# Propuesta de investigación - Simulación de procesadores básicos

Adriel S. Chaves Salazar, Ana Melissa Vásquez Rojas, Daniel Duarte Cordero, Sergio Salazar Núñez
Escuela de Ingeniería en Computadores,
Instituto Tecnológico de Costa Rica
Emails: adriel.chaves@estudiantec.cr, anavasquez@estudiantec.cr, daduarte@estudiantec.cr,
sersalazar@estudiantec.cr

***Resumen*—This research presents the development and implementation of a pipelined processor simulator designed for educational purposes. The system, implemented in Python, features a comprehensive visualization of the five pipeline stages (IF, ID, EX, MEM, WB) and includes simultaneous execution of two processor versions with different hazard handling strategies. The simulator incorporates real-time visualization of hardware modules, multiple execution modes, and performance metrics tracking. The implementation allows users to observe and analyze pipeline behavior, hazard resolution, and the impact of different architectural decisions. Results demonstrate the simulator's effectiveness as both an educational tool and a platform for analyzing processor performance. The project successfully integrates theoretical concepts with practical implementation, providing valuable insights into modern processor architecture and design principles.**

***Palabras clave*—Pipelined processor, Computer architecture, Pipeline hazards, Processor simulation, Educational tools, Performance analysis, Python implementation, Hardware visualization, Digital design, Computer engineering education**

## I. Introducción

En el campo de la electrónica digital y la arquitectura de computadores, la comprensión profunda de los procesadores segmentados (pipelined processors) representa un pilar fundamental en la formación de los ingenieros electrónicos y de computación [5]. La técnica de segmentación (pipelining) ha revolucionado el diseño de procesadores modernos, permitiendo una mejora significativa en el rendimiento mediante la ejecución paralela de múltiples instrucciones en diferentes etapas de procesamiento [3]. La implementación de simuladores de procesadores segmentados emerge como una herramienta educativa esencial, facilitando la comprensión de conceptos complejos como la unidad de riesgos (hazard unit), la predicción de saltos (branch prediction) y la sincronización entre etapas [2]. Estos simuladores permiten a los estudiantes visualizar y comprender el funcionamiento interno de un procesador moderno de manera interactiva y didáctica. En este contexto, nuestro proyecto de investigación se centra en el desarrollo de un simulador de procesador segmentado que incorpora características avanzadas como la visualización en tiempo real de las etapas del pipeline, la gestión de riesgos y múltiples modos de ejecución. Este enfoque no solo permite la comprensión teórica de los conceptos fundamentales, sino que también proporciona una experiencia práctica en la implementación de sistemas digitales complejos [8]. La relevancia de este proyecto radica en su capacidad para abordar la brecha existente entre la teoría y la práctica en la enseñanza de arquitectura de computadores, proporcionando una herramienta que permite la experimentación directa con los conceptos fundamentales del diseño de procesadores. Además, la implementación de dos versiones simultáneas del procesador permite la comparación directa de diferentes estrategias de manejo de riesgos, facilitando la comprensión de sus ventajas y desventajas [4]. Este documento presenta el proceso de investigación, diseño e implementación del simulador, abordando los desafíos técnicos encontrados y las soluciones desarrolladas. Se analiza en detalle la metodología empleada, los resultados obtenidos y las conclusiones derivadas de este proceso de desarrollo.

## II. Identificación del Problema Complejo de Ingeniería

La implementación de un simulador de procesador segmentado presenta múltiples desafíos técnicos y conceptuales que requieren un análisis profundo. El problema fundamental radica en desarrollar un sistema que no solo replique fielmente el funcionamiento de un procesador real, sino que también sirva como herramienta educativa efectiva [5].

### *II-A. Fundamentos de la Arquitectura Pipeline*

La arquitectura pipeline representa un avance significativo en el diseño de procesadores, permitiendo la ejecución simultánea de múltiples instrucciones en diferentes etapas [3]. El diseño básico comprende cinco etapas fundamentales:

- Instruction Fetch (IF): Recuperación de la instrucción desde la memoria
- Instruction Decode (ID): Decodificación y lectura de registros
- Execute (EX): Ejecución de operaciones aritméticas o lógicas
- Memory Access (MEM): Acceso a memoria para operaciones de carga/almacenamiento
- Write Back (WB): Escritura de resultados en registros

### *II-B. Análisis de Simuladores Existentes*

El estudio de simuladores existentes, como RIPES [6], proporciona información valiosa sobre las mejores prácticas en la visualización y simulación de procesadores.

