Universidad de Guadalajara


Centro Universitario Para las Ciencias Exactas e Ingenierías

Arquitectura de Computadoras | Sección: D03

# Proyecto Final: Procesador MIPS Pipeline de 5 Etapas y Decodificador

Alumno:
Camacho Guerrero Ramón Anthony

Docente:

Jorge Ernesto Lopez Arce Delgado

Guadalajara, Jalisco. 28 de noviembre de 2025

El presente proyecto consiste en la implementación de un procesador basado en la arquitectura MIPS (Microprocessor without Interlocked Pipeline Stages) utilizando la técnica de segmentación (Pipeline) de 5 etapas. Esta técnica permite incrementar el rendimiento del procesador al ejecutar múltiples instrucciones simultáneamente en diferentes fases.

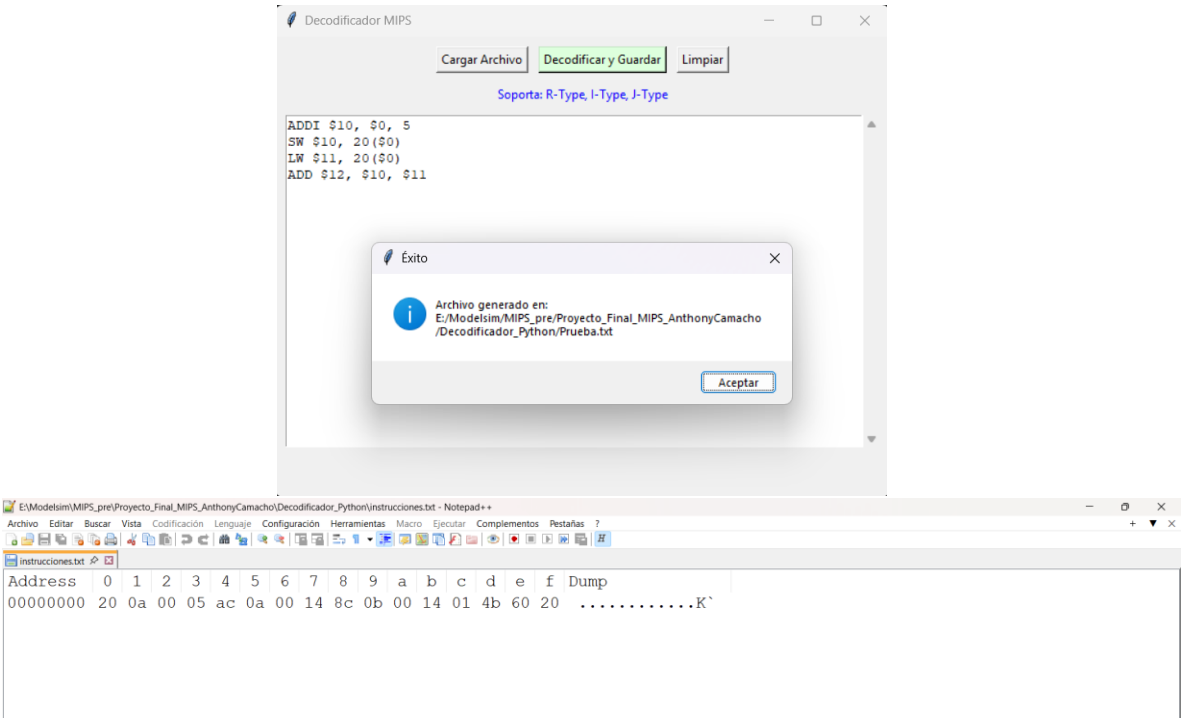
Adicionalmente, se desarrolló una herramienta de software en Python para decodificar instrucciones escritas en lenguaje ensamblador y convertirlas a código máquina (binario/hexadecimal), permitiendo la carga directa de programas en la memoria del procesador.

Como objetivo general se propone diseñar, implementar y simular un procesador MIPS de 5 etapas funcional en Verilog, junto con una herramienta de software complementaria para la generación automática de código máquina.

A su vez, los objetivos particulares que se han decidido durante la creación del proyecto son:

- Implementar las 5 etapas clásicas: Fetch, Decode, Execute, Memory, Write Back.
- Desarrollar una Unidad de Control capaz de manejar instrucciones Tipo R, Tipo I (aritméticas, lógicas, memoria y saltos) y Tipo J.
- Resolver conflictos de datos (Data Hazards) mediante la inserción de NOPs en el código ensamblador.
- Crear una interfaz gráfica (GUI) en Python que permita escribir código ensamblador, validarlo y exportarlo a formato binario compatible con la memoria del procesador.

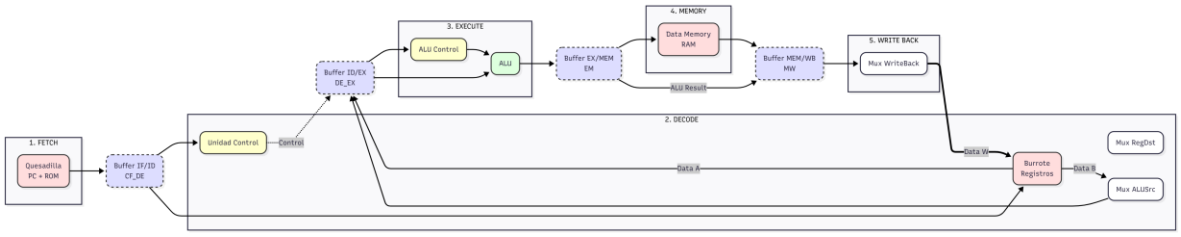
Se desarrolló un script en Python (Nucleo.py) utilizando la librería Tkinter para la interfaz gráfica. El núcleo del programa utiliza un diccionario de instrucciones que mapea los mnemónicos (como ADD, LW, BEQ) a sus respectivos *Opcodes* y *Functs*. El programa analiza cada línea de texto, identifica el tipo de instrucción (R, I, o J), convierte los registros y valores inmediatos a binario, y empaqueta el resultado en 32 bits (4 bytes) utilizando el formato *Big Endian*.




