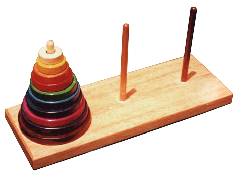
**Instituto Tecnológico y de Estudios Superiores de Occidente**

**Departamento de Electrónica, Sistemas e Informática**

Arquitectura de Computadoras

Email:

ie715316@iteso.mx [ie713891@iteso.mx](mailto:ie713891@iteso.mx)

****

**Práctica 2:**

Diseño y simulación de un procesador uni-ciclo basado en la arquitectura MIPS

**Team members:**

José Isaac Gallegos Velica Naum Jahaziel Nuño Contreras

**Teacher:** Pedro Saldaña Zepeda

Tlaquepaque, Jalisco, 16, noviembre, 2019

**Introducción:**

Como se vio en la práctica pasada, las torres de Hanoi tenían su complejidad particular. Ahora bien, el objetivo de esta práctica es implementar en Verilog un procesador basado en la arquitectura MIPS, el cual sea capaz de ejecutar las instrucciones descritas en la descripción de la práctica.

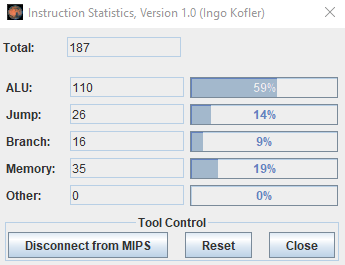
El reporte explica todo el desarrollo de la misma, el código para la construcción del single cycle MIPS, los diagramas de flujo del algoritmo, así como los resultados y las debidas conclusiones.

El programa desarrollado fue hecho para soportar las siguientes instrucciones: add, addi, sub, or, ori, and, andi, lui, nor, sll, srl, lw, sw, beq, bne, j, jal, jr.

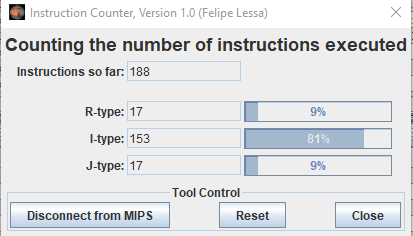
**Desarrollo**

1. **Incluir en el reporte el IC, CPI, clock rate y CPU time para implementación del MIPs. En el caso del IC se tiene que especificar el porcentaje de instrucciones de tipo R I y J.**

Porcentaje de cada tipo de instrucciones



IC:

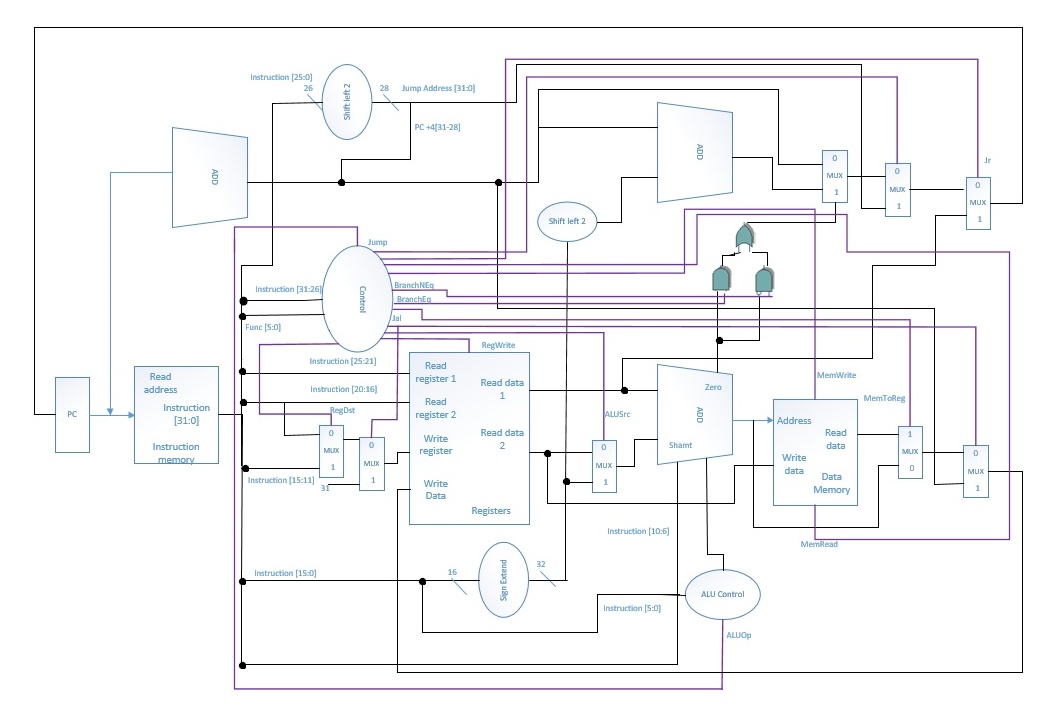


**Clock rate =** 250GHz

178 ciclos de reloj

**CPU time =** Clock cycles / clock rate = 178 / 250GHz = 712ps

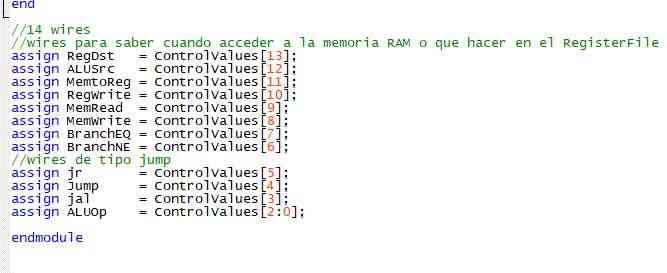
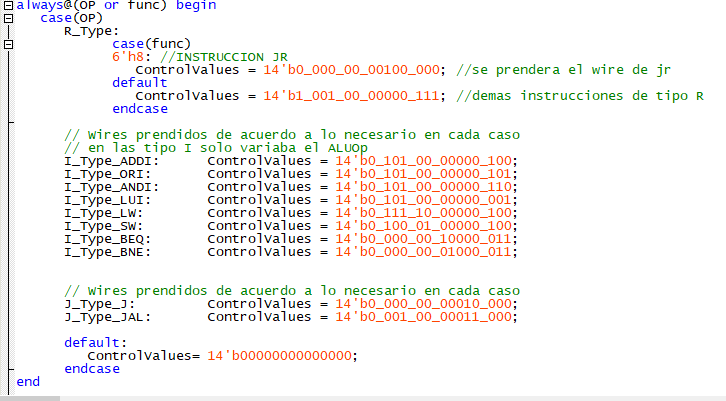
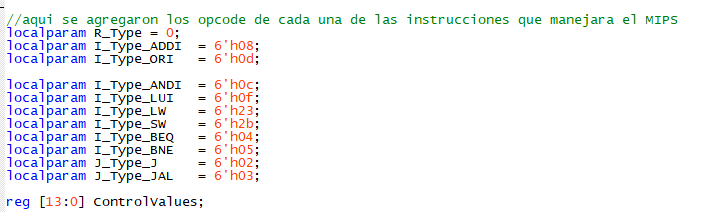
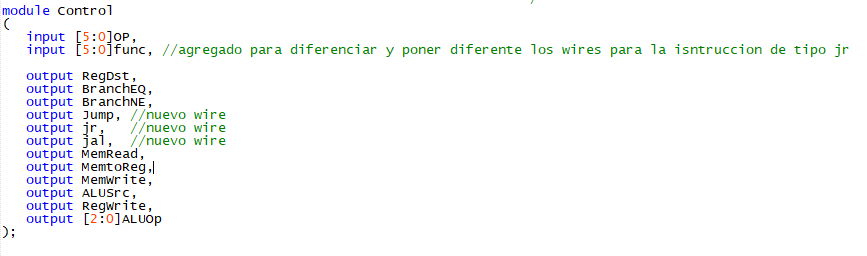
**CPI = CPU Clock cycles / Instruction Count =** 178/188 = 0.9468

