Guía de Usuario

**Introducción**

Esta guía proporciona instrucciones para utilizar el DataPath implementado en Verilog para ejecutar instrucciones básicas tipo R en un procesador. El Single Cycle DataPath consta de varios módulos que se encargan de controlar el flujo de datos y realizar operaciones aritméticas y lógicas de tipo R por lo pronto.

El Decodificador de Instrucciones MIPS es una herramienta que convierte instrucciones MIPS en formato ensamblador a su equivalente en código binario de máquina. Está diseñado para la conversión de programas escritos en lenguaje ensamblador MIPS a un formato más adecuado para la ejecución en un procesador MIPS.

**Requisitos**

* Archivos de Python (Decodificador.py, data.asm, data.asm, data.txt)
* Software de simulación Verilog (ModelSim)
* Archivos Verilog del proyecto (ALU.v, Memoria.v, BancoRegistros.v, UnidadDeControl.v, Multiplexor.v, Tipo\_R, ALU\_Control.v, DataPath\_TipoR.v, tb\_DataPath.v, MemoriaDeInstrucciones.v, PC.v , Sign\_Extend.v, Sumador.v )

**Funcionalidades Principales**

1. Abrir Archivo: Permite al usuario seleccionar un archivo en formato ASM (Assembly) que contiene instrucciones MIPS.
2. Modificar un archivo ya previamente guardado para agregar instrucciones en ensamblador
3. Guardar Archivo: Permite al usuario seleccionar la ubicación y el nombre del archivo donde se guardará el resultado de la conversión.
4. Decodificar y Guardar: Realiza la conversión de las instrucciones del archivo de entrada y guarda el resultado en el archivo de salida especificado.
5. Salir: Cierra la aplicación.

**Pasos para utilizar el decodificador**

1. Para escribir las instrucciones en ensamblador (.asm) para el decodificador, es necesario escribir cada parte de cada instrucción separada por comas, haciendo salto de línea, y respetando el orden que es el que se toman los bits para cada instrucción.
2. Por ejemplo para las Tipo R tenemos: OP, rs, rt, rd, shamt y funct. Los datos que debemos ingresar en lenguaje ensamblador para estas instrucciones son: OP, rs(registro 1), rt(registro 2) y rd (registro de destino), ya que shamt y funct se asignan automáticamente a los 32 bits.
3. Estas instrucciones toman los datos que estan en el banco de registro y le dicen al procesador que hacer con ellos.
4. Ejemplos:

add,5,6,12

sub,5,8,10

slt,2,1,0

or,1,1,11

1. Abrir la aplicación.
2. Seleccionar el archivo de entrada haciendo clic en el botón "Abrir".
3. Especificar el archivo de salida haciendo clic en el botón "Guardar".
4. Hacer clic en el botón "Decodificar y Guardar" para realizar la conversión.
5. Una vez completada la conversión, se mostrará un mensaje indicando que el archivo se ha decodificado y guardado correctamente.
6. Abre el archivo de salida data.txt para verificar que las instrucciones hayan sido decodificadas correctamente.

**Pasos para utilizar el DataPath**

1. Compilación de los archivos Verilog

* Abre el software de simulación Verilog.
* Crea un nuevo proyecto y agrega todos los archivos Verilog del proyecto.
* Compila los archivos Verilog para verificar que no haya errores de sintaxis.

1. Simulación del TestBench

* Abre el archivo TB\_DaraPath.v en el editor de texto del software de simulación.
* Simula el TestBench para verificar el funcionamiento del DataPath inciando el ciclo de reloj en 0 para no ver errores al momento de correr el test.
* Observa los resultados de la simulación para asegurarte de que las instrucciones se ejecuten correctamente.

1. Análisis de los resultados

* Analiza los resultados de la simulación para verificar que el DataPath esté ejecutando las instrucciones correctamente.
* Verifica que las señales de control (MemToReg, RegWrite, MemToWrite, ALUOp) se estén generando correctamente para cada instrucción.

1. Depuración de errores

* Si encuentras algún error durante la simulación, revisa el código Verilog de los módulos involucrados para identificar y corregir el problema.
* Vuelve a compilar y simular el TestBench para verificar que se haya corregido el error.

**Conclusión**

Podrás utilizar el DataPath implementado en Verilog para ejecutar instrucciones básicas tipo R en un procesador. Es necesario cuidar el ciclo de reloj, que comienze en 0 cada vez que se vaya a ejecutar la simulación de Verilog.