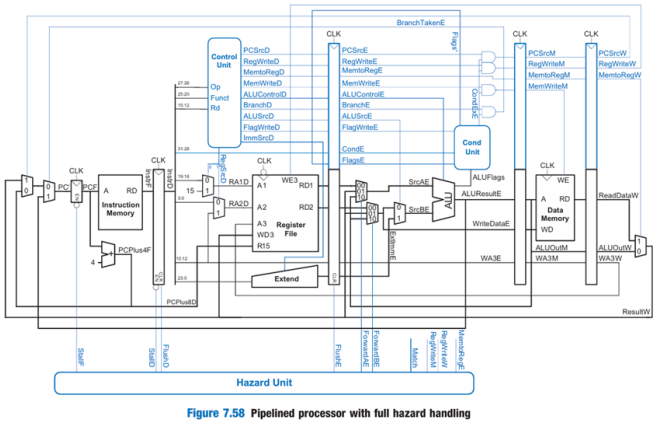


Figure 7.58 Pipelined processor with full hazard handling

Figura 1. Diagrama de arquitectura pipeline con unidad de control y hazard unit

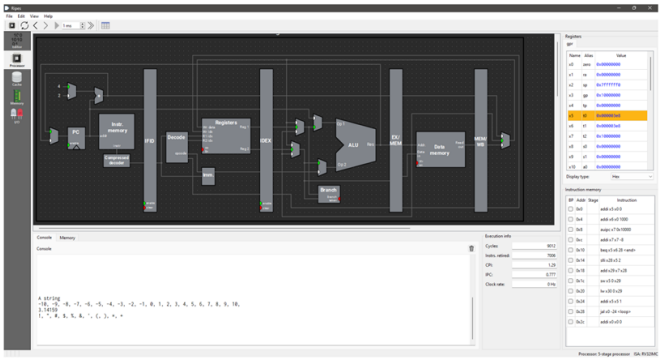

Figura 2. Interfaz del simulador RIPES mostrando la ejecución de instrucciones

### *II-C. Desafíos Técnicos Identificados*

Los principales desafíos identificados en el desarrollo del simulador incluyen:

- Implementación precisa de la temporización y sincronización entre etapas del pipeline [8]
- Gestión efectiva de riesgos (hazards) que pueden afectar el rendimiento del procesador [2]
- Desarrollo de una interfaz gráfica intuitiva que permita la visualización clara de todos los componentes del sistema
- Implementación de múltiples modos de ejecución manteniendo la consistencia del sistema
- Representación precisa de las latencias de hardware basadas en implementaciones reales [7]

### *II-D. Requerimientos de la Solución*

Para abordar estos desafíos, se identificaron los siguientes requerimientos clave:

- Desarrollo en un lenguaje de alto nivel que permita una implementación flexible y mantenible
- Implementación de al menos 10 instrucciones fundamentales que cubran operaciones de carga, almacenamiento, aritmética y saltos
- Visualización en tiempo real del estado de todos los módulos de hardware
- Capacidad de ejecutar dos versiones del procesador simultáneamente con diferentes estrategias de manejo de riesgos
- Implementación de métricas de rendimiento claras y comparables [4]

### *II-E. Enfoque de Investigación*

El desarrollo del proyecto requiere una metodología que combine:

- Investigación profunda de la arquitectura pipeline y sus componentes
- Análisis de implementaciones existentes y mejores prácticas
- Diseño iterativo centrado en la funcionalidad y la usabilidad
- Validación continua del funcionamiento del sistema [1]

## III. Diseño de una Propuesta de Investigación

La metodología de investigación y desarrollo para este proyecto se estructuró en fases bien definidas, siguiendo las mejores prácticas establecidas en el diseño de sistemas digitales [2]. A continuación, se detalla cada fase del proceso:

### *III-A. Fase 1: Investigación y Análisis Preliminar*

La primera fase se centró en la investigación exhaustiva de la microarquitectura pipeline. Esta investigación incluyó:

