

Universidad Autónoma de Baja California  
Facultad de Ciencias Químicas e Ingeniería



Alumno: Eduardo Marcelo Gutiérrez Soto  
Profesora: Evangelina Lara Camacho  
Practica 9 Procedimientos recursivos

**Teoría:**

**Explique paso a paso el algoritmo de Euclides usando como ejemplo dos números enteros diferentes a cero de su elección.**

El algoritmo de Euclides es un método que nos sirve para poder calcular lo que es el MCD Máximo Común Divisor de dos números para ello necesitamos seguir una serie de pasos que nos servirá para aplicar dicho algoritmo, consta de una serie de divisiones sucesivas para poder encontrar el MCD mediante la implementación del algoritmo haremos la prueba con los números 22 y 14.

**Vamos a calcular el Máximo común divisor de los números 22 y 14.**

Primero empezamos el 22 entre el 14 y tendremos como cociente el 1 y residuo 8, entonces pasaremos nuestro residuo como el próximo número a dividir y lo dividiremos con el número anterior que es el 14 nos da como resultado un cociente de 1 y un residuo de 6, ahora dividiremos el 8 entre el 6 nos da como residuo un 2 y como cociente un 1 y volvemos a hacer el mismo procedimiento dividiendo el 6 con el 2 y tendremos como resultado un cociente de 3 y un residuo de 0, cuando nuestro residuo sea igual a cero el algoritmo termina ya que se encontró el valor del MCD y el resultado sería el valor del residuo anterior que fue 2.

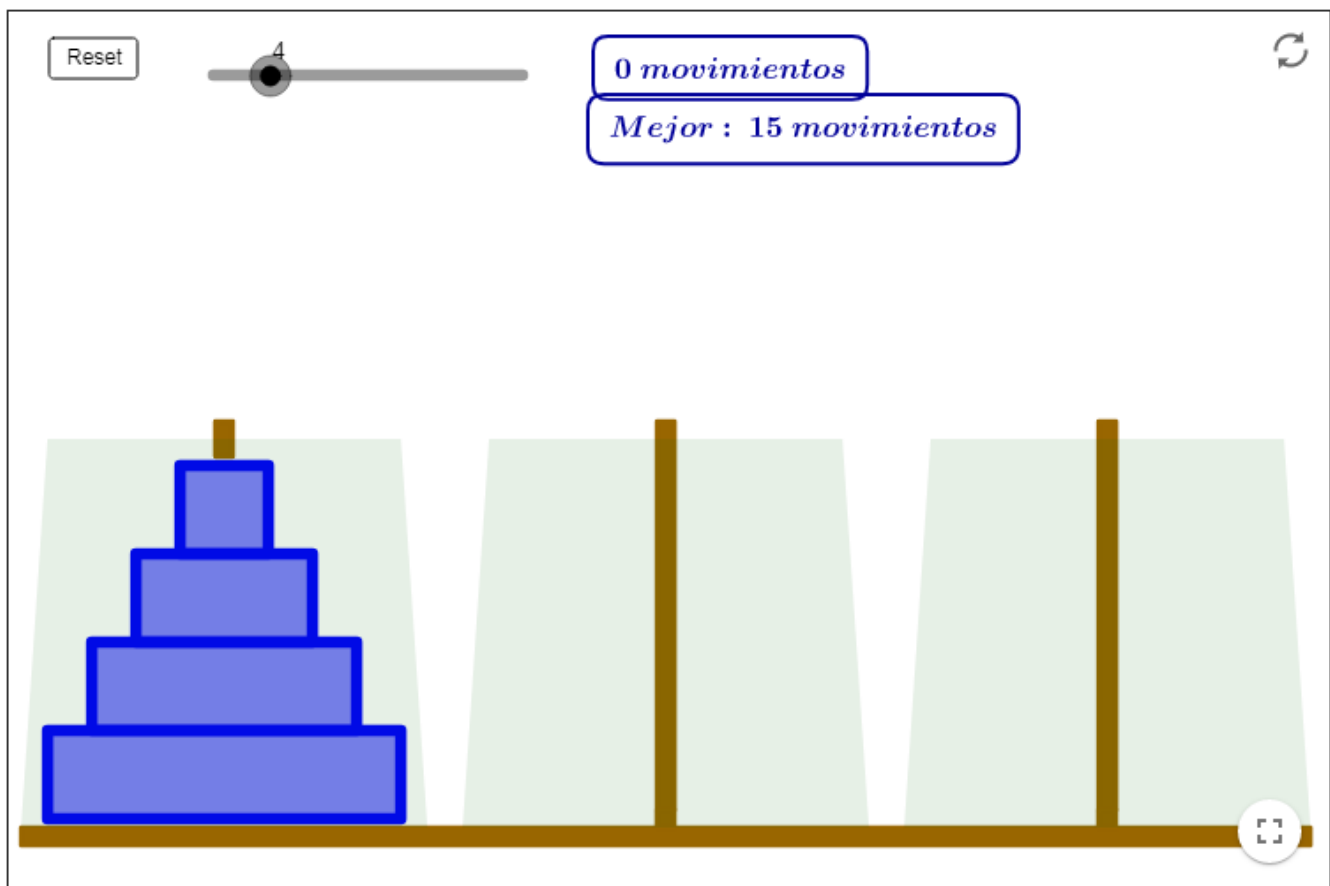
Cocientes	1	1	1	3
22	14	8	6	2
Residuos	8	6	2	0