**2. Incluir la micro-arquitectura propuesta (esquemático como los vistos en clase hecho un Visio o un programa equivalente), es decir, las modificaciones que permiten al MIPS ejecutar el programa de las torres de Hanói.**

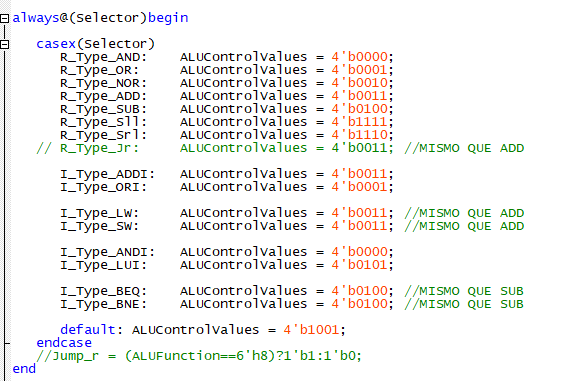
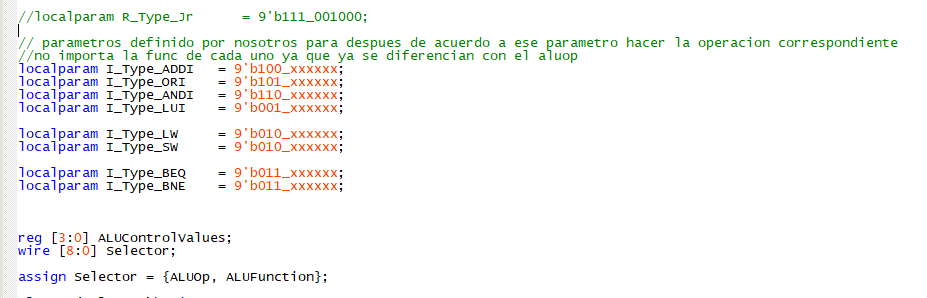
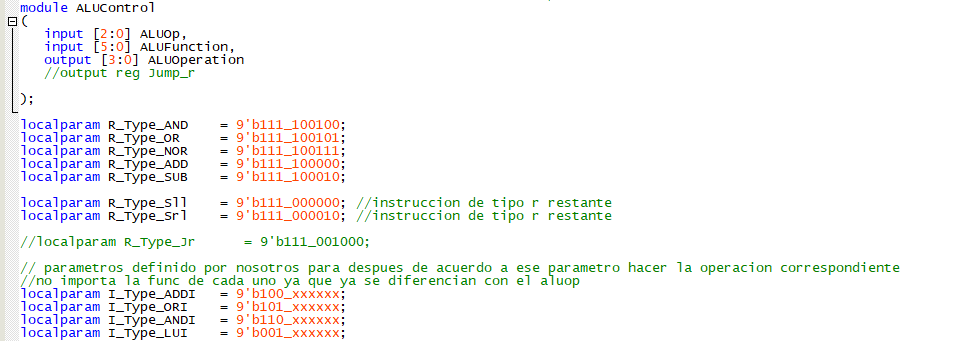
Para una mejor visibilidad, el diagrama se encuentra adjunto con los entregables.

1. **Se debe documentar las partes de los códigos de Verilog que se modificaron, estos es las instrucciones que se agregaron en las unidades de control así como la incorporación de nuevo hardware que permite la ejecución de dichas instrucciones (solo modificaciones).**

