**UNIVERSIDAD CATÓLICA BOLIVIANA “SAN PABLO”**

**SEDE SANTA CRUZ**

**INGENIERÍA EN INTELIGENCIA ARTIFICIAL**



**DESARROLLO DE EMULADOR INTERACTIVO BASADO EN LA ARQUITECTURA x86**

**NOMBRE:** STEVEN CUELLAR JUSTINIANO

**DOCENTE:**  LOAYZA CARRASCO PAULO CESAR

7 de octubre de 2025

Índice

[**1.** **INTRODUCCIÓN** 1](#_Toc210780785)

[**2.** **MARCO TEÓRICO** 1](#_Toc210780786)

[**2.1.** **Concepto e Historia de la Arquitectura de Computadoras** 1](#_Toc210780787)

[**2.2.** **Modelo de arquitectura de Von Neumman** 2](#_Toc210780788)

[**2.3.** **Modelo de arquitectura de Harvard** 3](#_Toc210780789)

[**2.4.** **Comparativa entre los modelos de Von Neumman y Harvard** 4](#_Toc210780790)

[**2.5.** **Componentes en común entre ambos modelos** 4](#_Toc210780791)

[**2.6.** **Organización de las arquitecturas x86 actuales** 5](#_Toc210780792)

[**3.** **METODOLOGÍA** 6](#_Toc210780793)

[**3.1.** **Herramientas utilizadas** 6](#_Toc210780794)

[**3.2.** **Acabado final del simulador** 6](#_Toc210780795)

[**4.** **CONCLUSIÓN** 6](#_Toc210780796)

[**5.** **BIBLIOGRAFÍA** 7](#_Toc210780797)

# **INTRODUCCIÓN**

La arquitectura de computadoras es un campo fundamental en la informática que define la estructura y el funcionamiento interno de los sistemas computacionales. En esencia, se refiere al diseño y organización de los componentes hardware y software que permiten a una computadora procesar datos, ejecutar instrucciones y realizar operaciones complejas. Este ensayo explora el concepto básico de la arquitectura de computadoras, su evolución histórica y un análisis detallado de dos modelos clave: Von Neumann y Harvard. Además, se comparan estos modelos, se describen componentes comunes y se examina la organización de las arquitecturas x86 actuales. Finalmente, se presenta un simulador interactivo diseñado para ilustrar el funcionamiento del modelo Von Neumann, incorporando elementos como memorias cache y virtual (Burks, Goldstine, & von Neumann, 1946).

El propósito de este documento es proporcionar una visión clara y estructurada de estos temas, destacando su relevancia en el desarrollo de la tecnología moderna. Analizar los fundamentos de la arquitectura de computadoras, con énfasis en los modelos Von Neumann y Harvard, para proporcionar una base teórica sólida y comprender su relevancia en los sistemas computacionales modernos. Y Diseñar un simulador interactivo del modelo Von Neumann que incluya memorias caches (L1, L2, L3) y memoria virtual, permitiendo visualizar el ciclo de instrucciones, el manejo de cuellos de botella y la optimización mediante caches para facilitar la comprensión práctica de estos conceptos.

# **MARCO TEÓRICO**

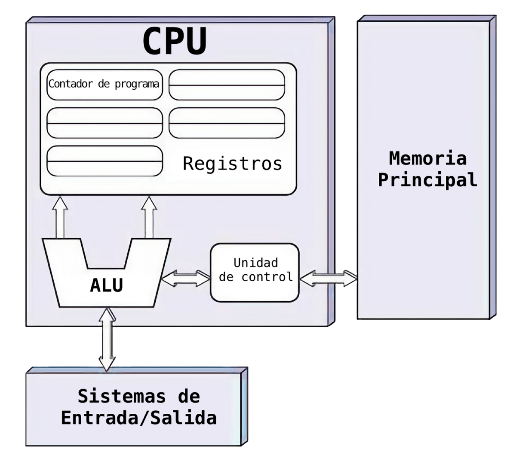
## **Concepto e Historia de la Arquitectura de Computadoras**

La arquitectura de computadoras se define como el conjunto de principios y métodos que describen la funcionalidad, organización e implementación de un sistema informático. Incluye aspectos como la forma en que se almacenan y procesan los datos, la interacción entre componentes y la eficiencia en el manejo de recursos (Patterson & Hennessy, 2017).

