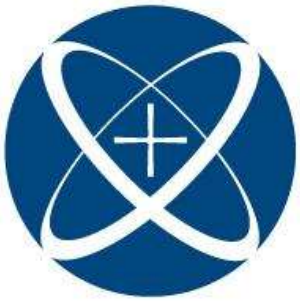
INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE OCCIDENTE

ARQUITECTURA COMPUTACIONAL



ITESO, Universidad Jesuita de Guadalajara

NOMBRE DEL DOCENTE

JUANPABLO IBARRA ESPARZA

INTEGRANTES:

Paulina Valadez Fernández 728412

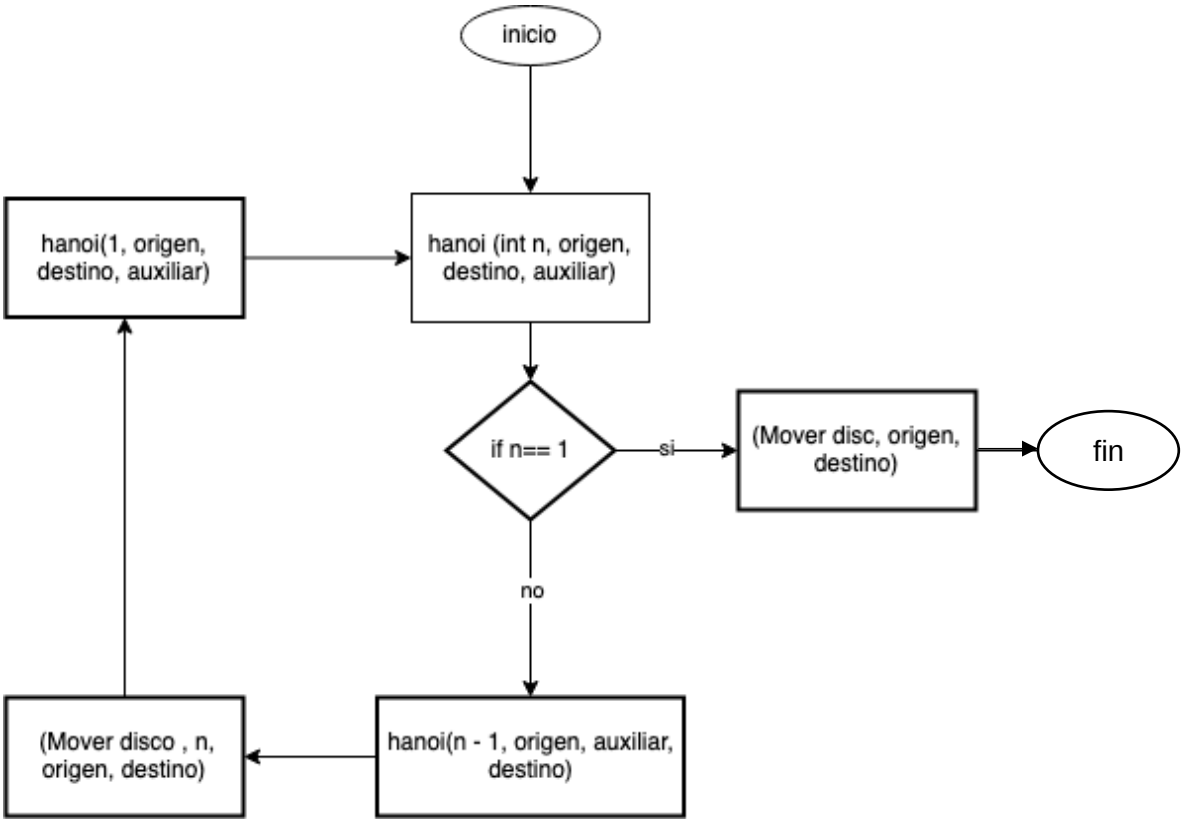
Perla Marina García García 728149

Práctica 1

25 de octubre de 2023

Repositorio: https://github.com/Pavito15/arquiComputacional/tree/practica1

1. **Diagrama de flujo del programa implementado realizado en Visio o programa equivalente.**



2. **Las decisiones que se tomaron al diseñar su programa se pueden tomar como referencia el diagrama de flujo del programa.**

