Proyecto Final Fase 01 - Data Path

**Integrantes**

Alan Sabastian Guzmán Marquez

Alcaraz Suarez Gabriel Isaí

Kevin Manuel Sanchez Agredano

Quintero Gonzalez Diego Gerardo

**Introducción**

En el desarrollo de sistemas digitales, la implementación de procesadores es fundamental para el funcionamiento de dispositivos electrónicos. Comprender los elementos básicos de un procesador, como la Memoria, la Unidad Aritmético-Lógica (ALU), y el Banco de Registros, es esencial para diseñar y construir sistemas computacionales eficientes y funcionales.

Una vez entendido el funcionamiento de estos elementos básicos, es posible avanzar en la implementación de procesadores más complejos. En esta fase del proyecto, se aborda la construcción de un "DataPath", que forma parte de una versión simplificada del procesador MIPS. El objetivo principal de este DataPath es implementar y decodificar instrucciones básicas tipo R. Las instrucciones tipo R se caracterizan por tener una estructura de 32 bits, dividida en seis elementos: OpCode, RS, RT, RD, Shamt y Function. El OpCode y el Function son campos de 6 bits cada uno. Al ingresar a los módulos correspondientes, se van separando y procesando para ejecutar la instrucción adecuada.

En resumen, esta práctica tiene como objetivo implementar un DataPath capaz de ejecutar instrucciones tipo R en un procesador, integrando módulos existentes con nuevos módulos de control y selección de datos. El éxito en esta implementación permitirá comprender en detalle el funcionamiento interno de un procesador y sentar las bases para diseños más complejos en el futuro.

**Objetivo**

* Objetivo General

El objetivo principal es crear un DataPath capaz de ejecutar instrucciones de tipo R, para lograrlo será necesario integrar los módulos ya usados en actividades anteriores (ALU, BR y Memoria).

Con nuevos módulos que gestionarán el flujo de los bits en el sistema, estos módulos pueden observarse en la Figura 2.

* Objetivos Particulares
  + Desarrollar un decodificador en Python que reciba una expresión de tipo R en lenguaje ensamblador y la convierta en lenguaje binario, incluyendo el código de operación (Opcode), los registros de origen (rs), destino (rd) y destino (rt), y el desplazamiento (shamt) si es aplicable.
  + Desarrollar una presentación, en un archivo README.md, en el repositorio remoto en el cual se estarán subiendo los archivos y carpetas que se estarán trabajando a lo largo del proyecto.
  + Desarrollar un módulo, el cual tome una entrada de 32 bits, la cual será una expresión en ensamblador traducida a binario, este módulo tendrá varias salidas las cuales serán cada una de las partes que tiene una expresión de tipo R (mencionadas en el punto anterior).
  + Desarrollar el módulo de Unidad de Control (UC) con una entrada de 6 bits para recibir el Código de Operación (Opcode) y generar tres salidas de 1 bit (MemToReg, RegWrite y MemToWrite) y una salida de 3 bits (ALUOp) para controlar los diferentes elementos del sistema de acuerdo con el OpCode recibido.
  + Implementar el módulo de ALU-Control, el cual recibirá los bits 0:5 de la instrucción y una señal de 3 bits de la UC. Este módulo generará una salida de 3 bits para indicar a la ALU la operación que debe realizar, basándose en las señales enviadas por la UC.
  + Diseñar e implementar un Multiplexor 2:1 con dos entradas de 32 bits cada una y una entrada de 1 bit para seleccionar qué entrada se pasa a la salida. La salida del multiplexor será de 32 bits.
  + Desarrollar un testbench mínimo para los módulos nuevos (UC, ALU-Control y Multiplexor) para verificar su correcto funcionamiento individual.
  + Crear un testbench para el módulo DataPath Tipo-R (DPTR), instanciando todos los módulos desarrollados y probándolos con 10 instrucciones, dos de cada una de las instrucciones mencionadas en la introducción (ALU, SUB, OR, AND, SLT).
  + Preparar una tabla con las instrucciones en formato ensamblador y su equivalente en código máquina. Los datos de esta tabla se utilizarán como entrada para el testbench del DPTR.
  + Realizar pruebas del módulo DPTR para garantizar su correcto funcionamiento en la ejecución de instrucciones de tipo R, incluyendo diferentes combinaciones de instrucciones y condiciones de entrada.

