

Instituto Tecnológico y de Estudios
Superiores de Occidente – ITESO



ITESO

Universidad Jesuita
de Guadalajara

Materia: Organización y Arquitectura de computadoras

Maestro: Álvaro Gutiérrez Arce

Práctica 1

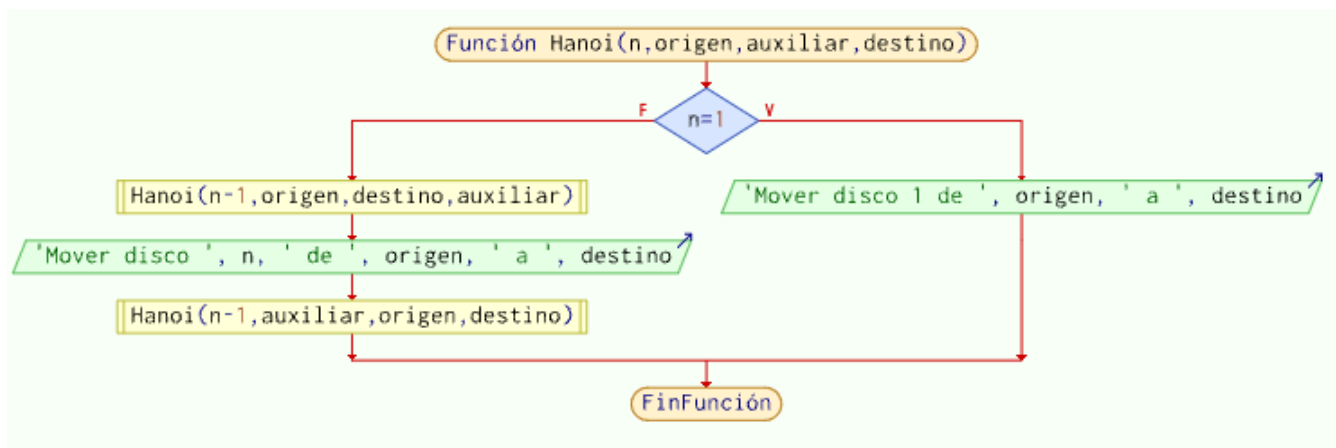
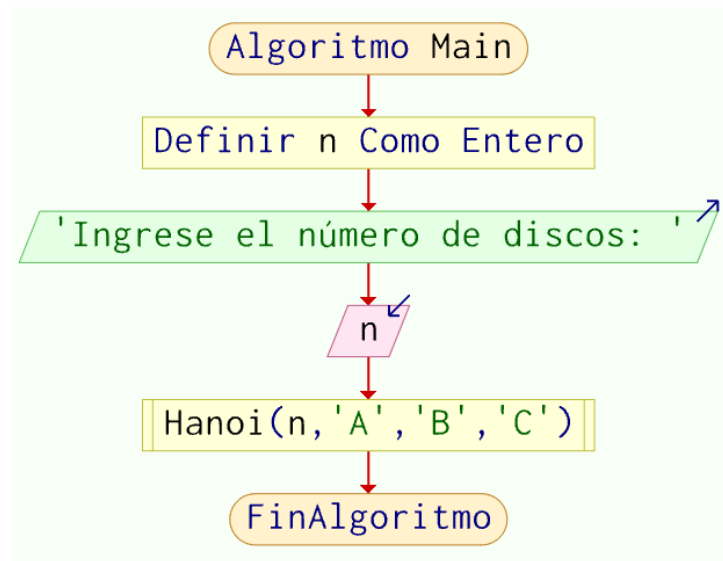
Fecha: 27/10/2024