**Describe brevemente en qué consiste el acertijo matemático Torres de Hanoi y muestre un ejemplo paso a paso de su resolución para una cantidad de discos mayor a 3.**

Es un juego que consiste en el uso de 1 torre con una cierta cantidad de discos en este caso para el ejemplo usaremos 4 discos.

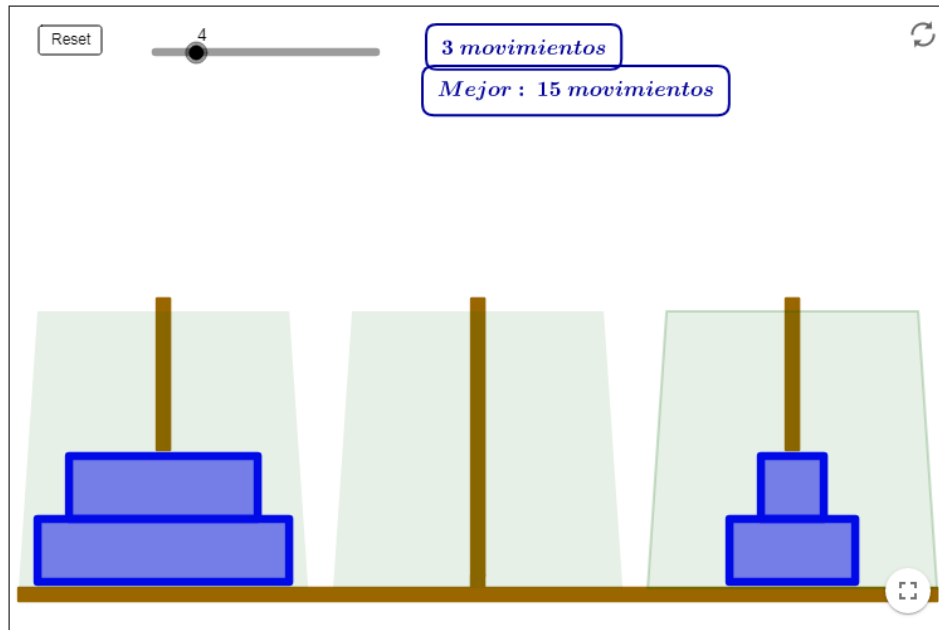
Las reglas son muy fáciles constan en una torre 4 discos de los cuales con una serie de movimientos hacia dos torres que se encuentran vacías se van a mover los discos, una de las reglas más importantes es que un disco grande no puede estar arriba de un pequeño, al inicio del juego tienes una torre con todos los discos organizados en forma ascendente y tiene que terminar de la misma forma en la torre 3 con una serie de combinaciones. Se necesita de una serie de combinaciones para poder hacer un movimiento de una columna hacia otra, ahora veremos un ejemplo de ello.