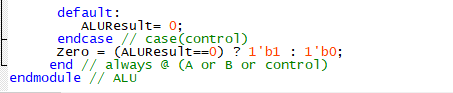
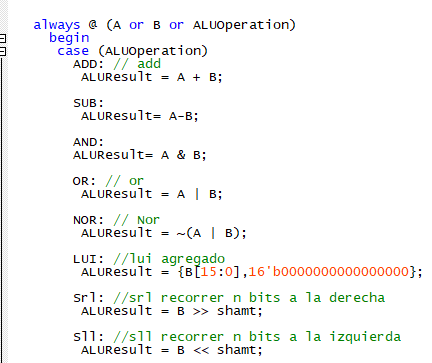
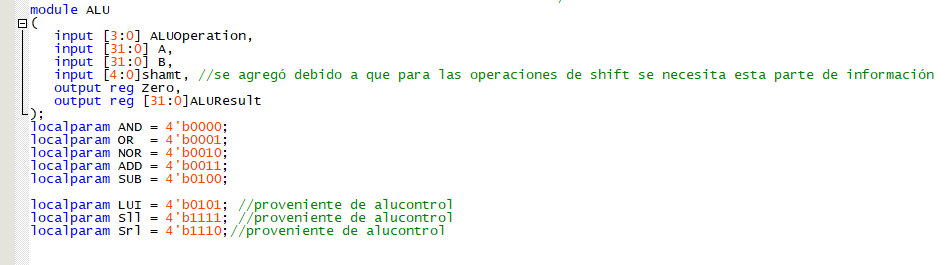
**Modulo control**



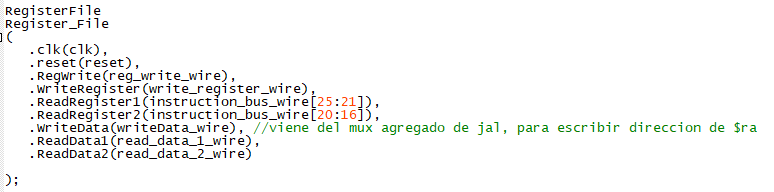
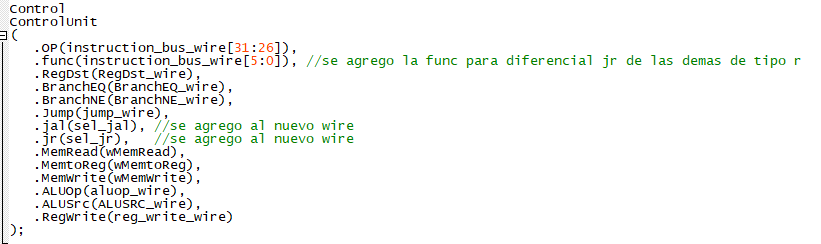
**Modulo ALU control**

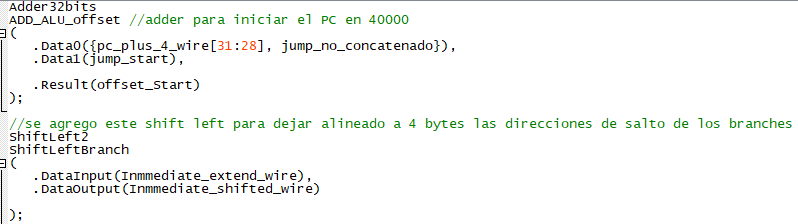


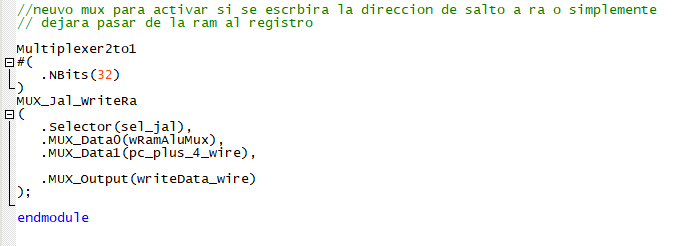
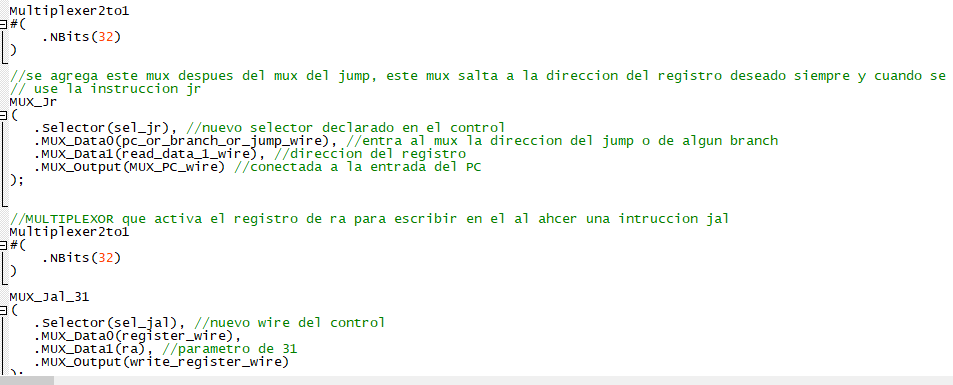
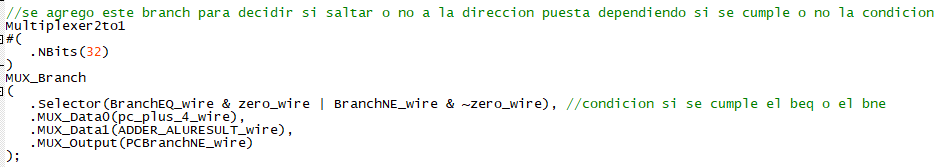
**Modulo ALU**