Históricamente, el origen de la arquitectura moderna se remonta a la década de 1940. Durante la Segunda Guerra Mundial, se desarrollaron las primeras computadoras electrónicas, como la ENIAC (1945), diseñada por John Mauchly y J. Presper Eckert, que era programable, pero requería reconexión manual. Un avance clave fue el informe de John von Neumann en 1945, titulado "First Draft of a Report on the EDVAC", que propuso un modelo donde programas y datos comparten la misma memoria. Esto marcó el nacimiento de la arquitectura Von Neumann, que influyó en la mayoría de las computadoras posteriores (Tanenbaum & Austin, 2013). En paralelo, el modelo Harvard, originado en la Mark I (1944) de Howard Aiken, separaba memorias para datos e instrucciones (Patterson & Hennessy, 2017). La evolución continuó con la miniaturización de componentes (transistores en los 1950s, circuitos integrados en los 1960s) y el surgimiento de arquitecturas como RISC y CISC en los 1980s, llevando a sistemas como x86 de Intel (Burks, Goldstine, & von Neumann, 1946).

## **Modelo de arquitectura de Von Neumman**

El modelo Von Neumann, también conocido como arquitectura de Princeton, se basa en una memoria unificada para almacenar tanto instrucciones como datos. La CPU accede a esta memoria a través de un bus único, procesando instrucciones secuencialmente: fetch (obtener instrucción), decode (decodificar), execute (ejecutar) y write-back (escribir resultados).



**Fig. 1:** Esquematización del modelo de Von Neumman

**Ventajas:**

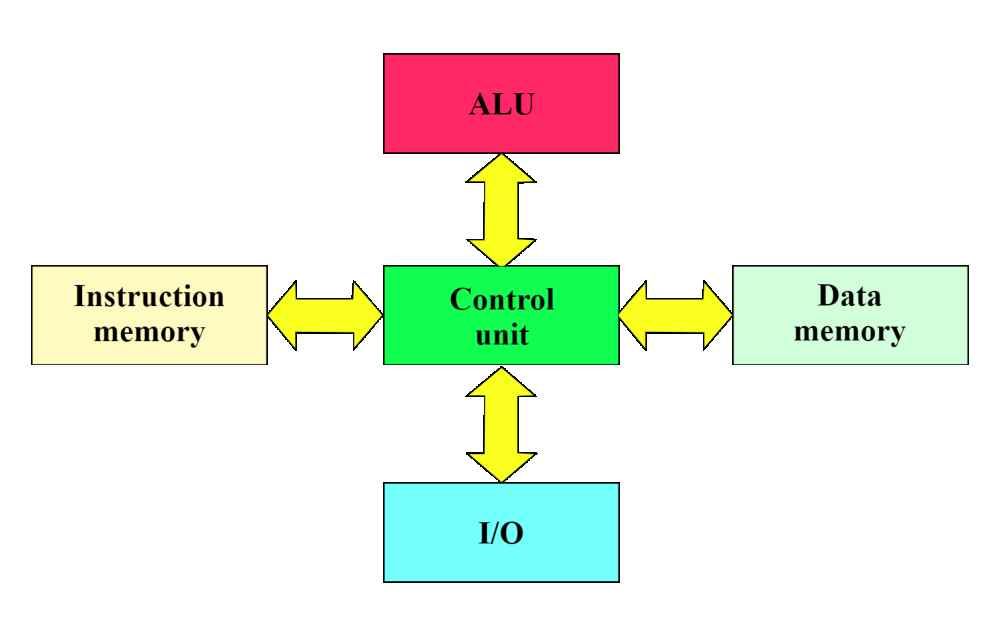
* **Simplicidad del diseño:** Al usar una sola memoria, reduce la complejidad hardware y costos.
* **Flexibilidad:** Permite que los programas modifiquen sus propias instrucciones, facilitando lenguajes de programación y sistemas operativos.
* **Eficiencia en espacio:** No requiere memorias separadas, optimizando el uso de recursos en sistemas con memoria limitada.