Este simulador del juego nos muestra que se necesita de una cantidad de 15 movimientos para poder completar el juego de las torres de hanoi haremos la prueba para ver movimiento a movimiento la resolución del juego.



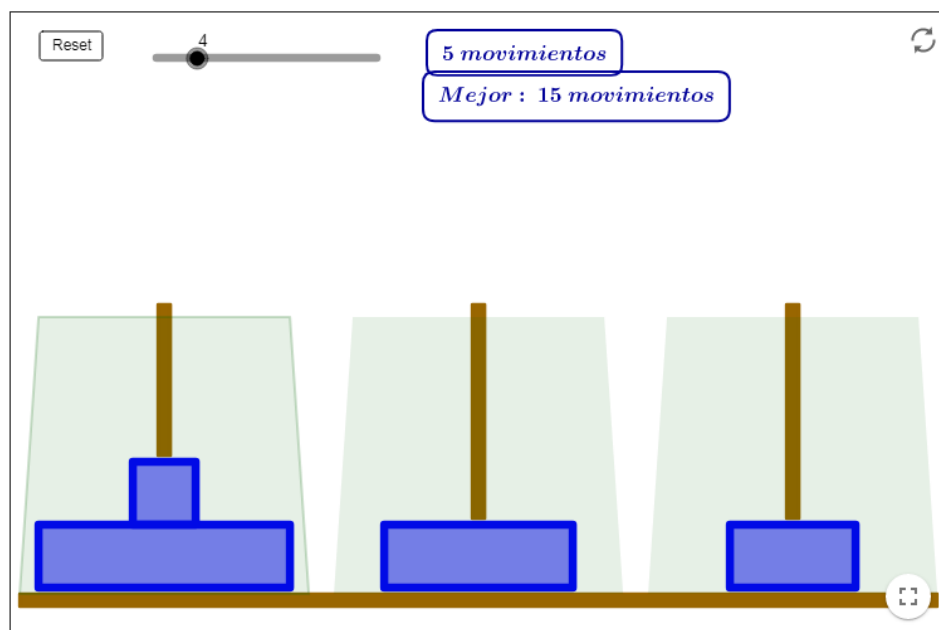
### Movimiento 1.

En mi primer movimiento realice 3 movimientos para poder tener en el otro extremo las piezas respetando el orden de tener la mas quena siempre arriba de una grande, primero moví la mas pequeña al centro y después la que le seguía a la tercer pila y luego moví la mas pequeña a la tercer pila y por ello quedo asi y ya en eso tenemos realizados 3 movimientos.



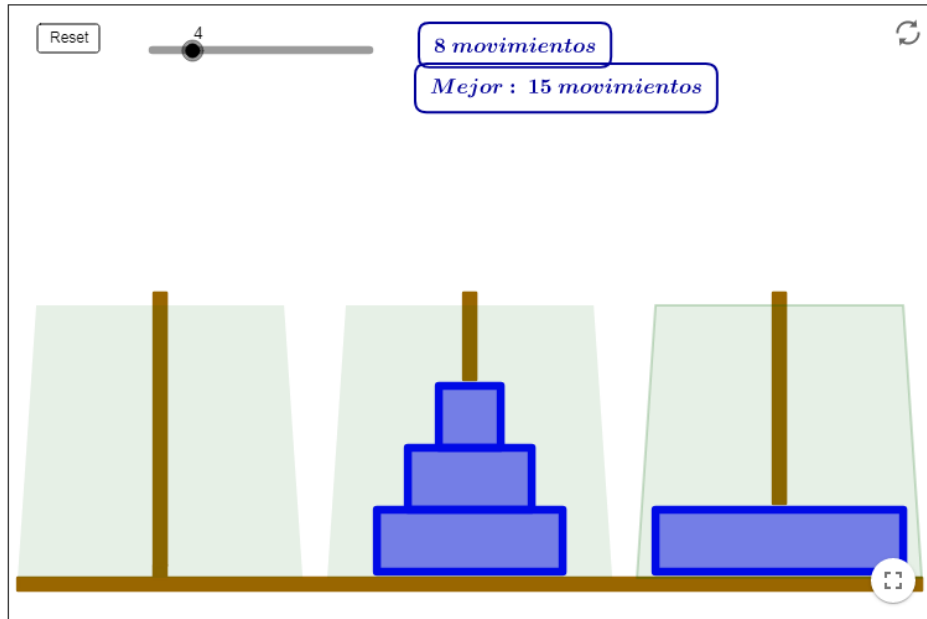
### Movimiento 2.