   Primero recordar lo que son las torres de Hanoi y cómo se realizaban los pasos, después se tomó en cuenta encontrar un programa en C y entender el funcionamiento de que realizaba dicho programa, el siguiente código fue nuestra base:

```c
#include <stdio.h>
#include <unistd.h>

void hanoi(int n, char origen, char destino, char auxiliar) {
    if (n == 1) {
        printf("Mover disco 1 desde %c a %c\n", origen, destino);
        return;
```

```
    }
    hanoi(n - 1, origen, auxiliar, destino);

    printf("Mover disco %d desde %c a %c\n", n, origen, destino);
    hanoi(1, origen, destino, auxiliar);
    hanoi(n - 1, auxiliar, destino, origen);
}

int main() {
    int num_discos;
    printf("Ingresa el número de discos: ");
    scanf("%d", &num_discos);
    printf("\nLos pasos a seguir son:\n");
    hanoi(num_discos, 'A', 'C', 'B');
    printf("Esperando 20 segundos antes de salir...\n");
    sleep(60);
    return 0;
}
```

También identificamos cuales eran los pasos en cada llamada recursiva:

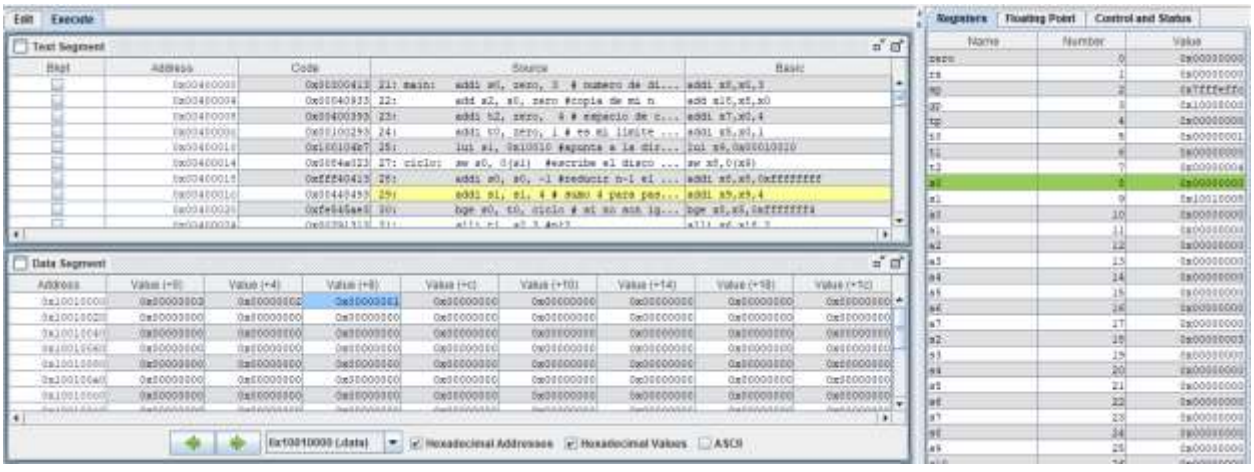
1.- Mover parte superior a NO Destino

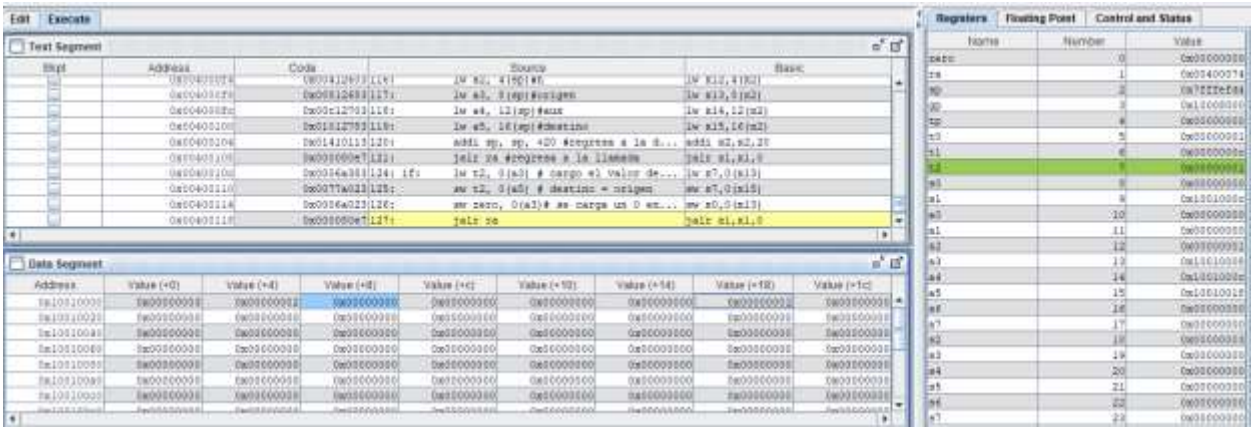
1.1 Ajustar apuntadores (Reducir Origen y Aumentar Destino)