**Desventajas:**

* **Cuello de botella de Von Neumman:** El bus único limita el ancho de banda, ya que no se pueden acceder instrucciones y datos simultáneamente, reduciendo el rendimiento.
* **Vulnerabilidad a errores:** La mezcla de datos e instrucciones puede llevar a corrupciones si un programa sobrescribe código accidentalmente.
* **Menor paralelismo:** Dificulta la ejecución concurrente en comparación con modelos que separan memorias.

## **Modelo de arquitectura de Harvard**

El modelo Harvard utiliza memorias separadas para instrucciones (ROM o flash) y datos (RAM), con buses independientes para cada una. Esto permite accesos simultáneos, común en microcontroladores y DSP (procesadores de señales digitales) (Patterson & Hennessy, 2017).



**Fig. 2:** Esquematización del modelo Harvard

**Ventajas:**

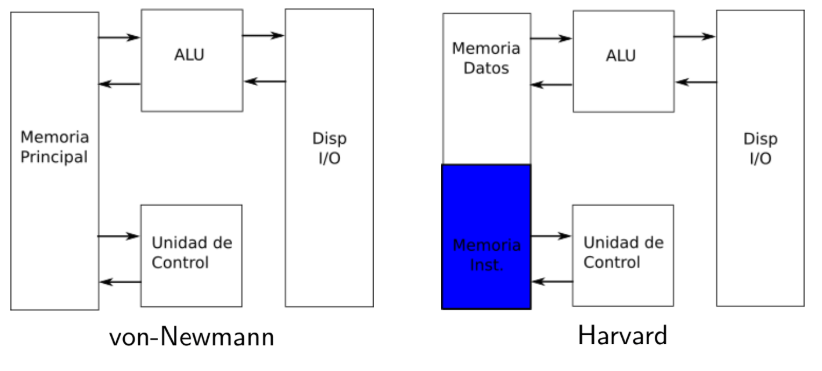
* **Mayor rendimiento:** Accesos paralelos eliminan el cuello de botella, ideal para aplicaciones en tiempo real como procesamiento de audio o video.
* **Seguridad mejorada:** La separación previene que datos modifiquen instrucciones, reduciendo riesgos de inyecciones de código.
* **Optimización para tareas específicas:** Facilita diseños embebidos donde las instrucciones son fijas y los datos variables.

**Desventajas:**

* **Mayor complejidad y costo:** Requiere hardware adicional para memorias y buses separados.
* **Menos flexibilidad:** Dificulta la modificación de programas en runtime, limitando su uso en sistemas generales.
* **Ineficiencia en memoria:** Puede desperdiciar espacio si una memoria se subutiliza mientras la otra se satura.

## **Comparativa entre los modelos de Von Neumman y Harvard**

Ambos modelos comparten el principio de procesamiento secuencial, pero difieren en la gestión de memoria. Von Neumann es más versátil y económico, dominante en computadoras de propósito general (PCs, servidores), mientras que Harvard destaca en eficiencia y velocidad para sistemas embebidos (microcontroladores como Arduino). En términos de rendimiento, Harvard evita el cuello de botella, pero a costa de complejidad; Von Neumann es más propenso a ineficiencias, pero facilita la programación. En la práctica, muchas arquitecturas modernas son híbridas, como la Modified Harvard, que combina memorias separadas con caches unificados para equilibrar ventajas (Patterson & Hennessy, 2017).



**Fig. 3:** Comparación entre los modelos de Von Neumman y Harvard

## **Componentes en común entre ambos modelos**

Las arquitecturas x86, desarrolladas por Intel y AMD, son evoluciones del modelo Von Neumann con extensiones modernas. Organizadas en un diseño CISC (Complex Instruction Set Computing), incluyen pipelining, superscalar y out-of-order execution para mejorar el rendimiento. Incorporan caches multinivel y soporte para memoria virtual, haciendo híbridos con elementos de Harvard en caches separados para datos e instrucciones.