**Desarrollo**

* Descripción de Python

Este código en Python está diseñado para decodificar instrucciones de ensamblador MIPS a su equivalente binario. Comienza con un diccionario llamado codigos\_TipoR que mapea operaciones de ensamblador como “add” y “sub” a sus respectivos códigos de operación (op) y función (funct) en binario.

La función decodificar\_instrucciones toma dos argumentos: un archivo de entrada con instrucciones en ensamblador y un archivo de salida donde se escribirán las instrucciones decodificadas. Dentro de esta función, se abre el archivo de entrada y se lee línea por línea. Cada línea se pasa a la función decodificar\_linea, que divide la instrucción en 4 partes (op, rs, rd, rt), busca los códigos binarios correspondientes en el diccionario y devuelve la instrucción en formato binario.

Finalmente, esta instrucción binaria se escribe en el archivo de salida. La estructura del código facilita poder agregar mas funciones y diccionarios para distintos tipos de instrucciones en las siguientes semanas.

* Decodificación de Instrucciones en Python

Para decodificar instrucciones de ensamblador MIPS a su equivalente binario. Este código decodifica instrucciones de ensamblador MIPS en formato op rs rt rd a su equivalente binario. Cada instrucción se busca en el diccionario codigos\_TipoR para obtener sus códigos de operación (op) y función (funct) en binario. Luego, se escribe la instrucción binaria en un archivo de salida.

La estructura del código permite agregar más funciones y diccionarios para distintos tipos de instrucciones en el futuro, lo que facilita su extensión y mantenimiento.

* Descripción de las instrucciones tipo R

Las instrucciones tipo R son un tipo de instrucción en lenguaje de máquina utilizadas en arquitecturas de computadoras, especialmente en procesadores tipo MIPS. Estas instrucciones involucran registros y operaciones aritméticas o lógicas. En una instrucción tipo R, los campos de la instrucción están organizados de la siguiente manera:

- OpCode: 6 bits

- RS: 5 bits (registro fuente 1)

- RT: 5 bits (registro fuente 2)

- RD: 5 bits (registro destino)

- Shamt: 5 bits (desplazamiento)

- Function: 6 bits (código de función específico de la operación)

La instrucción SLT (Set on Less Than) se utiliza para comparar dos valores y establecer un registro con el valor 1 si el primer valor es menor que el segundo, o con el valor 0 en caso contrario. Su formato en lenguaje ensamblador es:

SLT $rd, $rs, $rt

Donde $rd es el registro destino, $rs es el registro fuente 1 y $rt es el registro fuente 2. La operación SLT es útil en la implementación de estructuras de control de flujo condicionales.

* Investigación sobre la operación ternaria

La operación ternaria es una operación condicional que se encuentra en muchos lenguajes de programación. También se conoce como operador ternario. Su forma general es:

condición ? expresión1 : expresión2

Donde la condición es evaluada primero. Si es verdadera, se evalúa expresión1 y se devuelve su valor. Si es falsa, se evalúa expresión2 y se devuelve su valor. La operación ternaria es una forma concisa de escribir una estructura condicional if-else en una sola línea.

* Investigación sobre el proceso de compilación

El proceso de compilación es el proceso mediante el cual un programa escrito en un lenguaje de alto nivel se traduce a un lenguaje de máquina que la computadora puede entender y ejecutar. El proceso de compilación consta de varios pasos:

1. Análisis léxico: El código fuente se divide en tokens (palabras clave, identificadores, literales, etc.).

2. Análisis sintáctico: Se verifica la estructura gramatical del código fuente para garantizar que cumpla con las reglas del lenguaje.

