Proyecto final
Dak-G

Materia: Arquitectura de computadoras.

Equipo: Alan Guzman, Kevin Sanchez, Diego Quintero, Gabriel Alcaraz.

Fase 1

Cronograma de actividades

Actividades Fase 1	22 de abril	23 de abril	24 de abril	25 de abril	26 de abril	27 de abril	28 de abril
Verilog							
Decodficador							
Documentacion							
Management							

Objetivos

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.

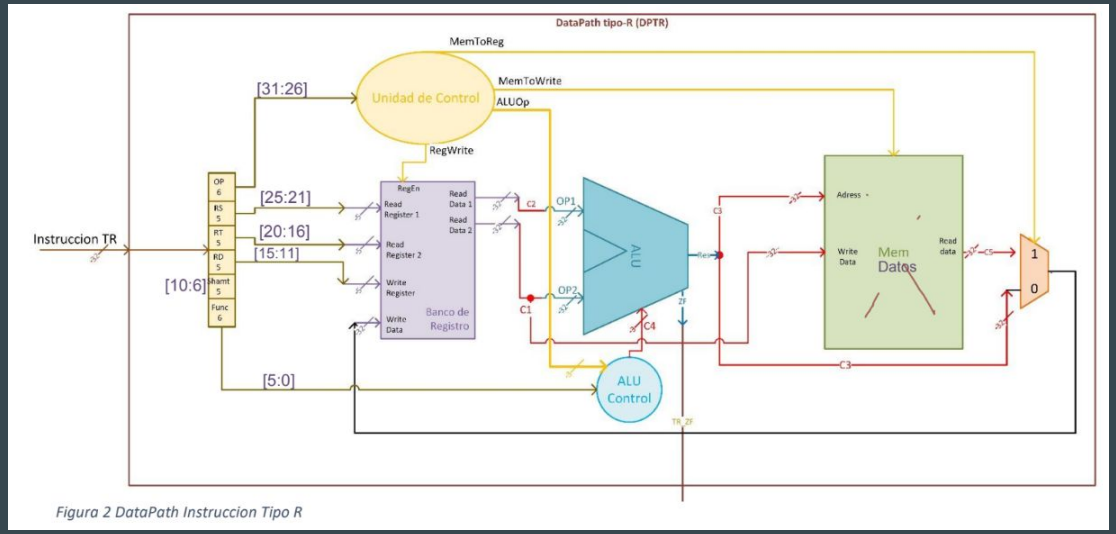


Figura 2 DataPath Instruccion Tipo R

Objetivos específicos

- 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 modulo, 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.

Objetivos específicos

- 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.

Entregables

Entregables [Reporte].pdf

- Descripción de la instrucciones tipo R, con énfasis en explicar la instrucción SLT.
- Investigación sobre la operación ternaria.
- Investigación sobre el proceso de compilación.

*Hay que leer la información y describirla con sus propias palabras.

Código en Verilog Archivos Verilog –

ALU.v, Mem.v, BancoReg.v, UnidadDeControl.v, Mux2_1_32.v, ALuControl.v, DPTR.v, TB_DPTR.v

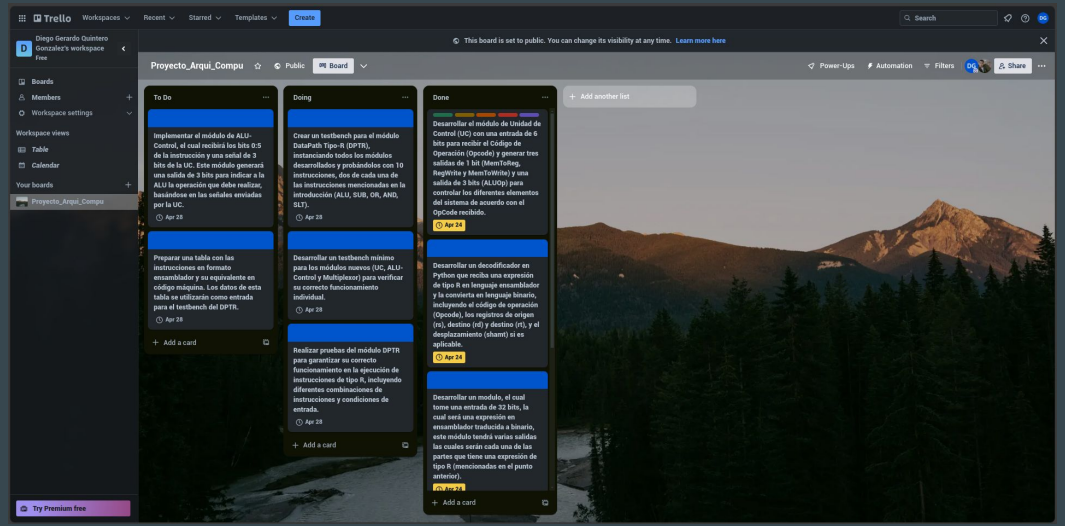
El reporte debe de presentarse con la simulación donde los resultados deben visualizarse en formato (ó radix) decimal.

