Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías

Ingeniería en Computación

Proyecto Final

Reporte de Trabajo Colaborativo presentado por

Humberto de Jesus Peña Dueñas
Karla Rebeca Hernández Elizarrarás
Elizabeth Arroyo Moreno

Alumnos de Cuarto Semestre ICOM

Arquitectura de Computadoras

Mtro. Jorge Ernesto Lopez Arce Delgado

Guadalajara, Jal. Mayo de 2025

[FASE 1]

Introducción.

La arquitectura MIPS (Microprocessor without Interlocked Pipeline Stages) es un referente en el ámbito académico por su diseño RISC (Reduced Instruction Set Computer), que se caracteriza por su simplicidad, eficiencia y estructura modular. Esta arquitectura logra un alto rendimiento mediante un conjunto reducido de instrucciones, ciclos de reloj predecibles y la técnica de segmentación (pipelining), lo que permite ejecutar múltiples instrucciones de manera paralela y optimizada. Su enfoque en la modularidad y la eficiencia la convierte en una herramienta pedagógica ideal para entender los principios fundamentales de los procesadores modernos.

Con este proyecto buscamos implementar un datapath MIPS de 32 bits capaz de ejecutar un conjunto específico de 28 instrucciones, clasificadas en tres tipos:

- Tipo R: Incluye operaciones como Add, Sub, Mul, Div, Or, And, Slt y Nop.
- Tipo I: Abarca instrucciones como Addi, Subi, Ori, Andi, Lw, Sw, Slti, Beq, Bne y Bgtz.
- Tipo J: Representada principalmente por la instrucción de salto J.

Este trabajo se basa en implementaciones previas de un datapath funcional para instrucciones R, I y J, siguiendo los principios descritos en el libro Computer Organization and Design: The Hardware/Software Interface de Patterson y Hennessy. Los módulos clave incluyen la Unidad de Control, el Banco de Registros, la ALU, las memorias de instrucciones y datos, así como buffers para gestionar el flujo de datos en un entorno de pipeline.

Tablas de Instrucciones

Instrucción	Tipo	Sintaxis
Add	R	Add $rd, $rs, $rt
Sub	R	Sub $rd, $rs, $rt
Mul	R	Mul $rd, $rs, $rt
Div	R	Div $rd, $rs, $rt
Or	R	Or $rd, $rs, $rt
And	R	And $rd, $rs, $rt
Addi	I	Addi $rt, $rs, immediate
Subi	I	Subi $rt, $rs, immediate

Instrucción	Tipo	Sintaxis
Ori	I	Ori $rt, $rs, immediate
Andi	I	Andi $rt, $rs, immediate

Instrucción	Tipo	Sintaxis
Lw	I	Lw $rt, $rs, immediate
Sw	I	Sw $rt, offset($rs)
slt	R	slt $rd, $rs, $rt
Slti	I	Slti $rt, $rs, $immediate
beq	I	beq $rs, $rt, offset
bne	I	bne $rs, $rt, offset
J	J	J target
nop	R	nop
bgtz	I	bgtz $rs, offset

[FASE 1]

Objetivo General.

Diseñar e implementar un datapath MIPS de 32 bits basado en arquitectura RISC, capaz de ejecutar un conjunto específico de 28 instrucciones (tipo R, I y J), simulando el flujo de datos y control en un entorno de pipeline de 5 etapas (IF, ID, EX, MEM, WB).

Objetivos específicos.

1. Analizar las instrucciones MIPS (R, I, J) para definir sus formatos binarios y su comportamiento en el datapath.
2. Diseñar un sistema modular que incluya:
   - Banco de registros.

- Memorias (instrucciones y datos).
- ALU, multiplexores y unidad de control.
- Buffers para gestionar el pipeline.

3. Codificar un programa en ensamblador MIPS y traducirlo a código máquina.
4. Precargar el programa en memoria y simular su ejecución.
5. Verificar el funcionamiento del datapath, asegurando que cada etapa del pipeline opere correctamente.
6. Documentar el proceso desde el diseño teórico hasta la implementación práctica.
7. Comprender cómo un procesador ejecuta instrucciones a nivel hardware.
8. Aplicar conceptos de pipeline, hazards y control de flujo de datos.
9. Integrar conocimientos de lógica digital, programación en Verilog y arquitectura de computadoras.

[FASE 1]

Desarrollo

Este proyecto se fundamenta en la arquitectura MIPS de 32 bits, siguiendo el modelo clásico de cinco etapas de pipeline:

- IF (Instruction Fetch): Obtención de la instrucción desde la memoria.
- ID (Instruction Decode): Decodificación y lectura de registros.
- EX (Execute): Ejecución de operaciones aritméticas o lógicas.
- MEM (Memory): Acceso a la memoria de datos para cargar o almacenar valores.
- WB (Write Back): Escritura de resultados en el banco de registros.

Para garantizar la sincronización entre etapas, se implementaron buffers intermedios (IF/ID, ID/EX, EX/MEM, MEM/WB), que evitan conflictos y aseguran un flujo de datos ordenado.