Ahora ya tenemos 5 movimiento lo que se realizo fue mover hacia el centro la pieza de arriba de la pila 1 y movimos la pieza mas pequeña de la pila 3 a la pila uno para poder seguir moviendo las piezas respetando la regla de que una pieza grande no puede estar arriba de una pequeña.



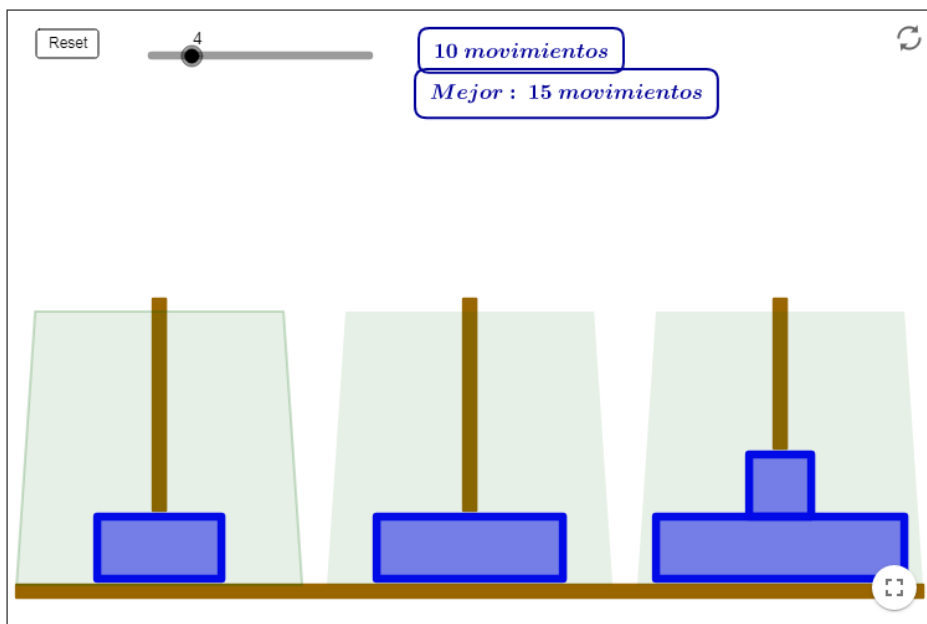
### Movimiento 3.

Ahora ya realizamos 8 movimientos y movimos la pieza mas grande de la pila uno a la tercer pila y en la pila del centro tenemos las demás piezas respetando el orden que hemos hablado de que siempre la pieza pequeña esta arriba de una mas grande.



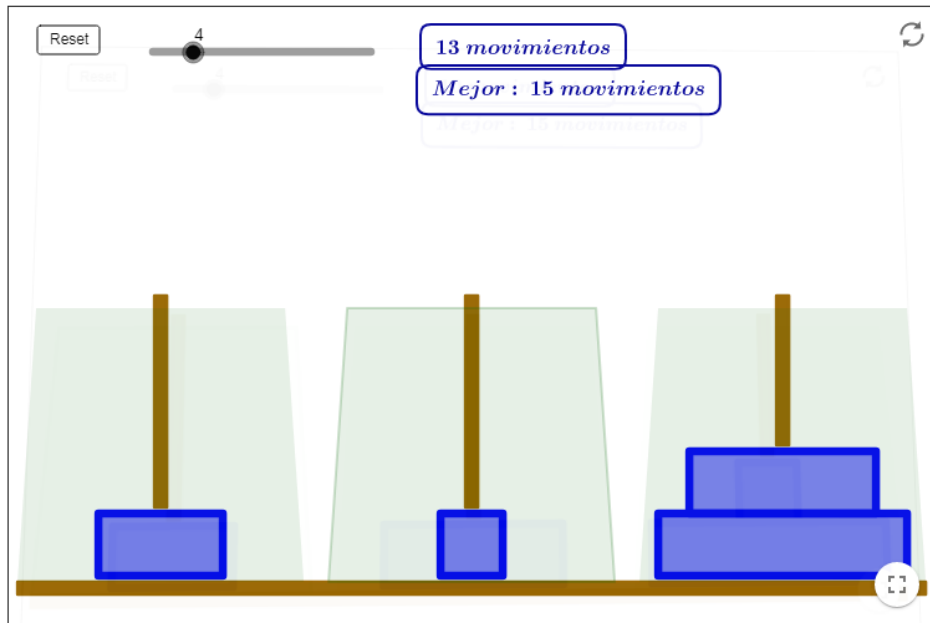
### Movimiento 4.

