UNIVERSIDAD
CATÓLICA
BOLIVIANA
SANTA CRUZ

INGENIERÍA EN INTELIGENCIA ARTIFICIAL

DOCUMENTO DE ARQUITECTURA DE COMPUTADORAS

Simulador de Arquitectura x86

Docente: LOAYZA CARRASCO PAULO CESAR

Estudiantes: Jose Agustin Mendoza Ureña

Leandro Emiliano Miranda Román

José Alfredo Zambrana

Santa Cruz - Bolivia
Octubre, 2025

Índice

1. Introducción

La arquitectura de computadoras es uno de los pilares fundamentales en el campo de la informática. Comprender el funcionamiento interno de los componentes de un procesador, la interacción con la memoria y el flujo de instrucciones resulta esencial para estudiantes y profesionales.

El **proyecto "Simulador Interactivo de Arquitectura x86"** nace como una herramienta educativa diseñada para permitir a los estudiantes **visualizar en tiempo real el flujo de ejecución de un programa en lenguaje ensamblador**, comprender los ciclos de instrucción y observar el comportamiento de los registros, la memoria física, la memoria virtual y el pipeline de un procesador x86 simplificado.

Este documento presenta de forma detallada la **planificación, desarrollo, funcionamiento e implementación** del simulador, con un enfoque didáctico e interactivo que facilita el aprendizaje de conceptos que usualmente resultan abstractos.

2. Justificación del Proyecto

La enseñanza de arquitectura de computadores suele basarse en modelos teóricos, diagramas y pseudocódigo. Sin embargo, los estudiantes a menudo enfrentan dificultades para **visualizar cómo se ejecuta realmente un programa a nivel de hardware**.

El simulador desarrollado ofrece:

- **Interactividad:** Permite observar el comportamiento de cada componente mientras se ejecuta el programa.
- **Visualización:** Muestra de forma gráfica los registros, el pipeline y los ciclos de ejecución.
- **Educación aplicada:** Complementa los conocimientos adquiridos en clase mediante prácticas experimentales.
- **Accesibilidad:** Se implementó en una plataforma conocida (Excel VBA o Google Sheets con JavaScript), eliminando barreras de uso.

En resumen, este proyecto responde a la necesidad de herramientas accesibles para comprender los principios de la arquitectura x86 de forma didáctica.

3. Objetivos

3.1 Objetivo General

Desarrollar un **simulador funcional e interactivo** de arquitectura x86 que ejecute programas en ensamblador, mostrando de manera clara y visual el funcionamiento interno de la CPU, el pipeline y los diferentes tipos de memoria.

3.2 Objetivos Específicos

1. Implementar una interfaz para **cargar programas en lenguaje ensamblador x86**.
2. Mostrar de forma visual el **estado de los registros y de la memoria física** durante la ejecución.
3. Simular el **pipeline del procesador**, incluyendo las etapas de Fetch, Decode, Execute, Memory y Write Back.
4. Implementar una representación de la **memoria virtual y la política LRU** para el manejo de páginas.
5. Generar **gráficos y bitácoras automáticas** que registren el flujo de ejecución y los cambios en el estado interno.
6. Gestionar el proyecto en **GitHub**, utilizando control de versiones y tableros de tareas.

4. Marco Teórico

4.1 Arquitectura x86

La arquitectura x86 es una de las más utilizadas en computadoras personales. Se basa en el modelo de **Von Neumann**, donde los datos y las instrucciones comparten la misma memoria.
Incluye registros de propósito general (AX, BX, CX, DX), registros de control (IP, SP), una ALU para operaciones aritméticas y lógicas, y memoria jerárquica (RAM, Caché, Disco).

4.2 CPU: Unidad de Control, ALU y Registros

- **Unidad de Control (CU):** Coordina el flujo de instrucciones, interpretando el código máquina.
- **ALU:** Realiza operaciones aritméticas (suma, resta) y lógicas (AND, OR, NOT).

- **Registros:** Son pequeñas celdas de memoria de acceso rápido.
  El simulador permite observar cómo los registros cambian valor a medida que se ejecutan las instrucciones.

4.3 Memoria

- **Memoria Física:** RAM principal utilizada para almacenar instrucciones y datos.
- **Caché:** Memoria intermedia que acelera el acceso a los datos más utilizados.
- **Memoria Virtual:** Simula la ampliación de memoria usando disco y técnicas de paginación.