**Modulo TOP**

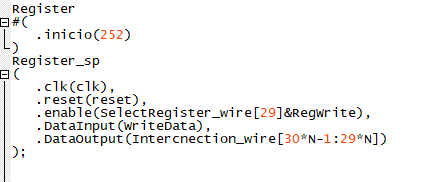






**Módulo Register File**

**Aquí se movio el sp hasta arriba de la memoria RAM**



**4. Explicación de la razón por la cual se modificó (o agrego módulos) el modelo en verilog.**

Se modificaron 4 principales bloques durante el desarrollo de la práctica, el top, ALUControl, ALU, y el módulo de control.

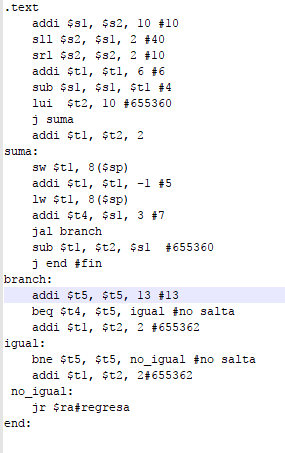
El bloque de ALU, se modificó en el sentido de que se agregó otra entrada, la entrada que decía la cantidad a recorrer en operaciones de tipo shift, debido a que para las operaciones de shift se necesita esta parte de información proveniente de la instrucción. Además, agregaron las operaciones que faltaban por realizar, por ejemplo, se agregó la operación de LUI, los shift.

En ALUControl, simplemente se cambió en el sentido de definir los parámetros correspondientes a cada tipo de instrucción. Mientras que el Control, fue uno de los módulos con mayor cambio ya que, se agregaron nuevos wires (jal, jr, beq, Branch not equal), con el fin de usarlos como selector en los multiplexores agregados en el top para poder implementar las instrucciones como jr, jal y los branches, en general todas las instrucciones de tipo j.

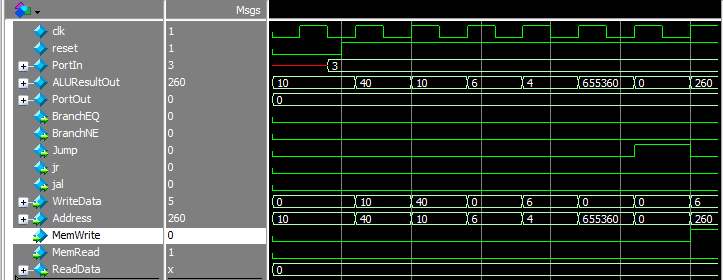
El top se cambió en el sentido en que se conectaron los nuevos multiplexores agregados, se agregó el offset al PC para poder iniciarlo en la misma dirección en la que se encuentra en MARS. Entre los multiplexores agregados, está el mux del Branch, el cual dice a que dirección se salta en caso de cumplir la condición del Branch, el mux de jr, el cual manda a saltar a la dirección del registro, además se agregaron 2 muxes para la instrucción jal, uno para mandar a escribir en el registro 31, correspondiente a jr, y otro afuera de la memoria RAM para escribir la dirección correspondiente en ra.

