Proyecto final

Materia: Arquitectura de computadoras.

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

Fase 1

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.

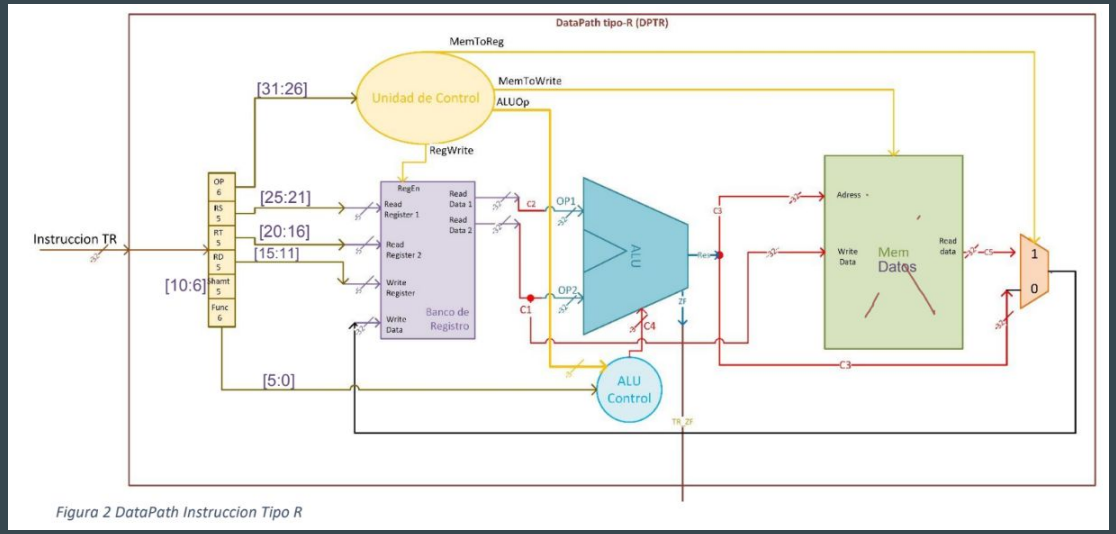


Figura 2 DataPath Instruccion Tipo R

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

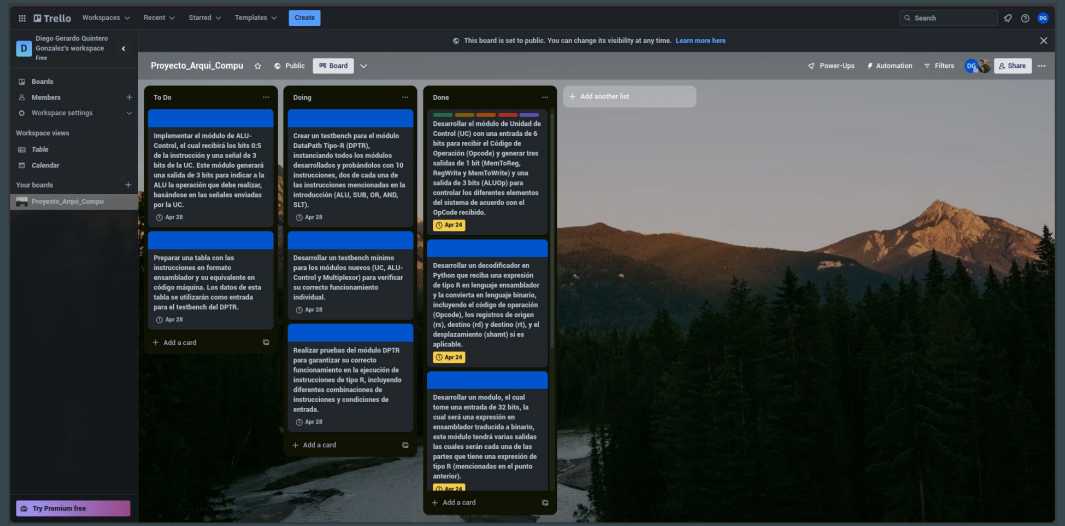
El reporte debe de presentarse con la simulación donde los resultados deben visualizarse en formato (ó radix) decimal.

*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**