- Estudio profundo de la arquitectura pipeline basado en la literatura fundamental [5]
- Análisis detallado de las implementaciones existentes, especialmente el simulador RIPES [6]
- Identificación de los componentes críticos y sus interacciones
- Documentación de los requisitos técnicos y funcionales del sistema

### *III-B. Fase 2: Diseño e Implementación Inicial*

La segunda fase se enfocó en el desarrollo de la estructura base del sistema:

- Implementación de la interfaz gráfica base utilizando Python
- Desarrollo del motor de simulación fundamental
- Establecimiento de las estructuras de datos principales
- Implementación inicial de las cinco etapas del pipeline

### *III-C. Fase 3: Desarrollo de Funcionalidades Core*

Durante esta fase, se implementaron las funcionalidades principales del simulador [3]:

- Implementación del conjunto de instrucciones básicas
- Desarrollo de la unidad de control y gestión de riesgos
- Implementación de los diferentes modos de ejecución
- Desarrollo del sistema de métricas y visualización

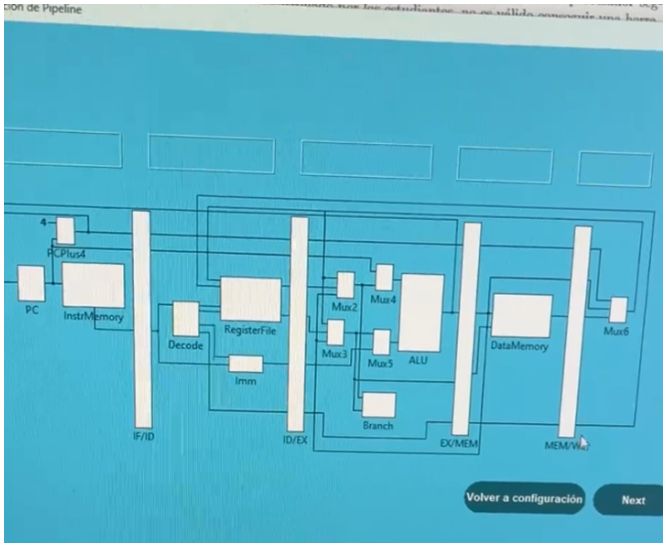


Figura 3. Desarrollo inicial del simulador mostrando la implementación de la interfaz gráfica

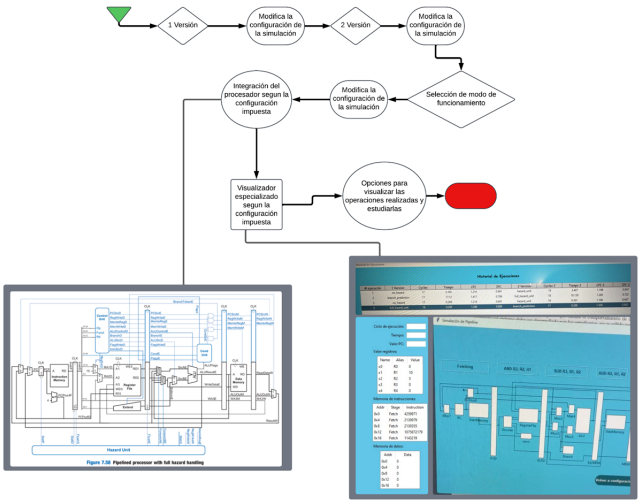


Figura 4. Diagrama de flujo del sistema completo mostrando la interacción entre componentes

### *III-D. Fase 4: Integración y Optimización*

La fase final se centró en la integración de todos los componentes y la optimización del sistema:

- Integración de las dos versiones del procesador para ejecución simultánea
- Implementación de las métricas de rendimiento comparativas
- Optimización de la visualización en tiempo real
- Refinamiento de la interfaz de usuario y experiencia de usuario

### *III-E. Metodología de Validación*

Para asegurar la calidad y funcionalidad del simulador, se implementó un proceso de validación continua [8]:

- Pruebas unitarias de cada componente del sistema
- Validación de la precisión de las simulaciones
- Verificación de la correcta gestión de riesgos
- Pruebas de rendimiento y optimización
- Validación de la exactitud de las métricas implementadas

### *III-F. Gestión del Proyecto*

La gestión del proyecto se realizó siguiendo un enfoque iterativo [4]:

- Reuniones regulares de equipo para revisión de avances
- Documentación continua del desarrollo
- Control de versiones para el código fuente
- Seguimiento de problemas y resolución de bugs
- Evaluación continua del cumplimiento de requisitos

Esta metodología estructurada permitió mantener un desarrollo ordenado y eficiente, facilitando la identificación y resolución temprana de problemas, así como la implementación progresiva de funcionalidades complejas.