Temas: Torres de Hanoi

Autor(es): Del Valle Ríos Luis Alberto - 751774

González Bravo Diego Ramon - 752684

Diagrama de flujo



Decisiones para diseñar el programa

Como vemos en el diagrama de flujo, deberemos tener un registro que guardara el numero de discos (s0 dicho por el profesor), pero además debido al algoritmo recursivo este deberá ir disminuyendo y aumentando conforme se llame la función por lo que usaremos a0 que es la que ira cambiando para que s0 siempre contenga el valor de los discos totales. Ya que queremos hacer la representación en el Data Segment, podemos tomar como si fuera un arreglo, por lo que la dirección de las columnas las guardaremos en s1, s2 y s3, de la misma forma por el ciclo recursivo estos valores se alteraran por lo que haremos una copia de estos que es la que cambiara en s4, s5 y s6. A diferencia del diagrama de flujo nosotros en lugar de escribir una oración tendremos que mover un disco, por lo que aparte tendremos otra función "mover", en esta será necesario localizar la columna pero también la fila para obtener la dirección del dato origen del disco y a donde queremos moverlo, ósea dirección destino, para lograr esto debemos saber que tan llena esta cada torre por lo que guardaremos también cuantos discos hay en cada torre en a1,a2 y a3. En mover tendremos que identificar que valor hay en s4 (torre origen) y en s6 (torre destino) por lo que con una implementación de un switch tenemos que encontrar cual de las torres s1,s2 o s3 esta en s4 y luego con otra cual esta en s6, ya que cuando las localicemos podremos guardar el numero de discos en cada una, guardando en a4 el valor de los discos en torre origen y en a5 los discos en torre destino. Restando s0 y a4 obtendremos la fila donde esta el disco y sumándole s4 tendremos la columna y por lo tanto la posición de donde está el disco, lo podemos guardar en t0, ocupamos guardar este valor por lo que usaremos t1 y luego haciendo lo mismo pero restando s0 menos a5 y luego 1 (ya que recordemos que la dirección empieza en cero) guardando de nuevo en t0, sumando s6 y guardando en esa dirección el dato en t1. Básicamente sería algo así mas aparte el proceso de llenado de la torre 1 donde podemos usar un algoritmo for, el guardado y obtención de datos antes y después de llamar a la función Hanói, evaluar el caso base de 1 o si s0 es menor que 1 no hacer nada, entre otras cosas que vayan fluyendo y revisando a lo largo del programa.

Simulación RARS para tres discos

Data Segment			
Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000001	0x00000000	0x00000000
0x10010020	0x00000002	0x00000000	0x00000000
0x10010040	0x00000003	0x00000000	0x00000000

Data Segment			
Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000002	0x00000000	0x00000000
0x10010040	0x00000003	0x00000000	0x00000001

Data Segment			
Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000
0x10010040	0x00000003	0x00000002	0x00000001

Data Segment			
Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000001	0x00000000
0x10010040	0x00000003	0x00000002	0x00000000

Data Segment			
Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000001	0x00000000
0x10010040	0x00000000	0x00000002	0x00000003

Data Segment			
Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000
0x10010040	0x00000001	0x00000002	0x00000003

Data Segment			
Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000002
0x10010040	0x00000001	0x00000000	0x00000003

Data Segment			
Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000000	0x00000000	0x00000001
0x10010020	0x00000000	0x00000000	0x00000002
0x10010040	0x00000000	0x00000000	0x00000003

Análisis del comportamiento del stack para el caso de 3 discos

1. Se guarda el valor de 3 discos, torre 1, torre 2, torre 3 y luego retorno a la línea 28 al zero, fin

[illegible]

2. Se guarda el valor de 2 discos, torre 1, torre 3, torre 2 y retorno a la línea 56
lw ra, 16(s0)

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7ffffefc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000002	0x10010000	0x10010008
0x7ffffef0	0x10010004	0x00400084	0x00000003	0x10010000	0x10010004	0x10010008	0x0040004c	0x00000000

3. Se recuperan los valores guardados en el punto 2 y se borran del stack

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7ffffc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe0	0x00000000	0x00000000	0x00000003	0x10010000	0x10010004	0x10010008	0x0040004c	0x00000000

4. Se guarda el valor de 2 discos, torre 1, torre 3, torre 2 y retorno a la instrucción 56 lw ra, 16(s0)

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7ffffc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000002	0x10010000	0x10010000
0x7ffffe0	0x10010004	0x00400084	0x00000003	0x10010000	0x10010004	0x10010008	0x0040004c	0x00000000

5. Se recuperan los valores guardados en el punto 4 y se borran del stack

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7fffffc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7fffffe0	0x00000000	0x00000000	0x00000003	0x10010000	0x10010004	0x10010008	0x0040004c	0x00000000

6. Se recuperan los valores guardados en el punto 1 y se borran del stack

[illegible]