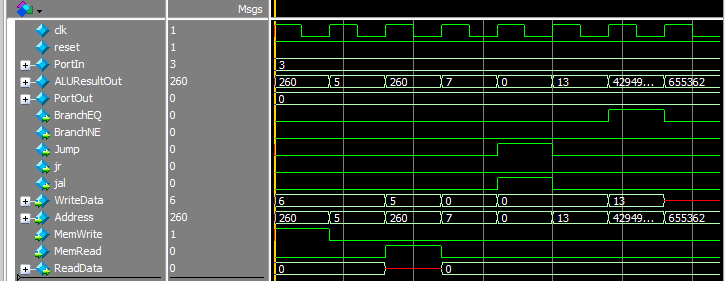
**5. Se debe entregar un programa y simulación en Modelsim para cada una de las siguientes de las instrucciones, beq, bne, lw, sw, j y jr.**

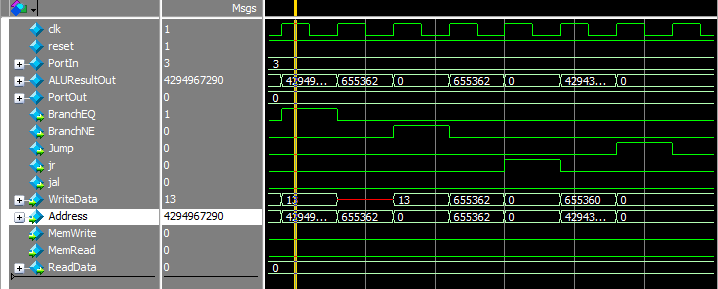
La simulación fue hecha con el siguiente programa



**Resultados**

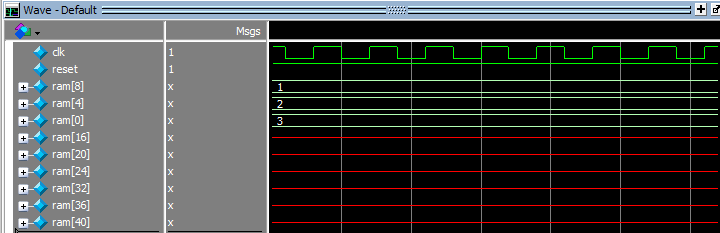




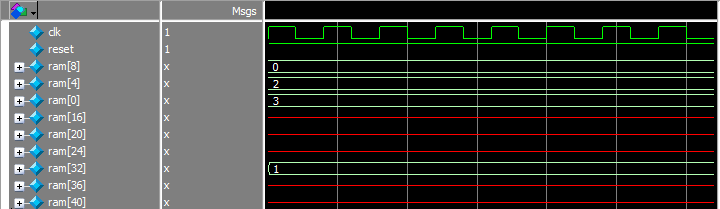


**6. Se debe incluir los resultados de simulación para el caso de 3 discos. Es decir, segmentos de simulación donde se muestren los movimientos en la memoria de dato (ver video torres de Hanoi en modelsim).**