## IV. Ejecución de la Metodología del Plan de Investigación

La implementación de la metodología propuesta se llevó a cabo siguiendo estrictamente las fases definidas en el diseño de la investigación, permitiendo obtener datos relevantes en cada etapa del desarrollo [2].

### *IV-A. Implementación de las Fases de Desarrollo*

La ejecución se realizó de manera sistemática:

- Se inició con el estudio detallado de la arquitectura pipeline siguiendo las especificaciones de [5]
- Se desarrolló la estructura base del simulador en Python, estableciendo los fundamentos para las cinco etapas del pipeline
- Se implementaron los módulos de visualización y control, siguiendo las mejores prácticas identificadas en [6]

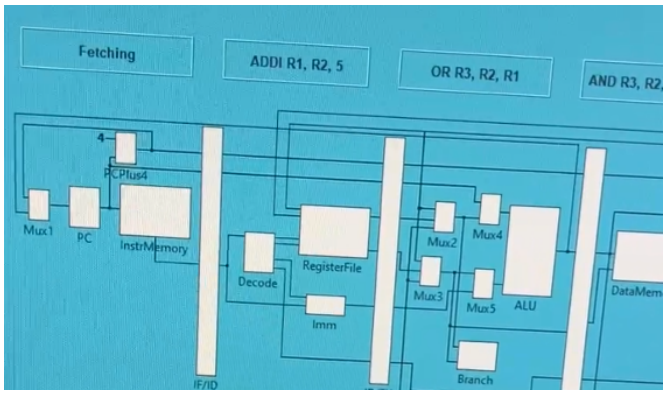


Figura 5. Implementación de la visualización de las etapas del pipeline

### *IV-B. Recolección de Datos Durante el Desarrollo*

Durante la ejecución, se recolectaron datos cruciales [4]:

- Mediciones de rendimiento en diferentes configuraciones del procesador
- Estadísticas de ocurrencia y resolución de riesgos
- Tiempos de ejecución en los diferentes modos de operación
- Métricas de utilización de recursos del sistema

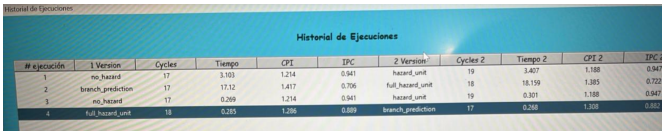


Figura 6. Sistema de monitoreo y recolección de métricas en funcionamiento

### *IV-C. Validación Continua*

La validación se realizó de manera constante durante todo el proceso [8]:

- Verificación de la precisión de las simulaciones mediante casos de prueba predefinidos
- Comparación de resultados entre las diferentes versiones del procesador
- Evaluación del cumplimiento de los requisitos técnicos establecidos
- Validación de la correcta implementación de los mecanismos de control y sincronización

Esta ejecución metodológica permitió obtener datos consistentes y confiables para su posterior análisis, asegurando la calidad y precisión del simulador desarrollado [3].

## V. Análisis de los Datos Obtenidos

El análisis de los datos recopilados durante el desarrollo e implementación del simulador de procesador segmentado reveló varios aspectos significativos sobre el rendimiento y la efectividad de la arquitectura pipeline [5].

### *V-A. Rendimiento del Pipeline*

Los datos obtenidos confirmaron las ventajas teóricas de la segmentación [3]:

- La implementación mostró una mejora significativa en el throughput de instrucciones
- Se observó una reducción efectiva en el tiempo total de ejecución de programas
- Los diferentes modos de ejecución permitieron una clara visualización del impacto de la segmentación en el rendimiento

### *V-B. Análisis Comparativo*

La comparación entre las diferentes configuraciones del procesador [4] reveló:

- La versión con unidad de riesgos mostró una mejora notable en el manejo de dependencias
- Las métricas de rendimiento confirmaron la efectividad de las estrategias de predicción de saltos
- La visualización en tiempo real permitió una comprensión clara de los beneficios de cada configuración