7. Se guarda el valor de 3 discos, torre 1, torre 2, torre 3 y retorno a la línea 28
jal zero, fin

[illegible]

8. Se guarda el valor de 2 discos, torre 2, torre 1, torre 3 y retorno a la línea 89
lw ra, 16(sp)

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1C)
0x7ffffefc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000002	0x10010004	0x10010000
0x7ffffefe0	0x10010008	0x004000e8	0x00000003	0x10010000	0x10010004	0x10010008	0x0040004c	0x00000000
0x7ffffeff0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0040004c	0x00000000

9. Se recuperan los valores guardados en el punto 8 y se borran del stack

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7fffffc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7fffffe0	0x00000000	0x00000000	0x00000003	0x10010000	0x10010004	0x10010008	0x0040004c	0x00000000

10. Se guarda el valor de 2 discos, torre 2, torre 1, torre 3 y retorno a la línea 89 lw ra, 16(sp)

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7ffffefc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000002	0x10010004	0x10010000
0x7ffffef0	0x10010008	0x004000e8	0x00000003	0x10010000	0x10010004	0x10010008	0x0040004c	0x00000000

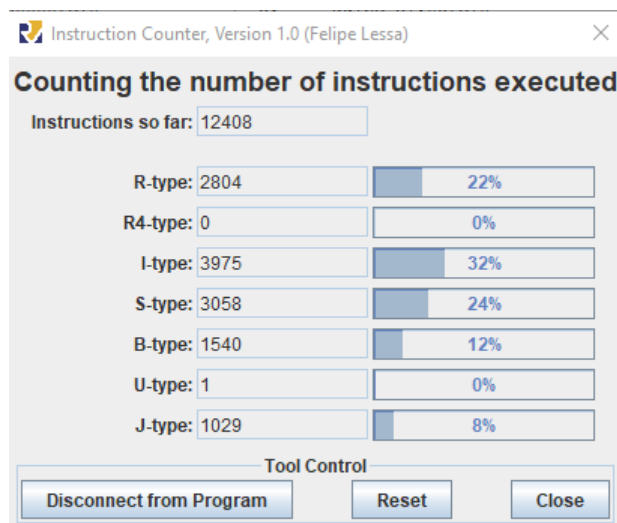
11. Se recuperan los valores guardados en el punto 10 y se borran del stack

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7ffffefc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffef0	0x00000000	0x00000000	0x00000003	0x10010000	0x10010004	0x10010008	0x0040004c	0x00000000

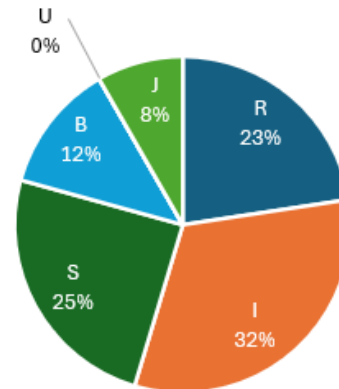
12. Se recuperan los valores guardados en el punto 7 y se borran del stack

[illegible]











Instruction count (IC) y porcentaje de instrucciones de tipo R, I y J para 8 discos



Cantidad de cada tipo de instruccion para 8 discos



Grafica de incremento del IC para las torres de Hanói en los casos de 4 a 15 discos