Realizamos 2 movimientos mas para abrir paso a la pieza mas grande de la pila 2 moviendo hacia los extremos cada una de las piezas que tenia arriba de ella.



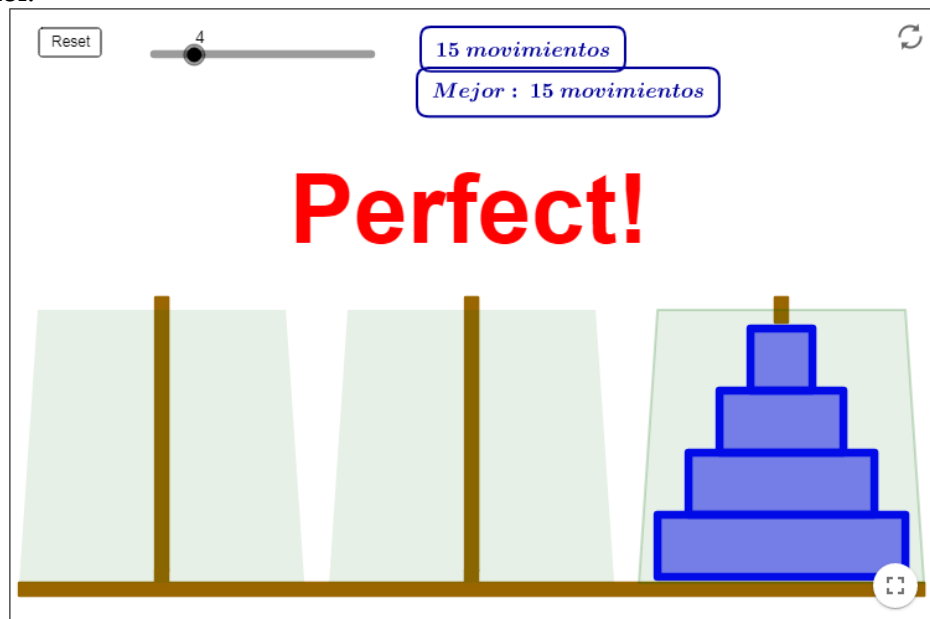
### Movimiento 5.

Movimos la pieza mas pequeña de la pila 3 hacia la pila 1 para poder mover la pieza de la pila dos hacia la pila 3, después movimos la pieza pequeña de la pila 1 a la pila 2. en eso realizamos 3 movimientos.



### Movimiento 6.

Ahora movimos las demás piezas en su orden respectivo, y terminaos en 15 movimientos el juego de las torres de haoi.



## Desarrollo

Programe el procedimiento recursivo mcd el cual recibe dos números en AL y BL y retorna en AL su máximo común divisor obtenido por medio del algoritmo de Euclides.

## CÓDIGO:

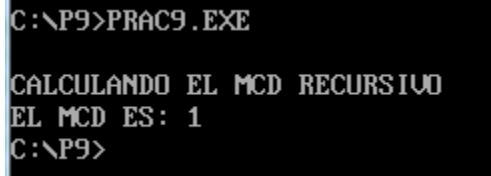
```
MCD PROC
;PROGRAMA RECURSIVO DEL MCD EL CUAL RECIBE DOS NUMEROS
;EN AL Y BL Y RETORNA EN AL SU MAXIMO COMUN DIVISOR POR EL
;ALGORITMO DE EUCLIDES.
DIV BL      ;AL=AX/BL
MOV AL,BL   ;MOVEMOS EL VALOR DE BL COMO DIVIDENDO
CMP AH,0    ;COMPARAMOS EL RESIDUO CON 0
JE @@FIN
MOV BL,AH   ;MOVEMOS EL VALOR DEL RESIDUO A BL PARA HACER DIVISIONES CON EL
MOV AH,0
CALL MCD    ;LLAMAMOS EL PROC PARA HACER LA RECURSION
@@FIN:
MOV AL,BL   ;SI EL RESIDUO ES CERO ENTONCES MANDAMOS EL RESIDUO ANTERIOR
ADD AX,30H  ;SE AJUSTA A SU VALOR VERDADERO

RET
ENDP
```