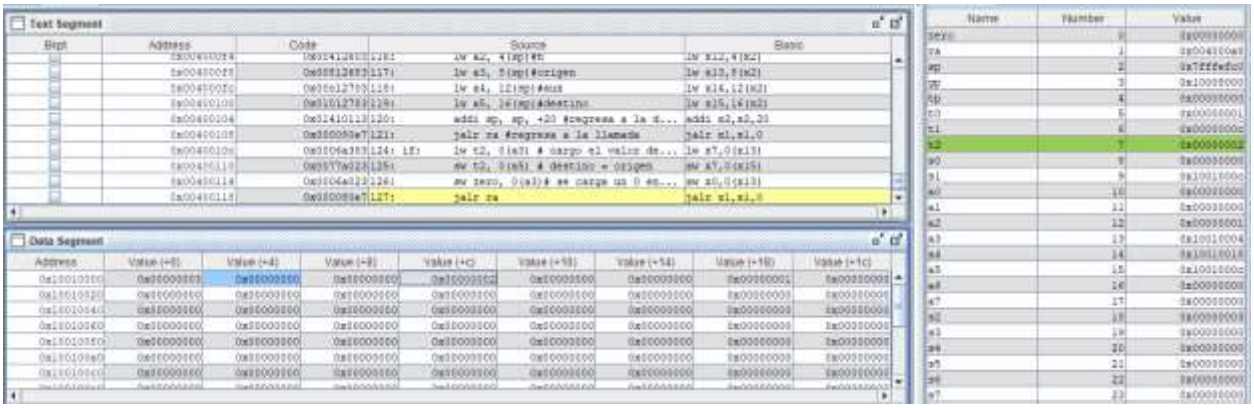
2.- Mover pieza actual a Destino

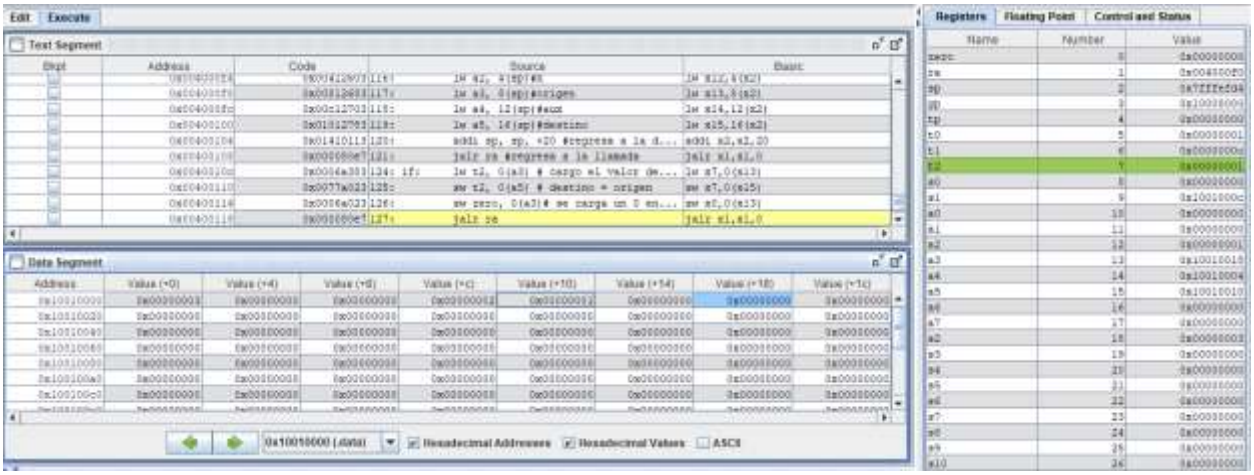
3.- Mover parte superior a Destino

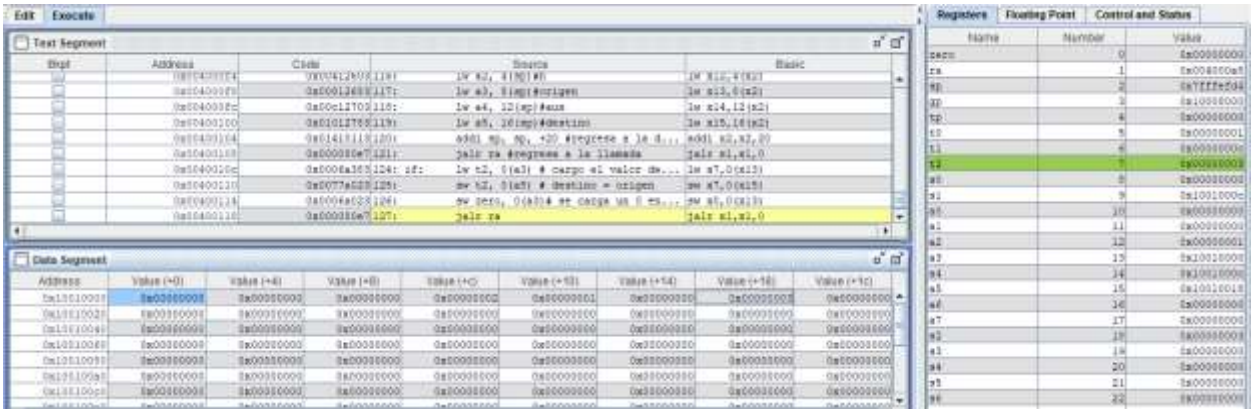
3. **Una simulación en el RARS para 3 discos (La simulación son impresiones de pantalla de data segment del RARS).**

