**Instituto Tecnológico y de Estudios Superiores de Occidente**

Departamento de Electrónica, Sistemas e Informática

**ARQUITECTURA COMPUTACIONAL**

Logo, company name

Description automatically generated

**PRÁCTICA 2**

Presenta

**Laura Griselda González Camacho – 734049, ISC**

**Jazmín Nahíl Méndez Santana – 734043, ISC**

Arquitectura computacional

Fecha: 10/04/2023

## Clock rate máximo de la implementación del RISC-V

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

## CPU time

* CPI: 1 ciclo p/instrucción
* IC: 277

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

* Clock rate: 39.42 MHz
* CPU Time = 277 / 39.42
* CPU Time = 7.0269 µs

## Micro-arquitectura propuesta

DIAGRAMA

Diagrama

Descripción generada automáticamente

## Explicación de agregar o modificar módulos en Verilog

* CONTROL: Agregamos valores de control para las instrucciones de jal y jalr. También tipos de instrucciones.
* ALU CONTROL: Agregamos las instrucciones necesarias.
* ALU: Agregamos las operaciones de las instrucciones necesarias.
* INMEDIATE UNIT: Agregamos las estructuras de los buses de instrucción para los diferentes tipos de instrucciones.
* PC-REGISTER: Inicializamos en 32'h00400000.
* REGISTER Y REGISTER-FILE: Agregamos valor de inicialización a todas las instancias de Register. Todas inician en 0 y el stackpointer inicia en 32’h400;
* TOP-LEVEL: Los cambios a todos los módulos, dos variables de output en su módulo para el clock rate. 4 MUX.
  + Mux para las Branch: indica si la instrucción tiene la bandera de Branch encendida. Dependiendo de eso le pasa al siguiente valor del pc lo correspondiente según el tipo de instrucción y sus datos.
  + Mux para la memoria: indica si el resultado que se escribirá en el registro indicado es sacado de la memoria o si va directo del resultado de la ALU.
  + Mux para Jal: indica si se escribe en el registro la salida de la memoria o el resultado del registro ra.
  + Mux para Jalr: indica si se escribe como el siguiente pc el valor de ra (el pc actual más 4) o el resultado del mux de la Branch.

## ModelSim de instrucciones importantes

* BEQ
  + Código ASM

Imagen que contiene Tabla

Descripción generada automáticamente

* + Simulación
    - Código BEQ: 1001

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

* BNE
  + Código ASM

Texto

Descripción generada automáticamente

* + Simulación
    - Código BNE: 1010

Diagrama

Descripción generada automáticamente

* JAL, JALR
  + Código ASM

Tabla

Descripción generada automáticamente con confianza baja

* + Simulación
    - Código JAL: 1011
    - Código JALR: 1100

Una captura de pantalla de un videojuego

Descripción generada automáticamente con confianza media

* LW, SW
  + Código ASM

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

* + Simulación
    - Código SW: 1000
    - Código LW: 1000

Interfaz de usuario gráfica, Escala de tiempo

Descripción generada automáticamente

## Simulación Torres de Hanoi con 3 discos

* Se llena la Torre A

Imagen que contiene Diagrama

Descripción generada automáticamente

* Se pasa el disco 1 de la Torre A a la Torre C

Imagen que contiene Texto

Descripción generada automáticamente

* S pasa el disco 2 de la Torre A a la Torre B

Diagrama

Descripción generada automáticamente con confianza media

* Se pasa el disco 1 de la Torre C a la Torre B

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

* Se pasa el disco 3 de la Torre A a la Torre C

Diagrama

Descripción generada automáticamente con confianza baja

* Se pasa el disco 1 de la Torre B a la Torre A

Imagen que contiene Texto

Descripción generada automáticamente

* Se pasa el disco 2 de la Torre B a la Torre C

Imagen que contiene Texto

Descripción generada automáticamente

* Se pasa el disco 1 de la Torre A a la Torre C

Diagrama

Descripción generada automáticamente con confianza baja

* Torre C al final el programa

Imagen que contiene Texto

Descripción generada automáticamente

## Código hanoi.asm



## Link del repositorio privado en Github

<https://github.com/lauraggc/Practica2AC.git>

\*\* En program memory alterar la ruta del readme a la carpeta local, en src/hanoi.dat.

## Conclusiones

**Jazmín Méndez**

En general, el uso de Quartus, el lenguaje Verilog y la arquitectura vista en clase fueron temas bastantes complejos de entender, sin embargo, una vez manejados considero que pudimos avanzar de manera eficiente. Sin embargo, encontramos retos importantes, como las branches, los jal y jalr, que pueden ser complicados de implementar requiriendo una mayor planificación para evitar errores en la ejecución del código. También me gustaría mencionar que a través de esta práctica pude comprender de mejor manera el flujo de la arquitectura de las instrucciones, observando cómo se relacionan los diferentes módulos y sus conexiones. En resumen, considero que la práctica fue fructífera para mis conocimientos de la asignatura, donde en el proceso nos encontramos con varios retos, problemas con las herramientas y más, pero supimos superarlos y entregar un buen código.

**Laura González**

Esta práctica me ayudó a comprobar que había entendido los temas de Single Cycle en clase. Fue de mucha ayuda haber puesto atención y haber participado en esas clases, ya que no se me presentaron dudas que no pudiera resolver en lo que respectaba a los diagramas. Fue muy buen ejercicio de repaso de lo que hemos visto hasta ahora y tuvimos que usar mucha lógica para analizar cómo acomodar los módulos para que dieran el resultado que nosotras esperábamos. Había factores que no tomamos a consideración al principio que después resultaron un problema, por ejemplo que nuestro código de asm era muy largo y tuvimos que aumentar el program memory Depth. Me ayudó mucho a aprender cosas que no había razonado en los ejercicios en clase. La simulación de ModelSim presentó un reto ya que no podíamos analizar con exactitud dónde teníamos errores de lógica, por lo que revisábamos todo múltiples veces hasta solucionar el error. Por último, al principio fue un reto analizar los resultados de nuestro código por medio de la simulación de ModelSim. Estamos acostumbradas a revisar los resultados de forma directa, y en la simulación teníamos que encontrar la variable correcta que nos mostraría lo que deseábamos saber.