## VALORES DE PRUEBA

```
MOV AL,99    ;DATO 1
MOV BL,50    ;DATO 2
CALL MCD
CALL PUTCHAR;IMPRIMIMOS EL DATO DE AL
```

## EJECUCIÓN:



```
C:\P9>PRAC9.EXE

CALCULANDO EL MCD RECURSIVO
EL MCD ES: 1
C:\P9>
```

Se realizó el MCD de los números AL=99 y BL=50 el cual nos dio como resultado un 1, este valor fue verificado por el método manual y si es el resultado correcto y también por una calculadora Online de MCD.

Programe el procedimiento recursivo printBinRec el cual despliega en pantalla el valor de AL en formato binario.

## CÓDIGO:

```
MOV AL,0DAH
MOV DX,OFFSET MENS_3
CALL PUTS
CALL printBinRec
;*****

printBinRec PROC
;DESPLIEGA EN PANTALLA EL VALOR DE AL EN FORMATO BINARIO

MOV AH,AL
CMP AH,0 ;CASO BASE DONDE EL REG BL SEA 0
JNE @@NXT
MOV AL,'0'
CALL PUTCHAR;PARA PODER IMPRIMIR EL ULTIMO CERO
JMP @@FIN
@@NXT:
MOV AL,0 ;INICIALIZAMOS AL EN 0
SHL AH,1 ;DEZPLAZAMOS UN BIT A LA IZQUIERDA
ADC AL,30H ;SUMAMOS CON ACARREO EN EL REGISTRO AL
CALL PUTCHAR
MOV AL,AH
CALL printBinRec
@@FIN:
RET
ENDP
```

## EJECUCIÓN:

```
EL NUMERO EN BINARIO ES: 11011010
C:\P9>_
```



Programe el procedimiento recursivo TorresDeHanoi el cual despliega en pantalla los pasos a seguir para resolver el acertijo matemático que lleva su nombre. Recibe en AL la cantidad de discos a mover, en BL el nombre del pilar origen, en CL el nombre del del pilar auxiliar y en DL el nombre del pilar destino. Los nombres en BL, CL y DL corresponden a un solo carácter ASCII, por ejemplo: 'A', 'B' y 'C' respectivamente.

### CÓDIGO:

Así mandamos a llamar la función con sus respectivos parámetros que recibe.

```
MOV DX,OFFSET MENS_7
CALL PUTS
XOR AX,AX
XOR BX,BX
XOR CX,CX
XOR DX,DX ;LIMPIAMOS LOS REGISTROS
MOV AL,3
MOV BL,'A'
MOV CL,'B'
MOV DL,'C'
CALL TorresDeHanoi
```

y la función es.

El caso base es

```
TorresDeHanoi PROC
PUSH AX
PUSH BX
PUSH DX
PUSH CX ;SALVAMOS REGISTROS

CMP AL,1 ;HACEMOS EL CASO BASE EL CUAL SERIA SOLO TENER 1 DISCO
JNE @@HANOI_E
PUSH DX ;HACEMOS OTRA COPIA DEL DATO DE DX
MOV DX,OFFSET mENS_4 ;MENSAJE DE DISCO
CALL PUTS
ADD AL,30H ;PARA HACER VALOR IMPRIMIBLE DEL NUMERO
CALL PUTCHAR
MOV DX,OFFSET mENS_5 ;CARACTER DE :
CALL PUTS
MOV AL,BL
CALL PUTCHAR
MOV DX,OFFSET mENS_6 ;MENSAJE DE CARACTER ->
CALL PUTS
POP DX
MOV DH,0
MOV AL,DL
CALL PUTCHAR
JMP @@FIN
```