3. Análisis semántico: Se verifica el significado del código para detectar errores semánticos.

4. Generación de código intermedio: Se genera un código intermedio que representa el programa de manera más abstracta.

5. Optimización de código: Se aplican diversas técnicas para mejorar el código intermedio y hacerlo más eficiente.

6. Generación de código objeto: Se genera el código objeto específico de la arquitectura de la computadora.

7. Enlazado: Se combinan los diferentes módulos de código objeto y bibliotecas para formar un programa ejecutable.

El proceso de compilación es fundamental para la programación en lenguajes de alto nivel, ya que permite que los programas escritos por los programadores se ejecuten en la computadora de manera eficiente.

* Unidad de Control (UC)

La Unidad de Control recibe una entrada de 6 bits correspondiente al Código de Operación (OpCode) y tiene tres salidas de 1 bit: MemToReg, RegWrite y MemToWrite. Además, tiene una salida de 3 bits, ALUOp, que se conecta a la ALU-Control. Esta unidad define las señales que se enviarán a los diferentes elementos del sistema dependiendo del OpCode que reciba.

* ALU-Control

Esta unidad recibe una entrada de 6 bits, que corresponde a los bits 0:5 de la instrucción, y una entrada de 3 bits que llega de la UC. Tiene una salida de 3 bits que se envía a la ALU para indicar la operación que debe realizar. La ALU-Control escucha a la UC y, dependiendo del código que reciba, hace caso o no de los bits 5:0 de la instrucción.

* Mux 2:1

Este multiplexor tiene dos entradas de 32 bits cada una y una entrada de 1 bit para seleccionar qué entrada se pasa a la salida. La salida es de 32 bits, donde se canaliza el dato seleccionado.

* TestBench

Para el TestBench del DataPath Tipo-R (DPTR), se debe hacer un test para los módulos nuevos. Se debe probar el DPTR enviándole 10 instrucciones, dos de cada una de las instrucciones mencionadas en la introducción. Se debe incluir una tabla con las instrucciones en formato ensamblador y otra con el código máquina correspondiente.

**Conclusiones**

* Kevin Manuel Sanchez Agredano

Se ha trabajado en la implementación de un DataPath en Verilog para ejecutar instrucciones básicas tipo R en un procesador, así como en la creación de un código en Python para decodificar instrucciones de ensamblador MIPS a su equivalente binario.

Este proyecto ha sido una oportunidad para aplicar conocimientos teóricos en el diseño y la implementación de sistemas digitales, así como en el desarrollo de herramientas de software para facilitar tareas relacionadas con el procesamiento de instrucciones de ensamblador. Ha permitido también explorar la integración de hardware y software en el contexto de sistemas informáticos, lo cual es fundamental en el campo de la arquitectura de computadoras.

* Diego Gerardo Quintero Gonzalez

En esta práctica, logramos diseñar e implementar un DataPath capaz de ejecutar instrucciones básicas tipo R. Se integraron los módulos existentes con nuevos módulos de control, permitiendo el funcionamiento adecuado del procesador en la ejecución de las instrucciones mencionadas.

Además se desarrolló un decodificador que tiene la finalidad de convertir instrucciones MIPS en código binario. El cuál diseñado para ser utilizado con archivos de texto que contienen instrucciones MIPS escritas en un formato específico.

Creo que como 1er fase pudimos lograr de buena forma los objetivos que se tenían planteados para poder presentar una primera etapa de calidad y a tiempo.

# Bibliografía

Rocha Pacheco, N. (2021). Diseño, implementación y validación de un núcleo de procesador basado en el conjunto de instrucción RISC-V. https://repositorio.uniandes.edu.co/entities/publication/ee3d1db5-1a73-4855-a3cb-c5827305bc30

Ramírez, C. A. R., Quintero, D. M. M., & Builes, J. A. J. (2021). Un Ambiente visual integrado de desarrollo para el aprendizaje de programación en robótica. *Investigación e Innovación en Ingenierías*. https://revistas.unisimon.edu.co/index.php/innovacioning/article/view/3957