Práctica 1 – Organización y Arquitectura de Computadoras

Torres de Hanoi

ITESO, Universidad Jesuita de Guadalajara

Diego Arturo Orozco Sánchez – 739246

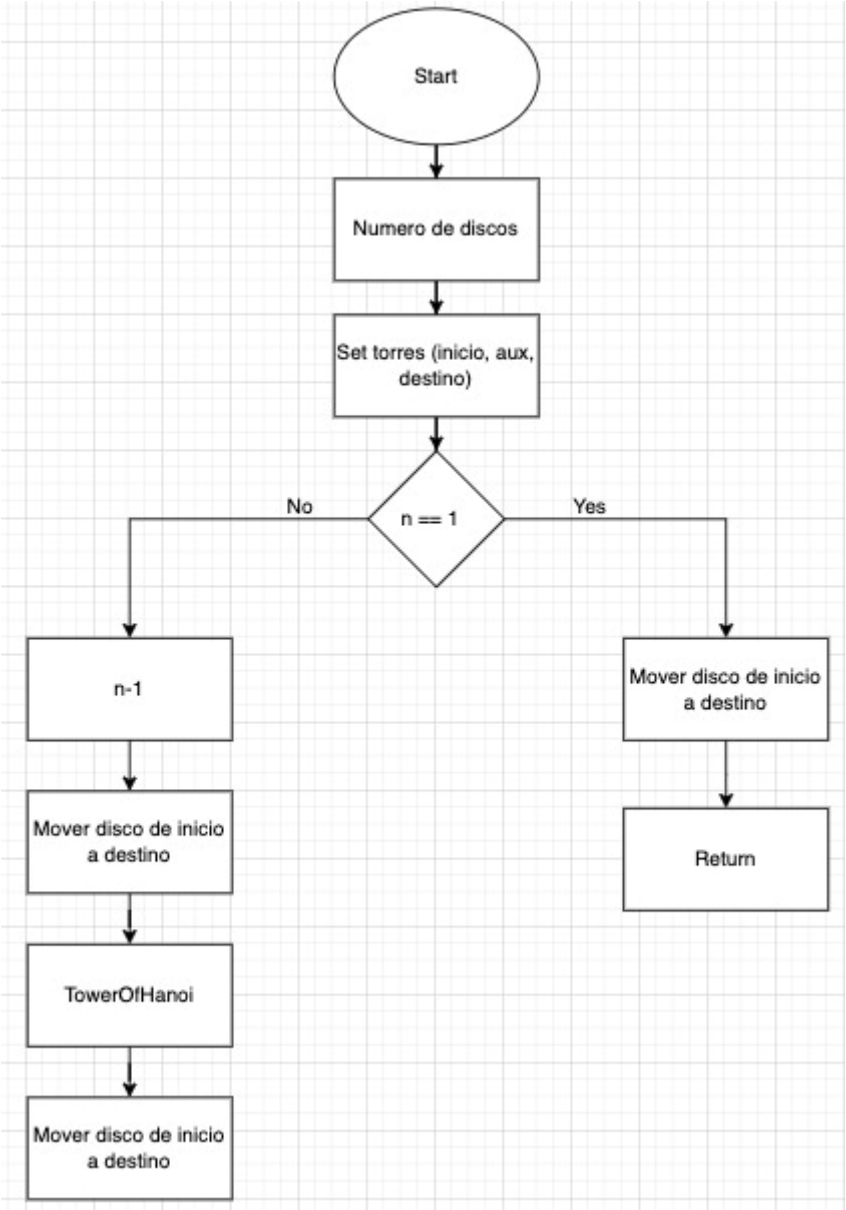
Isaac Ramírez Robles-Castillo – 748062

Profesor:

Juanpablo Ibarra Esparza

Fecha: 31/10/2023

Diagrama de flujo:



Decisiones de diseño:

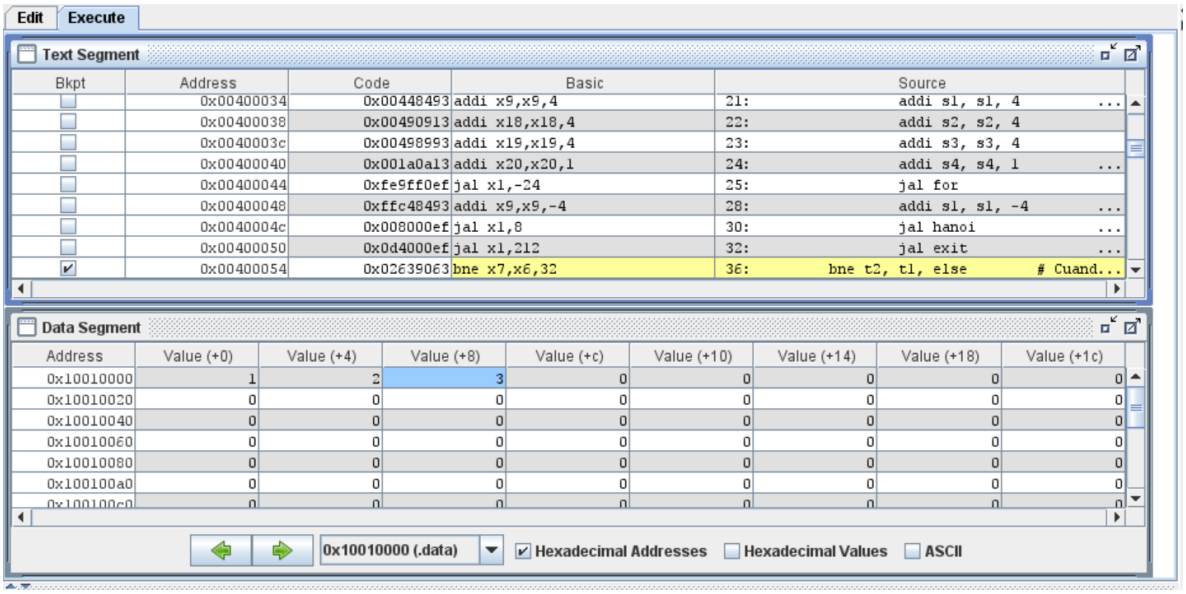
Principalmente nos basamos en el código de Fibonacci que hicimos en clase, además del algoritmo de C que tomamos como referencia.

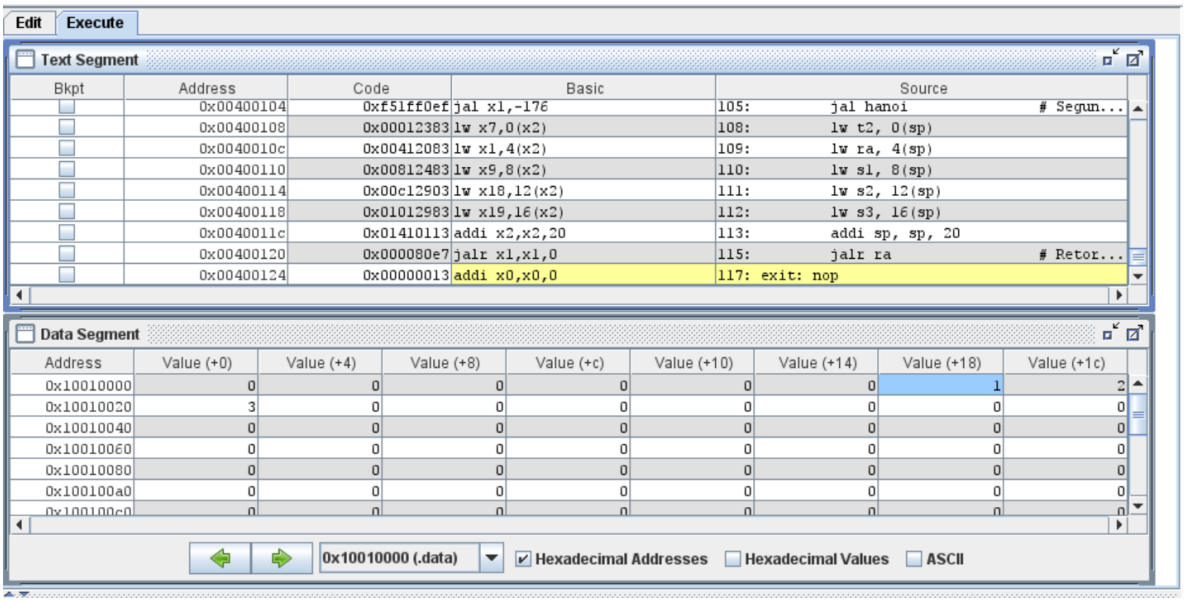
En el main, primero definimos todas las variables que vamos a usar en el programa, como la inicialización de la RAM usando lui, el número de discos, la variable del ciclo for al llenar la primera torre, y demás.

Primero se llena la primera torre usando una simulación de un ciclo for, que se llama a sí mismo recursivamente hasta que termine.

Después buscamos el caso default del algoritmo (dónde solo queda un disco), y el else nos lleva a la recursión fuerte del algoritmo. A continuación, se especifica cómo funciona la recursión y el stack.