y la parte recursiva es.

```
;PARTE RECURSIVA SALIMOS DEL CASO BASE
@@HANOI_E:
XCHG CL,DL ;INTERCAMBIAMOS VALORES
PUSH DX ;GUARDAMOS EL VALOR EN LA PILA
MOV DX,OFFSET mENS_4
CALL PUTS ;IMPRIMIMOS EL MENSAJE DE DISCO
ADD AL,30H ;AJUSTAMOS EL VALOR NUMERICO
CALL PUTCHAR ;IMPRIMIMOS EL NUMERO
MOV DX,OFFSET mENS_5 ;IMPRIMIMOS EL CARACTER :
CALL PUTS
PUSH AX ;SALVAMOS EL VALOR DE AL
MOV AL,BL ;MOVEMOS EL CARACTER DEL ORIGEN QUE ESTA EN BL
CALL PUTCHAR
MOV DX,OFFSET mENS_6 ;IMPRIMIMOS EL ->
CALL PUTS
POP AX ;RECUPERAMOS AX
MOV AH,0 ;PARA NO TENER BASURA EN LA PARTE ALTA
POP DX ;RECUPERAMOS EL VALOR DE DX
MOV DH,0 ;PARA NO TENER BASURA EN LA PARTE ALTA
PUSH AX ;GUARDAMOS OTRA VEZ EL VALOR DEL N-1
MOV AL,DL ;MOVEMOS EL VALOR DEL DESTINO
CALL PUTCHAR
POP AX
MOV AH,0 ;PARA NO TENER BASURA EN LA PARTE ALTA
SUB AL,30H ;RESTAMOS LA DIFERENCIA DEL AJUSTE
DEC AL ;DECREMENTAMOS AL OTRA VEZ
XCHG CL,DL
XCHG BL,CL
CALL TorresDeHanoi
```

## EJECUCIÓN.

```
JUEGO DE TORRES DE HANOI
DISCO 3 : A -> B
DISCO 2 : B -> A
DISCO 1 : A -> C
C:\P9>
```

para hacer este ejercicio primero lo hice en lenguaje c el cual si me funciono perfectamente con su recursivo ya que si lo quería hacer desde un principio en ensamblador se me hacia algo difícil pero al final lo hice por que lo que hice fue traducir mi propio código de c a ensamblador pero al momento de hacerlo no se por que me dio tantos problemas por que nomas me imprimia la primer parte de la secuencia de hanoi. El código que hice en c fue el siguiente.

```
#include<stdio.h>

void hanoi(int n, int origen, int destino, int aux);

int main(){
    int n;

    printf("Digite el numero de discos para LA TORRE DE HANOI: ");
    scanf("%d",&n);

    hanoi(n,1,3,2);
}

void hanoi(int n, int origen, int destino, int aux){
    if(n == 1){
        printf("[%d]-->[%d]\n",origen, destino);
    } else
    {
        hanoi(n-1,origen,aux, destino);
        printf("[%d]-->[%d]\n",origen, destino);
        hanoi(n-1,aux,destino,origen);
    }
}
```

pero no se que fue lo que contemple mal que no me quiso funcionar bien, y esa fue mi única dificultad en el ejercicio de las torres de hanoi, no se si fue la manera en como lo interprete pero no me salio del todo bien el ejercicio.

## **CONCLUSIÓN**

La practica estuvo muy interesante el como realizar en ensamblador lo que son los procesos recursivos una manera muy fácil de como solucionar una serie de problemas de una manera mas fácil, no todo se puede resolver con la recursividad pero las cosas que podemos resolver con ese método nos hacen las cosas de una manera mas fácil solo hay que contemplar siempre el posible caso base que vayamos a tener ya que con eso es lo que nos vamos a guiar para poder hacer la funcion recursiva.

## **DIFICULTADES**

Mi única dificultad fue el juego de las torres de hanoi ya que no pude del todo realizarlo bien ya que no pude interpretar de la manera correcta el código que hice en c pasarlo a ensamblador, por que todos los ejercicios lo hice de esa manera primero los hago en c y ya luego los paso a ensamblador para poder plasmar la lógica de una manera mas facil.