*Nota: Los nombres de los archivos “.v”, módulos y elementos de entrada y salida son opcionales, los aquí presentados son ejemplos.

Trello

Se decidió poner las metas en la plataforma “Trello” para poder llevar un mejor control de las distintas etapas en las que se encuentran las mismas, así como también el poner fecha y hora de vencimiento a cada una para poder realizarlas y entregarlas en tiempo y forma.

https://trello.com/b/MYGzWthO/proyectoarquicompu



Guia para git

Debido a que se estará trabajando con git para trabajar en equipo a lo largo de desarrollo del proyecto, se mencionarán a continuación algunos de los comandos más necesarios para utilizar git mediante la terminal.

- Clonar un repositorio remoto: **git clone <URL_del_repositorio>**
- Agregar cambios al área de preparación (stage): **git add <archivo>**
- Confirmar cambios preparados: **git commit -m "Mensaje del commit"**
- Actualizar el repositorio local con los cambios remotos: **git pull origin <rama>**
- Enviar cambios locales al repositorio remoto: **git push origin <rama>**
- Crear una nueva rama: **git branch <nombre_de_rama>**

Guia para git

- Cambiar a una rama específica: **git checkout <nombre_de_rama>**
- Combinar cambios de una rama a otra: **git merge<nombre_de_otra_rama>**
- Ver el estado actual del repositorio: **git status**
- Ver el historial de commits: **git log**
- Descartar cambios locales no deseados: **git checkout -- <archivo>**
- Revertir un commit (útil para deshacer cambios): **git revert <ID_del_commit>**

Para repositorios remotos:

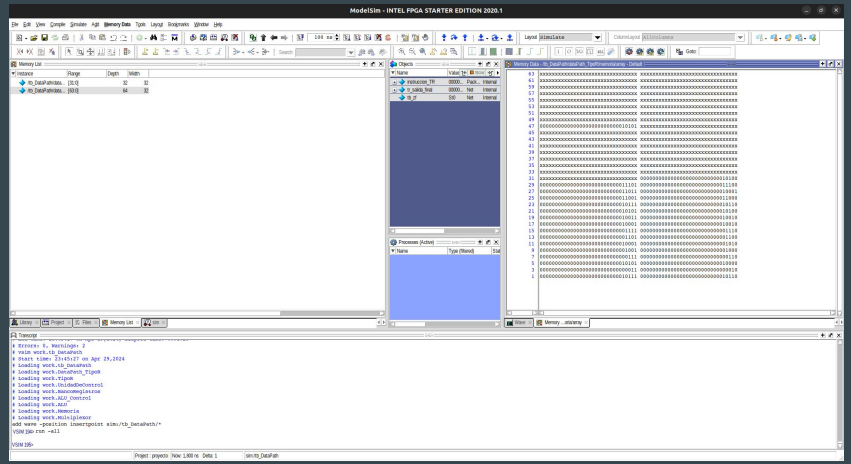
- Agregar un repositorio remoto de GitHub a un repositorio local de Git: **git remote add origin <URL_del_repositorio_remoto>**
- Enviar cambios locales a GitHub (push): **git push origin <rama>**
- Clonar un repositorio de GitHub: **git clone <URL_del_repositorio_en_GitHub>**
- Obtener cambios de GitHub (pull): **git pull origin <rama>**
- Ver los repositorios remotos configurados: **git remote -v**

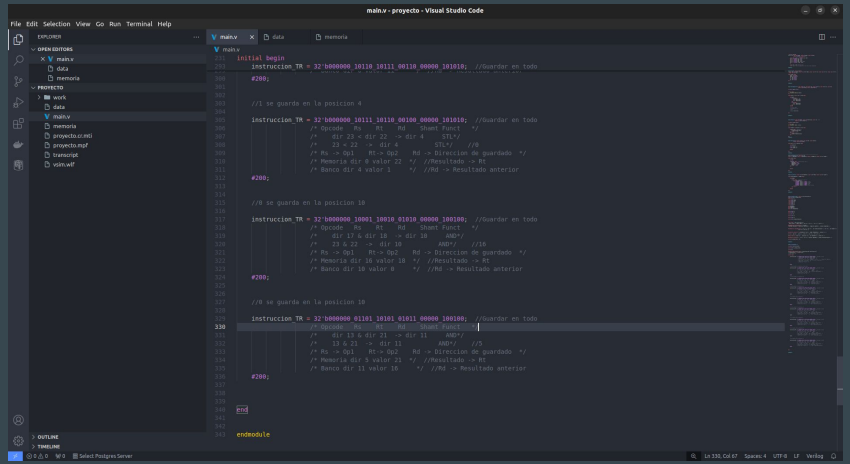
Trabajo realizado por: Diego Quintero - Management