Módulos Principales.

1. PC (Program Counter):
   - Responsable de mantener y actualizar la dirección de la siguiente instrucción.
   - Incrementa su valor en 4 bytes por ciclo, a menos que se active un salto (Branch o Jump).

2. Memoria de Instrucciones:
   - Almacena el programa en código máquina, precargado mediante un script en Python que traduce instrucciones ensamblador a binario.

3. Banco de Registros:
   - Permite la lectura simultánea de dos registros y la escritura en uno.

- Utiliza registros específicos para almacenar resultados temporales, punteros y datos intermedios.

4. ALU y ALU Control:
   - Soporta operaciones aritméticas (Add, Sub), lógicas (And, Or) y comparaciones (Slt).
   - La unidad ALU Control interpreta las señales de la Unidad de Control y el campo funct para determinar la operación exacta.

5. Unidad de Control:
   - Genera señales críticas para cada tipo de instrucción (R, I, J), como habilitación de escritura en registros, selección de operaciones ALU y acceso a memoria.

6. Memoria de Datos:
   - Almacena valores requeridos por el programa y soporta operaciones de carga (Lw) y almacenamiento (Sw).

Pruebas y Verificación.

Con el objetivo de asegurar el correcto funcionamiento del datapath, se llevaron a cabo diversas simulaciones de manera progresiva, abarcando cada una de las etapas del flujo de datos. El proceso comenzó con la verificación de la lectura precisa de las instrucciones almacenadas en la memoria, asegurando que se recuperaran correctamente para su posterior ejecución. A continuación, se evaluó el comportamiento de los registros, comprobando que fueran actualizados de forma adecuada con los valores correspondientes durante cada ciclo de instrucción.

Posteriormente, se analizó el flujo de datos entre los diferentes módulos del sistema, prestando especial atención al tránsito de información a través de los buffers entre las distintas etapas del pipeline. Esto permitió verificar que los datos se transfirieran correctamente sin pérdidas ni errores de sincronización.

Asimismo, se monitoreó cuidadosamente el funcionamiento del pipeline al ejecutar instrucciones con dependencias entre sí, con el fin de detectar la presencia de posibles conflictos, tanto de datos como de control, y comprobar si el diseño era capaz de gestionarlos correctamente.
Para todo este proceso de verificación, se utilizó el entorno de simulación ModelSim, el cual permitió observar las señales internas. Se realizaron pruebas detalladas paso a paso, analizando cada transición y resultado, con el propósito de confirmar que el datapath cumpliera con el comportamiento esperado en todas las situaciones evaluadas.

[FASE 1]

Conclusión general.

La realización de este proyecto facilitó una comprensión mucho más detallada del funcionamiento interno de un procesador MIPS, particularmente en lo que respecta al procesamiento de instrucciones dentro de un datapath segmentado. A través de la implementación de los diferentes módulos en Verilog, así como la conversión manual de un programa escrito en lenguaje ensamblador a su forma binaria en código máquina, se logró fortalecer de manera significativa tanto los conocimientos teóricos como las habilidades prácticas en el área de arquitectura de computadoras.

Asimismo, la etapa de simulación y verificación del sistema completo permitió observar cómo interactúan los distintos componentes entre sí, haciendo evidente la importancia de la sincronización precisa entre etapas del pipeline. Este proceso ayudó a visualizar el recorrido de los datos a lo largo del datapath y resaltó el papel fundamental del control en la ejecución ordenada de las instrucciones. La experiencia contribuyó a consolidar conceptos clave como el manejo de riesgos de datos, el flujo de control y la importancia de una buena organización modular.

Conclusión Personal.

Desde mi perspectiva, este proyecto representó un reto considerable, pero al mismo tiempo se convirtió en una experiencia de gran valor académico y formativo. Me permitió integrar y aplicar de manera práctica distintos conocimientos adquiridos en cursos previos.

Aunque en algunos momentos surgieron dificultades como la integración correcta de los módulos o la identificación de errores en la ejecución del programa, cada obstáculo superado fortaleció mi capacidad de análisis y resolución de problemas. Ver el sistema funcionando correctamente después de múltiples pruebas y ajustes fue muy satisfactorio.

El tema me pareció particularmente interesante y útil, ya que me ofreció una visión más clara de cómo operan los procesadores desde una perspectiva de diseño. Considero que este tipo de proyectos son fundamentales para cualquier estudiante de ingeniería en computación, ya que acercan los conceptos abstractos al mundo real..

Bibliografías

Abd-El-Barr, M., & El-Rewini, H. (2004). *Fundamentals of computer organization and architecture* (Vol. 38). John Wiley & Sons.

Stalling, W. (2005). Organización y Arquitectura de Computadores (7a ed.) Pearson-Prentice-Hall.

Patterson, D. A. (2007). Computer organization and design: The hardware/software interface. Morgan Kaufmann.

Hwang K. (s.f.). Arquitectura de Computadores y Procesamiento Paralelo. Mc Graw-Hill.