Simulación con 3 discos:

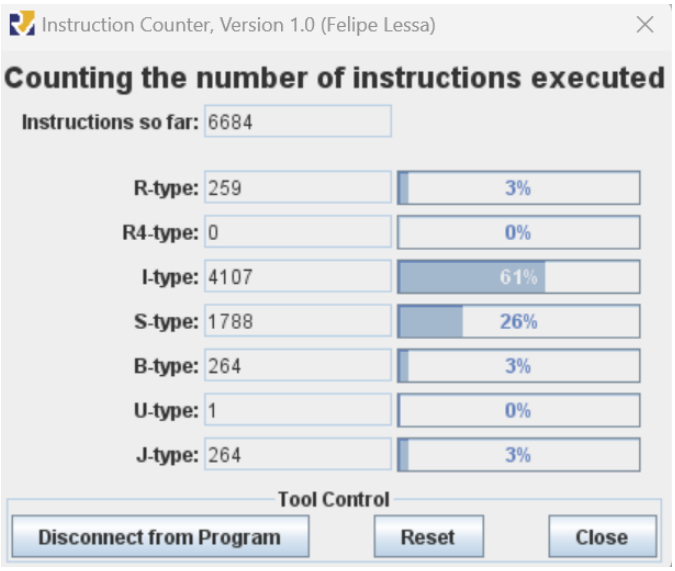




Análisis del comportamiento del stack con 3 discos:

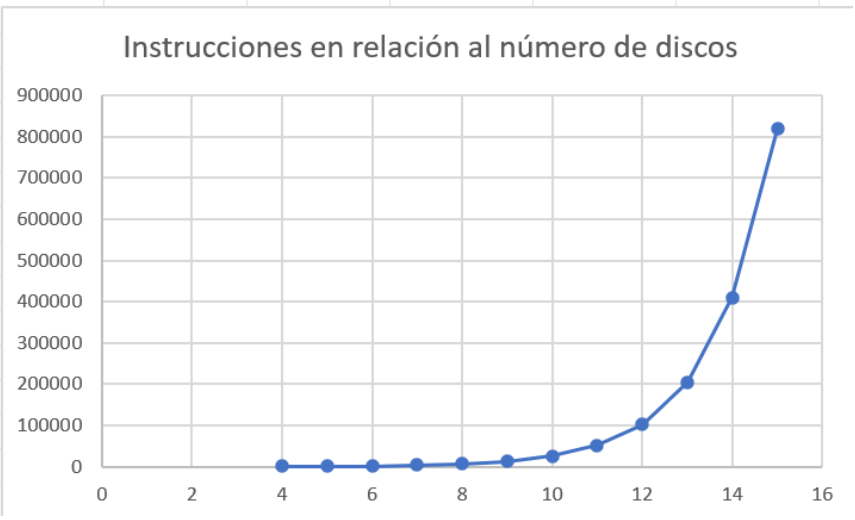
Para entender lo que hace el stack con 3 discos, debemos referirnos a la parte del código dónde empezamos la recursión. Se reservaron 20 bits en el stack, pues necesitamos 4 bits para 5 registros, y lo primero de que debemos hacer es el push con sw como tal. Después, con esa memoria liberada, se hacen los movimientos necesarios para al final reponer la memoria con lw.

Instruction count para 8 discos:



Gráfica del instruction count de 4 a 15 discos:

DISCOS	IC
4	400
5	807
6	1614
7	3221
8	6428
9	12835
10	25642
11	51249
12	102456
13	204863
14	409670
15	819277



Conclusiones:

Diego: Personalmente, creo que haber aprendido ensamblador es bueno para saberlo, pero como ingenieros en sistemas creo que está de más. La programación en ensamblador es bastante diferente a lo que yo estoy acostumbrado, y honestamente nos costó bastante. Como tal, el algoritmo de C en el que nos basamos es fácil de entender, y también el problema de las Torres de Hanoi, pero traducirlo a ensamblador y si fue una práctica difícil. El saber como manejar los registros, las variables temporales, manejo de memoria del stack, y demás conocimientos que eran muy importantes para saber implementarlo. Sin embargo, me gustó probar mis habilidades en un reto así de demandante.

Isaac: En el desarrollo de la práctica se tuvo analizar a detalle el algoritmo de Torres de Hanoi para plasmarlo a un esquema de ensamblador, se profundizó en el uso de variables (declaraciones y guardado de información), manejos de memoria para guardar las posiciones de cada variable y poder interpretar los resultados "gráficamente". Se uso recursividad para poder implementar un código eficiente, practico y poder usar el número de discos que se necesiten. Personalmente encontré muy desafiante la actividad debido a la diferencia de lógica que se necesita para pasar un código a ensamblador y el uso de la memoria me pareció complicado ya que se necesita tener una buena noción de los comportamientos en el stack.