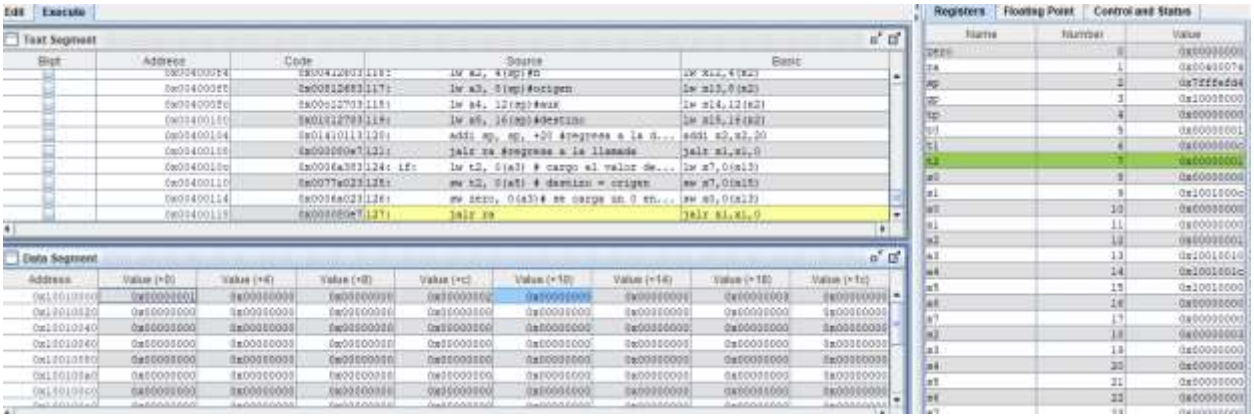


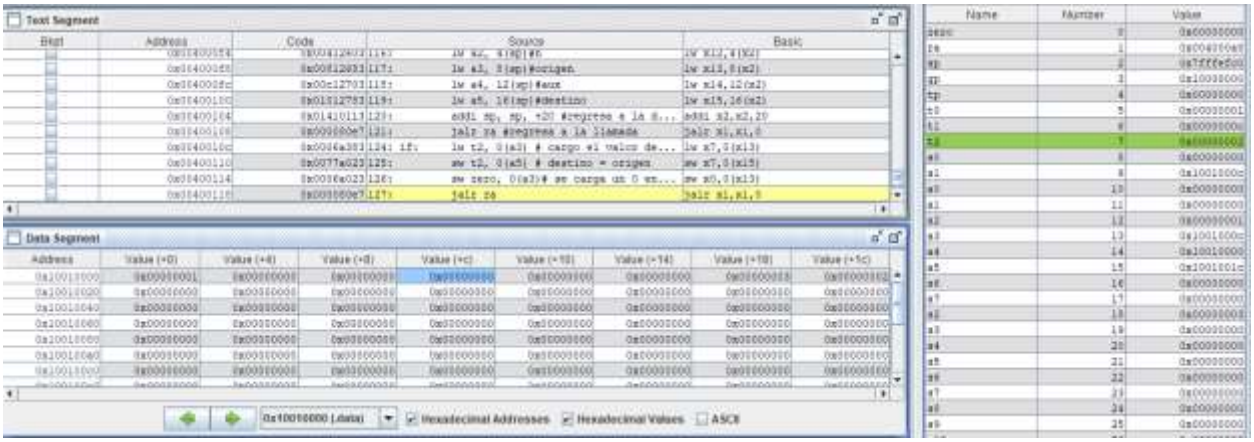


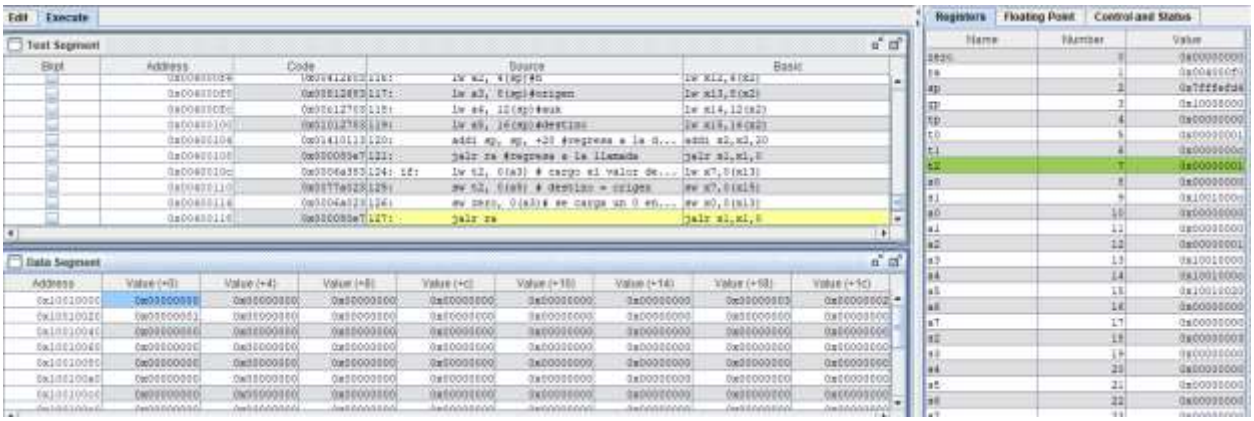












4. Análisis del comportamiento del stack para el caso de 3 discos.

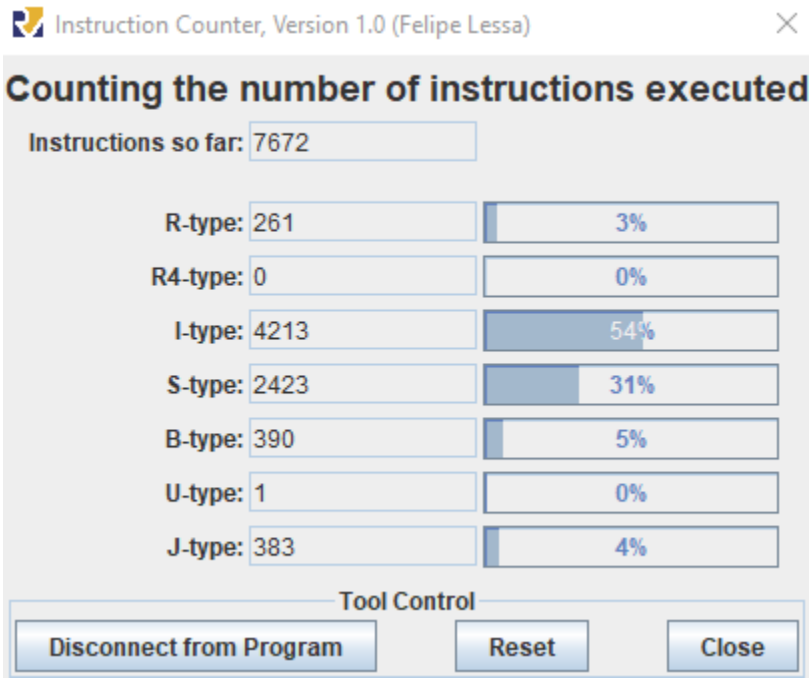
Primero modificamos la dirección de memoria en donde comenzará el stack pointer, en este caso será desde la dirección 0x7fffefe8, realizamos el push escribiendo la dirección del destino *0x10010018* (‘A’) en la dirección más alta para ello aplicamos un offset de 16, después en la dirección anterior escribimos la dirección del auxiliar *0x1001000c* (‘B’) con un offset de 12, luego la dirección del origen *0x10010000* (‘C’) su offset es de 8, después se escribe el valor de n en ese momento que es 3, su offset es de 4, finalmente en la dirección con un offset 0 se guarda el return address *0x00400040* la