El diseño en Verilog se estructuró en módulos interconectados:

1. Fetch (Quesadilla): Contiene el *Program Counter* y la Memoria de Instrucciones.
2. Decode (Burrote + UnidadControl): Decodifica la instrucción, lee los registros y genera las señales de control.
3. Execute (ALU + ALUControl): Realiza operaciones aritméticas y cálculo de direcciones.
4. Memory (DataMemory): Maneja la lectura y escritura de datos (RAM).
5. Write Back: Escribe el resultado final en el banco de registros.

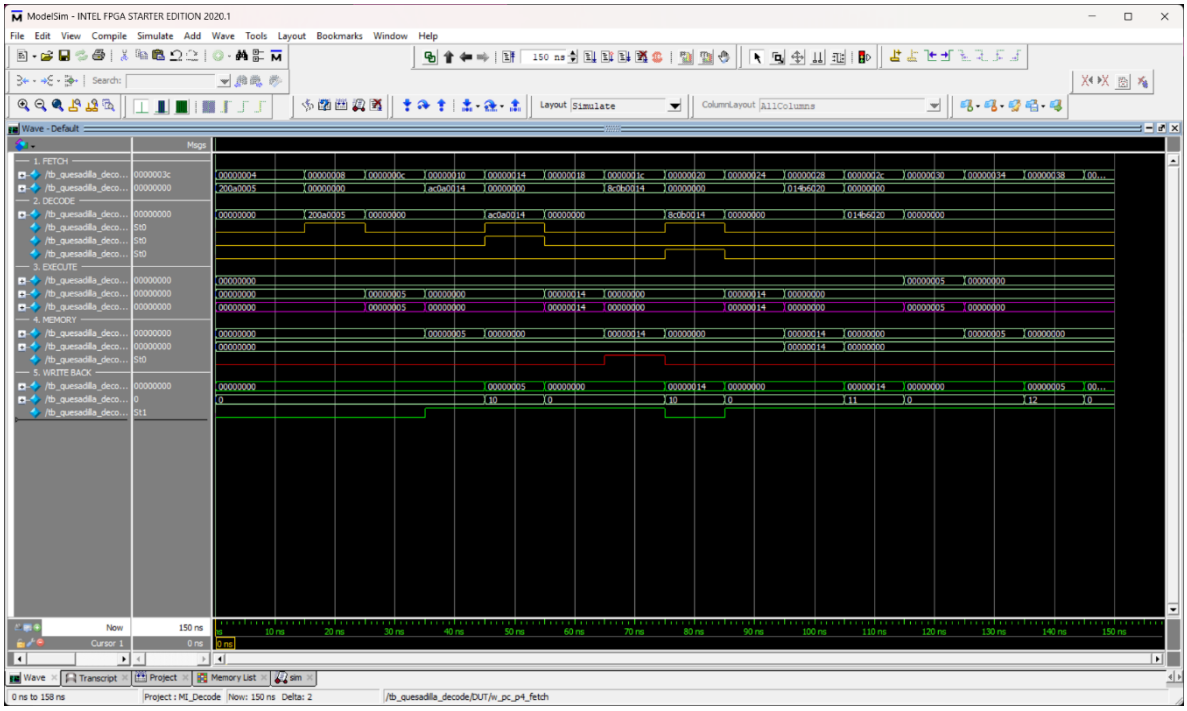
Las etapas están separadas por registros de segmentación (*Buffers*) para mantener el flujo de datos sincronizado.



Para validar el funcionamiento, se ejecutó el siguiente programa de prueba que demuestra el uso de instrucciones aritméticas, acceso a memoria y manejo de riesgos de datos:

```
ADDI $10, $0, 5          ; Carga 5 en $10
SW $10, 20($0)           ; Guarda 5 en Memoria[20]
LW $11, 20($0)           ; Lee de Memoria[20] a $11
ADD $12, $10, $11        ; Suma 5 + 5 = 10 en $12
```

En la siguiente captura se observa el flujo de las instrucciones. En la etapa final (*Write Back*), se verifica que el registro de destino (w_dest_reg_wb) recibe el valor 12 y el dato escrito (w_wb_data) es 10, confirmando que la suma y el ciclo de memoria fueron exitosos.




Se logró implementar exitosamente un procesador MIPS segmentado capaz de ejecutar un conjunto variado de instrucciones. La integración entre el software (decodificador) y el hardware (Verilog) fue correcta, permitiendo automatizar la generación de código máquina. Durante el desarrollo, se observó la importancia de manejar los *Data Hazards*, ya que sin la inserción de instrucciones NOP, el procesador intentaba leer datos que aún no estaban disponibles, resultando en errores de cálculo. La solución implementada aseguró la integridad de los datos.

Referencias:


- Patterson, D. A., & Hennessy, J. L. *Computer Organization and Design: The Hardware/Software Interface*.
- Documentación del Proyecto Final, Curso de Arquitectura de Computadoras.