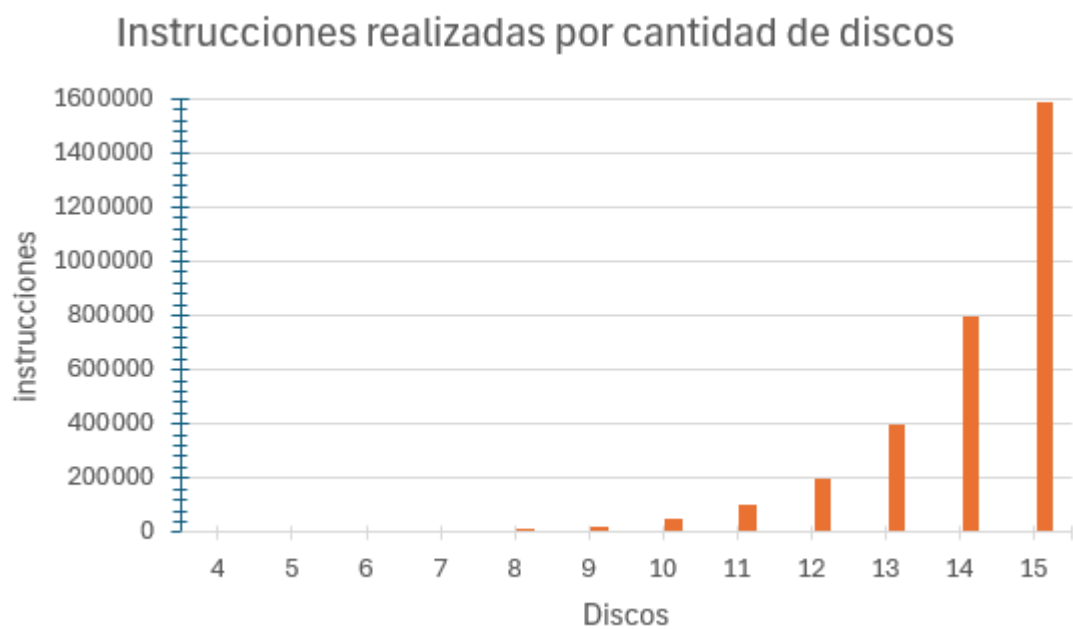
 Instruction Statistics, Version 1.0 (Ingo Kofler) ×	
Total: <input type="text" value="743"/>	4 discos
 Instruction Statistics, Version 1.0 (Ingo Kofler) ×	
Total: <input type="text" value="1525"/>	5 discos
 Instruction Statistics, Version 1.0 (Ingo Kofler) ×	
Total: <input type="text" value="3083"/>	6 discos
 Instruction Statistics, Version 1.0 (Ingo Kofler) ×	
Total: <input type="text" value="6193"/>	7 discos
 Instruction Statistics, Version 1.0 (Ingo Kofler) ×	
Total: <input type="text" value="12407"/>	8 discos
 Instruction Statistics, Version 1.0 (Ingo Kofler) ×	
Total: <input type="text" value="24829"/>	9 discos
 Instruction Statistics, Version 1.0 (Ingo Kofler) ×	
Total: <input type="text" value="49667"/>	10 discos
 Instruction Statistics, Version 1.0 (Ingo Kofler) ×	
Total: <input type="text" value="99337"/>	11 discos
 Instruction Statistics, Version 1.0 (Ingo Kofler) ×	
Total: <input type="text" value="198671"/>	12 discos
 Instruction Statistics, Version 1.0 (Ingo Kofler) ×	
Total: <input type="text" value="397333"/>	13 discos

Instruction Statistics, Version 1.0 (Ingo Kofler) X

Total: 794651 14 discos

Instruction Statistics, Version 1.0 (Ingo Kofler) X

Total: 1589281 15 discos



Conclusiones

Luis: Esta practica se me hizo muy interesante ya que pude hacer uso de todo lo que aprendí en el semestre, logre comprender el funcionamiento de varias de las instrucciones de RISC-V y como implementar la recursión haciendo uso del stack, este ultimo siendo muy importante ya que si no lo tuviéramos no tendríamos donde guardar la información, mas en este algoritmo que fue las torres de Hanoi porque entre mas discos hay mas datos se ocupan guardar. Uno de los problemas que tuvimos a la hora de hacerlo fue que no sabíamos como implementar de forma correcta la función mover, pero evaluando vimos que necesitábamos guardar el tamaño de cada torre para a la hora de mover un disco de un lugar a otro revisáramos de que torre a que torre se hará el movimiento y al mismo tiempo en otros registros guardar el tamaño de la torre origen y de la torre destino.

Diego: En esta practica se puso a prueba nuestra lógica de programación a un gran nivel ya que tuvimos que ver como implementar varias funciones con recursividad para resolver el problema de las torres, si bien al principio hubo bastantes problemas con algunas funciones como la de mover, supimos identificar el problema y resolverlo.

Creo que esta práctica fue interesante ya que el saber sobre cómo funciona el ensamblador te da una perspectiva diferente a la que siempre vemos en la programación habitual.

Liga de Git Hub: https://github.com/LuisMamey/Luis_Practica1_OyA