cual sería la dirección a donde regresaría cuando termine la llamada. Esto lo hace cada vez que entra a una llamada recursiva sin embargo los valores cambian dependiendo del disco en el que se encuentra, ya que puede cambiar el origen, el auxiliar y el destino para cada llamada recursiva.

5. **Incluir en el instruction count (IC) y especificar el porcentaje de instrucciones de tipo R, I y J para 8 discos.**

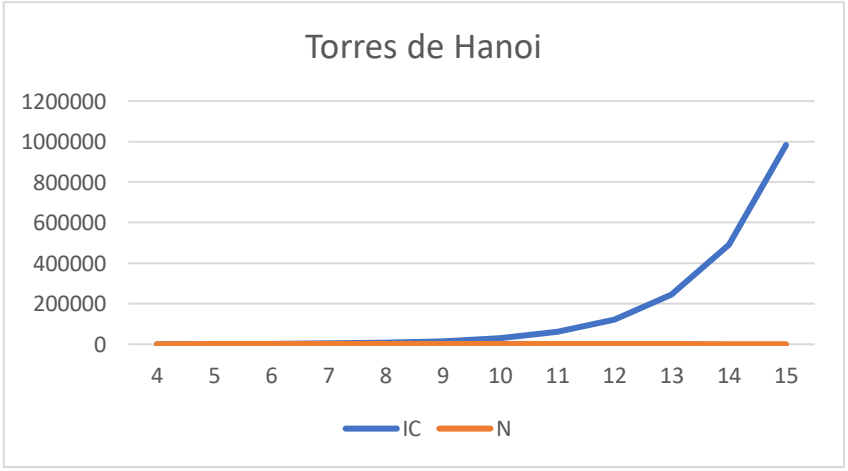
R -> 3%

I -> 54%

J ->4%



6. **Una gráfica que muestre cómo se incrementa el IC para las torres de Hanoi en los casos de 4 a 15 discos.**



N	IC
4	456
5	940
6	1904
7	3828
8	7672
9	15356
10	30720
11	61444
12	122888
13	245772
14	491536
15	983060

7. Conclusiones de cada integrante del equipo.

Perla: Al realizar esta práctica puede entender más el comportamiento del stack pointer para realizar las llamadas recursivas, de lo cual nunca me había preocupado ya que ensamblador necesita todas las instrucciones necesarias para escribir directamente en la memoria. Fue muy necesario comprender los pasos que realizaba las llamadas para poder traducirlas, ya que dependiendo del disco en el que se encontraba el origen, destino y auxiliar cambian para poder cumplir la regla de no poner un disco más grande encima de uno mas pequeño, y al modificar los argumentos estos deben de tener las nuevas direcciones de memoria que se habían guardado en el stack pointer.

Pau: Para esta práctica aprendí cómo se implementa el famoso jueguito de torres de hanoi utilizando llamadas recursivas y la pila que es nuestro stack. La recursión nos ayudó bastante para poder dividir el problema en subproblemas más pequeños y resolverlos de manera eficiente.