* **CPU (Unidad Central de Procesamiento):** Incluye la ALU (Unidad Aritmético-Lógica) para operaciones matemáticas y lógicas, y la UC (Unidad de Control) para coordinar el flujo de instrucciones.
* **Registros:** Como el IR (Registro de Instrucciones) para almacenar la instrucción actual, MAR (Registro de Dirección de Memoria) para direcciones, MDR (Registro de Datos de Memoria) para datos temporales, y FLAGS para estados (ej. cero, carry).
* **Memoria Principal:** RAM para almacenamiento volátil.
* **Sistemas de Memoria Avanzados:** Caches (L1, L2, L3) para acceso rápido, y Memoria Virtual para extender la RAM usando disco.

Estos elementos forman el núcleo de cualquier arquitectura, adaptándose a las diferencias en buses y memorias.

## **Organización de las arquitecturas x86 actuales**

Las arquitecturas x86, desarrolladas por Intel y AMD, son evoluciones del modelo Von Neumann con extensiones modernas. Organizadas en un diseño CISC (Complex Instruction Set Computing), incluyen pipelining, superscalar y out-of-order execution para mejorar el rendimiento. Incorporan caches multinivel y soporte para memoria virtual, haciendo híbridos con elementos de Harvard en caches separados para datos e instrucciones (Patterson & Hennessy, 2017).

**Componentes Principales:**

* **RAM (Random Access Memory):** Memoria principal volátil para datos y programas temporales.
* **MMU (Memory Management Unit):** Gestiona la traducción de direcciones virtuales a físicas, protegiendo la memoria.
* **ALU (Arithmetic Logic Unit):** Realiza operaciones aritméticas y lógicas.
* **UC (Unidad de Control):** Dirige el flujo de operaciones en la CPU.
* **MDR (Memory Data Register):** Almacena datos transferidos entre CPU y memoria.
* **MAR (Memory Address Register):** Contiene la dirección de memoria a acceder.
* **FLAGS:** Registros de estado que indican resultados de operaciones (ej. overflow, zero).
* **IR (Instruction Register):** Almacena la instrucción actual decodificada.
* **Caches L1, L2 y L3:** Memorias rápidas jerárquicas; L1 es la más rápida y pequeña (por núcleo), L2 intermedia, L3 compartida entre núcleos.
* **Memoria Virtual:** Extiende la RAM usando espacio en disco, gestionada por el MMU para paging y swapping.

# **METODOLOGÍA**

## **Herramientas utilizadas**

Para el desarrollo del simulador y la documentación asociada, se emplearon las siguientes herramientas:

* **Excel:** Utilizado como base para la interfaz gráfica y el modelado de componentes, permitiendo simular flujos de datos y memorias de manera visual.
* **VBA (Visual Basic for Applications):** Empleado para programar la lógica interactiva del simulador, incluyendo ciclos de fetch-execute y gestión de caches.
* **Git:** Sistema de control de versiones para guardar la documentación, rastrear cambios en el código VBA y asegurar el correcto uso y mantenimiento del simulador.

## **Acabado final del simulador**

# **CONCLUSIÓN**

En resumen, la arquitectura de computadoras ha evolucionado desde modelos pioneros como Von Neumann y Harvard hasta arquitecturas avanzadas como x86, que integran componentes eficientes para manejar demandas modernas. El modelo Von Neumann destaca por su versatilidad, mientras que Harvard por su rendimiento en aplicaciones específicas. La comparativa revela que las híbridas son ideales para el equilibrio. El simulador diseñado refuerza estos conceptos, demostrando la importancia de caches y memoria virtual. En el futuro, con avances en IA y computación cuántica, estas arquitecturas continuarán adaptándose, impulsando innovaciones tecnológicas.

# **BIBLIOGRAFÍA**

Patterson, D. A., & Hennessy, J. L. (2017). Computer Organization and Design: The Hardware/Software Interface (5th ed.). Morgan Kaufmann.

Tanenbaum, A. S., & Austin, T. (2013). Structured Computer Organization (6th ed.). Pearson Education.

Burks, A. W., Goldstine, H. H., & von Neumann, J. (1946). Preliminary discussion of the logical design of an electronic computing instrument. Institute for Advanced Study, Princeton University.