- Realización de la presentación, en la cual se describen el cronograma de actividades, los objetivos, entregables, la guía de git. Todo lo anterior relacionado a la primera fase del proyecto.
- Revisión y apoyo en dudas que tenían mis compañeros durante la realización de sus actividades

Trabajo realizado por: - Verilog

- Creacion de los modulos: ALU, ALU_Control, BancoRegistros, DataPath_TipoR, Memoria, Multiplexor, tb_DataPath, TipoR, UnidadDeControl.
- Probar la simulación y el guardado de los resultados de las operaciones





Trabajo realizado por: - Decodificador (Python)

- GUI Decodificador 1, el cual convierte instrucciones de Tipo R en lenguaje ensamblador a lenguaje máquina o binario. Incluye una GUI (Graphic User Intreface) básica en el que se ingresan archivos en “.asm” y se selecciona el archivo de salida que sea un “.txt” en el que se sobreescribiran los datos.

Trabajo realizado por: Kevin Sanchez - Documentación

- Reporte: En él se incluyen una introducción general de todo el proyecto y la fase 1, objetivos (generales y particulares) en los cuales se trabajo durante la fase 1 con todo el equipo, desarrollo donde se explica todo lo que se realizó e incluyendo investigaciones que se pedían y conclusiones de cada uno de los integrantes.
- Readme (github): En él se muestran Título e imagen de portada, insignias, índice, descripción del proyecto, estado del proyecto, demostración de funciones y aplicaciones, acceso al proyecto, tecnologías utilizadas, personas contribuyentes, personas desarrolladoras del proyecto, conclusión.
- Guia de usuario: Se incluye una pequeña introducción, los requisitos que se requieren para poder utilizar el programa, pasos para utilizar los programas, conclusión.

Nuevos roles para la fase 2

- Documentación - Gabriel Alcaraz
- Decodificador - Kevin Sanchez
- Verilog - Diego Quintero
- Management - Alan Guzman

Guía de usuario

El decodificador de es un programa que convierte instrucciones MIPS en código binario. Está diseñado para ser utilizado con archivos de texto que contienen instrucciones MIPS escritas en un formato específico. Aquí hay una descripción paso a paso de cómo utilizar este decodificador.

1. Preparación de los archivos de entrada y salida

- Antes de utilizar el decodificador, asegúrate de tener un archivo de entrada que contenga las instrucciones MIPS que deseas decodificar en la carpeta del programa llamado “data.asm”.
- Además, prepara un archivo de salida donde se escribirán las instrucciones decodificadas en formato binario también en la misma dirección, llamado “data.txt”.

2. Ejecución del decodificador

- Para iniciar el proceso de decodificación, ejecuta la función decodificar_instrucciones proporcionando el nombre del archivo de entrada y el nombre del archivo de salida como argumentos.
- Por ejemplo: decodificar_instrucciones ("entrada.asm", "salida.txt").

Guía de usuario

3. Observación de los resultados

- Una vez que el decodificador haya completado el proceso, verifica el archivo de salida especificado.
- En el archivo de salida, encontrarás las instrucciones MIPS decodificadas en formato binario.

4. Interpretación de los resultados

- Cada línea en el archivo de salida corresponde a una instrucción MIPS decodificada en formato binario.
- Cada instrucción binaria está compuesta por varios campos, como el opcode, los registros fuente y de destino, y el funct.
- Utiliza estos resultados binarios según sea necesario para tu aplicación específica.

5. Personalización del decodificador

- Si se desea agregar más códigos de operación MIPS al diccionario codigos_TipoR, se puede llevar a cabo tomando en cuenta las necesidades que se tengan.
- Se debe asegurar de seguir el formato correcto de los códigos de operación y funciones MIPS.

6. Creación de archivos

- Se puede utilizar la función creararchivos para crear los nombres de archivo de entrada y salida de manera conveniente.

Conclusión

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. Está diseñado para ser utilizado con archivos de texto que contienen instrucciones MIPS escritas en un formato específico.