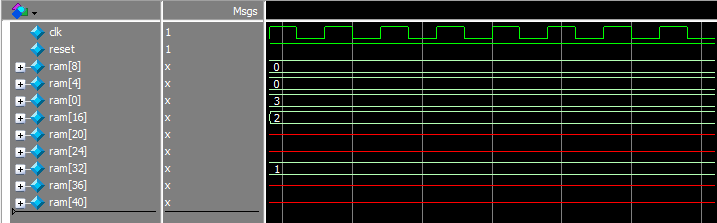
Inicialización de la primera torre



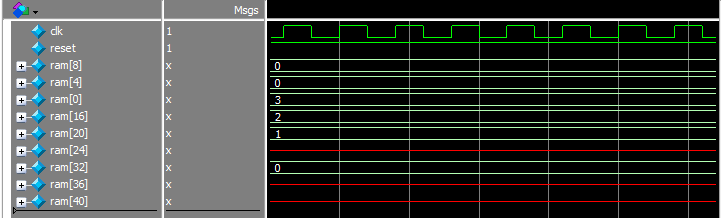
El disco 1 se pasa a la torre final



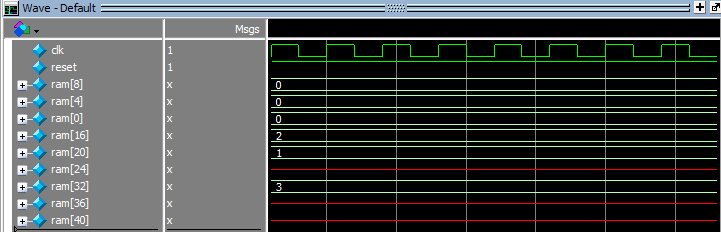
Disco 2 se pasa a la torre auxiliar



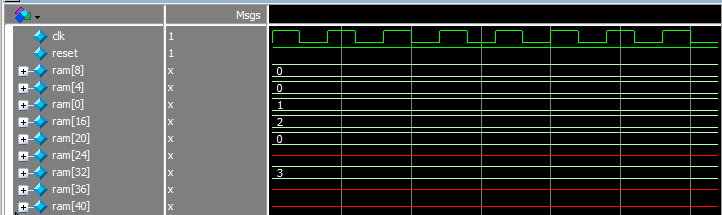
Disco 1 pasa a la torre auxiliar



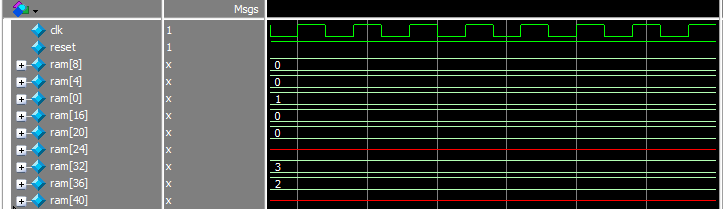
Disco 3 pasa a la torre final



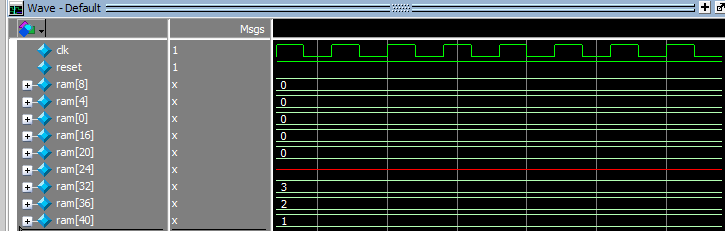
Disco 1 pasa a la torre inicial

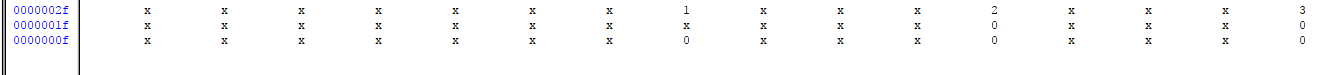


Disco 2 pasa a la torre final

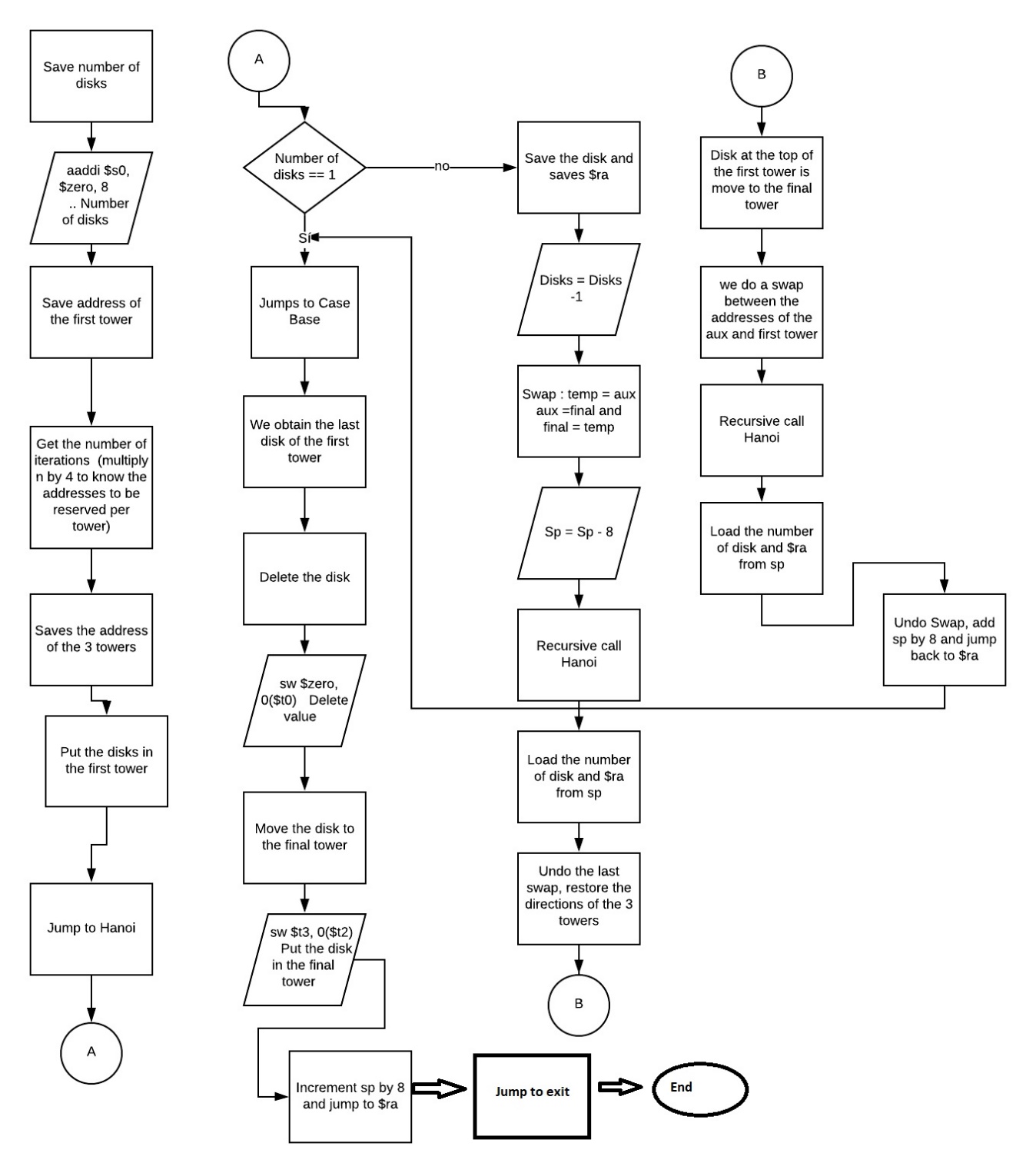


Primer disco pasa a la torre final, fin





**7. Diagrama de flujo donde se explique la forma en la que se implementó el programa torres de Hanói.**



**8. El archivo hanoi.asm con el código de la práctica 1, ya con las correcciones necesarias si no funcionó al 100%, y el archivo text.dat. Su procesador debe ser capaz de resolver el problema de las torres de Hanói sin errores.**

En el archivo hanoi se cambiaron las direcciones de las torres, para que concordaran con las direcciones de la RAM de nuestro MIPS.

**10. Se deben incluir los archivos Verilog que se generaron para práctica**

Véase carpeta adjunta

**Enlace a github**

[**https://github.com/jahaziel2903/MIPS-Single-cycle.git**](https://github.com/jahaziel2903/MIPS-Single-cycle.git)

**Conclusiones**

Isaac:

Se logro el objetivo de la práctica, pudimos construir el MIPS uniciclo y probar el programa de Hanói sobre él. Con esto aprendimos como se desarrolla toda la etapa de control sobre los bloques de memoria y la ALU, con lo que reforzamos todos los aprendizajes en clase del flujo de datos en el procesador y la arquitectura de las instrucciones manejadas.

Jahaziel:

Con el desarrollo de la práctica se logró entender el datapath de las distintas instrucciones implementadas dentro del single cycle MIPS Processor, así como la arquitectura de la misma.