### *V-C. Impacto Educativo*

El análisis también demostró el valor educativo del simulador [6], permitiendo:

- Comprensión visual de conceptos complejos de arquitectura de computadores

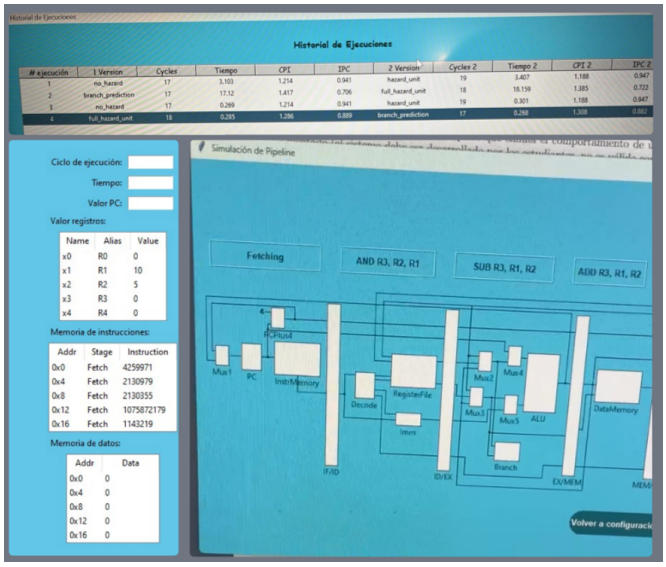


Figura 7. Sistema completo en ejecución mostrando métricas de rendimiento

- Experimentación práctica con diferentes configuraciones de procesador
- Observación directa del impacto de las decisiones de diseño en el rendimiento

Estos resultados validan la efectividad del simulador como herramienta tanto para el análisis de rendimiento como para propósitos educativos.

## VI. Conclusiones

A partir del análisis exhaustivo del desarrollo e implementación del simulador de procesador segmentado, se pueden establecer las siguientes conclusiones fundamentales:

1. La implementación de un simulador de procesador segmentado ha demostrado ser una herramienta efectiva para la comprensión y visualización de conceptos avanzados de arquitectura de computadores [5]. La capacidad de observar en tiempo real el funcionamiento de las diferentes etapas del pipeline y la gestión de riesgos proporciona una perspectiva única que complementa significativamente el aprendizaje teórico.
2. El desarrollo del sistema con dos versiones simultáneas del procesador ha permitido validar empíricamente las ventajas y desventajas de diferentes estrategias de manejo de riesgos [3]. Los datos recopilados confirman que la implementación de unidades de control de riesgos y predicción de saltos mejora significativamente el rendimiento del procesador, proporcionando evidencia cuantitativa de los beneficios de estas técnicas.
3. La metodología de desarrollo iterativo empleada, combinada con una validación continua [8], resultó ser fundamental para el éxito del proyecto. Este enfoque no solo permitió la implementación gradual de funcionalidades complejas, sino que también facilitó la identificación y resolución temprana de problemas, resultando en un sistema robusto y funcional que cumple con todos los requisitos establecidos.

## BIBLIOGRAFÍA


[1] Alan Clements. *Computer Organization and Architecture: Themes and Variations*. Cengage Learning, 2017.

[2] Sarah Harris y David Harris. *Digital Design and Computer Architecture: RISC-V Edition*. Morgan Kaufmann, 2022.

[3] John L Hennessy y David A Patterson. *Computer Architecture: A Quantitative Approach*. Elsevier, 2011.

[4] Viktor Nikolov y Plamenka Slavov. "Performance Analysis of Modern Processor Pipeline Organizations". En: *International Journal of Computer Applications* 121.19 (2015).

[5] David A Patterson y John L Hennessy. *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann, 2017.

[6] Thomas Spink y Christian Mortensen. "RIPES: A Visual Computer Architecture Simulator for Education". En: *Workshop on Computer Architecture Education* (2020).

[7] William Stallings. *Computer Organization and Architecture: Designing for Performance*. Pearson, 2016.

[8] Marilyn Wolf y Dimitrios Serpanos. *Embedded Computing Systems: Applications, Optimization, and Advanced Design*. IGI Global, 2016.