4.4 Ciclo de Instrucción

Cada instrucción pasa por las siguientes etapas:

1. **Fetch:** Obtener la instrucción desde memoria.
2. **Decode:** Interpretar el código binario.
3. **Execute:** Ejecutar la operación en la ALU.
4. **Memory:** Acceder a datos en memoria si es necesario.
5. **Write Back:** Guardar los resultados.

4.5 Pipeline

El pipeline permite ejecutar múltiples instrucciones de manera solapada, aumentando el rendimiento. Sin embargo, introduce **hazards**: conflictos de datos, control o recursos, que el simulador identifica gráficamente.

4.6 Políticas de Memoria

Se implementó la política **LRU (Least Recently Used)** para reemplazo de páginas en memoria virtual.

5. Metodología

5.1 Plataforma de Desarrollo

El simulador se desarrolló en [indicar si fue **Excel con VBA** o **Google Sheets con Apps Script**], aprovechando sus capacidades de celdas, gráficos y scripts para automatizar procesos.

5.2 Organización en GitHub

Se creó un **repositorio privado** para alojar el código fuente.

- Se usó **GitHub Projects (Kanban)** para planificar y distribuir tareas.

5.3 Distribución de Tareas

Cada integrante fue responsable de una sección:

- [Equipo de desarrollo]: Registros y CPU.
- [Equipo de desarrollo]: Memoria y políticas LRU.
- [Equipo de desarrollo]: Gráficos y bitácora.

6. Descripción Detallada de las Hojas del Simulador

6.1 Hoja “Programa”

- Permite ingresar el **código ensamblador**.
- Incluye botones para **cargar, ejecutar y reiniciar** el programa.
- Muestra el **contador de programa (PC)** y el estado actual de la instrucción.

6.2 Hoja “Gráfica”

- Presenta **gráficos dinámicos** de uso de memoria y progreso de instrucciones.
- Utiliza colores para resaltar etapas activas y fallos de caché.

6.3 Hoja “Procesos”

- Lista las instrucciones cargadas y muestra su **estado actual en el ciclo de instrucción**.
- Incluye una tabla con tiempos de ejecución por ciclo.

6.4 Hoja “Bitácora”

- Registra eventos importantes: inicio de ejecución, interrupciones, fallos de caché, resultados finales.

6.5 Hoja “Memoria Física”

- Muestra la **distribución de direcciones de RAM**.
- Se actualiza cada vez que se escribe o lee un dato.

6.6 Hoja “Memoria”

- Representa la **memoria virtual** y el intercambio de páginas según LRU.

6.7 Hoja “Gráfico MV”

- Visualiza de forma gráfica qué páginas están activas en la RAM.

6.8 Hoja “Pipeline”

- Representa el paso de cada instrucción por las etapas IF, ID, EX, MEM, WB.
- Identifica **hazards** y los marca visualmente.

6.9 Hoja “Consola”

- Muestra mensajes en tiempo real: errores, estados y salidas del programa.

6.10 Hoja “Registros”

- Lista todos los registros de la CPU con su **valor actualizado después de cada ciclo**.

7. Funcionamiento General

1. El usuario ingresa un **programa ensamblador** en la hoja Programa.
2. El simulador ejecuta **ciclo por ciclo**.
3. Todas las hojas (Memoria, Registros, Pipeline, Gráfica) se **actualizan en tiempo real**.
4. Se puede ejecutar **paso a paso** o automáticamente hasta finalizar.

8. Gestión del Proyecto con GitHub

- El repositorio contiene documentación.
- Se realizó seguimiento de tareas mediante un **Kanban** con columnas: *Por hacer*, *En progreso*, *Finalizado*.

9. Resultados y Pruebas

- Se cargaron programas de prueba como: sumas, restas y movimientos entre registros.
- El pipeline mostró la ejecución simultánea de instrucciones sin conflictos graves.
- La memoria virtual reaccionó adecuadamente a fallos de página bajo la política LRU.

10. Conclusiones

El **Simulador Interactivo de Arquitectura x86** logra explicar de forma visual el ciclo de ejecución de un procesador real.
Se convirtió en una herramienta útil para la enseñanza práctica, reforzando conceptos teóricos sobre **memoria, pipeline y registros**.

11. Bibliografía

- Stallings, W. (2021). *Computer Organization and Architecture: Designing for Performance* (11th ed.). Pearson.
- Tanenbaum, A. S., & Austin, T. (2013). *Structured Computer Organization* (6th ed.). Pearson.
- Patterson, D. A., & Hennessy, J. L. (2017). *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufman