos necesarios (equipo de cómputo, licencias de software libre y acceso a Visual Studio Code) están disponibles sin costo adicional.

Referencias

[1] J Djordjevic, B Nikolic, B Tanja, and A Milenković. Cal2: Computer aided learning in computer architecture laboratory. *Computer Applications in Engineering Education*, 16(3):199–207, 2008.

[2] R Giorgi and G Mariotti. WebriSC-V: A web-based education-oriented RISC-V pipeline simulation environment. In *Proceedings of the Workshop on Computer Architecture Education, WCAE 2019*, 2019.

[3] R M Ihtemam, R Khan, and M A Qayum. Investigative analysis of three RISC-V micro-architectures in the context of computer architecture research. In *2024 27th International Conference on Computer and Information Technology, ICCIT 2024 - Proceedings*, 2024.

[4] S S Khairullah. Realization of a 16-bit MIPS RISC pipeline processor. In *HORA 2022 - 4th International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Proceedings*, 2022.

[5] J Mallidu and S V Siddamal. A survey on in-order 5-stage pipeline RISC-V processor implementation. *Lecture Notes in Networks and Systems*, 2023.

[6] H Oztekin, F Temurtas, and A Gulbag. A software-based interactive system on BZK.SAU.FPGA10.1 micro computer design for teaching computer architecture and organization. In *ELECO 2011 - 7th International Conference on Electrical and Electronics Engineering*, 2011.

[7] M S Poojary and H K Ravi. Design of RV32I instruction set architecture. In *International Conference on Smart Systems for Applications in Electrical Sciences, ICSSES 2025*, 2025.

[8] N Senske. Confronting the challenges of computational design instruction. In *Rethinking Comprehensive Design: Speculative Counterculture - Proceedings of the 19th International Conference on Computer-Aided Architectural Design Research in Asia, CAADRIA 2014*, 2014.

[9] K Vollmar and P Sanderson. MARS: An education-oriented MIPS assembly language simulator. In *Proceedings of the Thirty-Seventh SIGCSE Technical Symposium on Computer Science Education*, 2007.

[10] Y Yang, X Li, Z Liu, and W Ke. RM2PT: A tool for automated prototype generation from requirements model. In *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion (ICSE-Companion 2019)*, 2019.