Karachi Campus

Discovering Knowledge

Department Of Software Engineering

Computer Architecture and Organization (LAB)

COURSE: CEN-221 TERM: FALL 2020,
CLASS: BSE- 3(B)

Submitted By:

Rabia Amjad	**02-131192-016**
Rida Zahoor Siddiqui	**02-131192-053**
Rabbia Ilyas	**02-131192-029**

Submitted to:

Sir Muhammad Rehan Baig

Date Of Submission: 22-01-21

Tower Of Hanoi

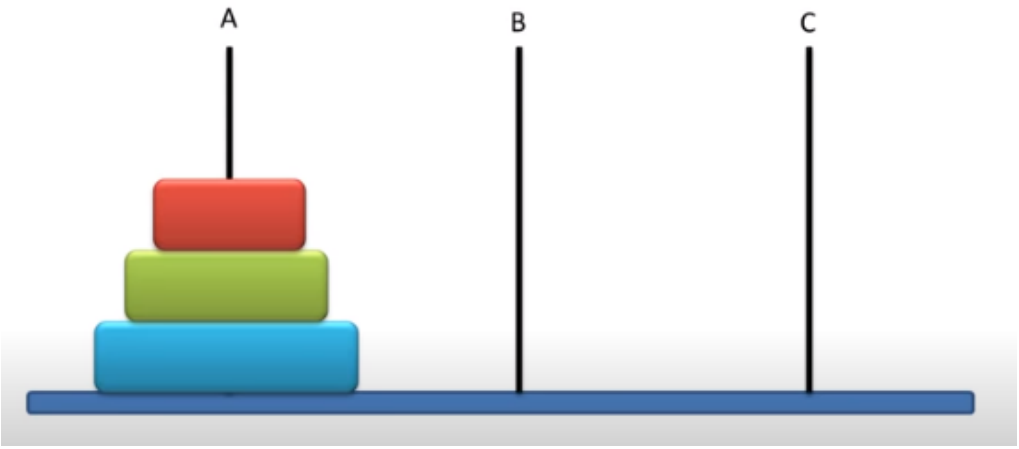
Objective: The object of making this project is to take number of discs from users and show that how tower of Hanoi concept works.

Rules Of Tower Of Hanoi:

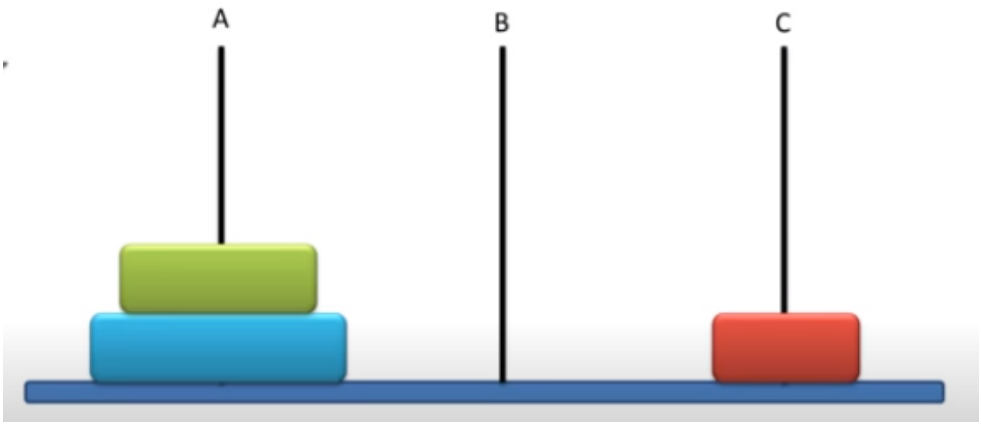
1. Only one disc can move at a time.
2. No larger disc can move appear on a smaller disc.
3. Only one top most disc can be moved from one tower and it can appear only at the top most position of another tower.
4. Transfer all the disks in right most rod in ascending order and fewest possible.

How Tower Of Hanoi Works?

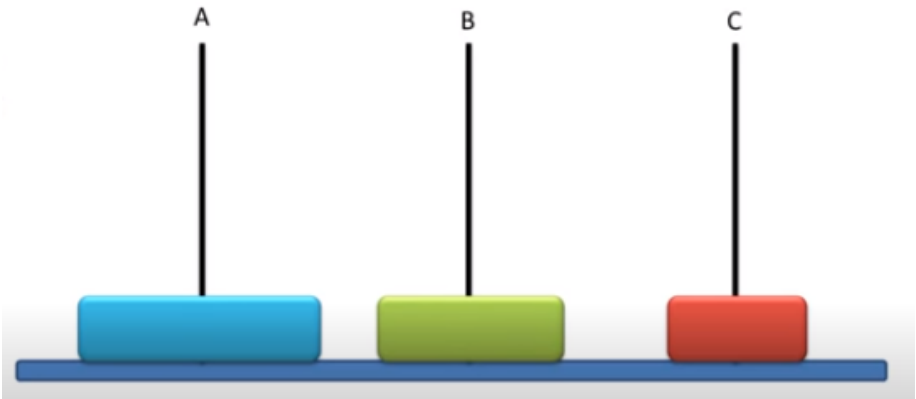
Tower of Hanoi is a puzzle of mathematics where we have three rods and n discs. The purpose of the puzzle is to move the entire stack to another rod by following above rules. We can understand this simply by using 3 discs. The number of steps of moving stack from one rod to another can be calculated by formula '2^n-1' (where n=total number of discs), so as we have 3 discs then total number of steps will be 7.



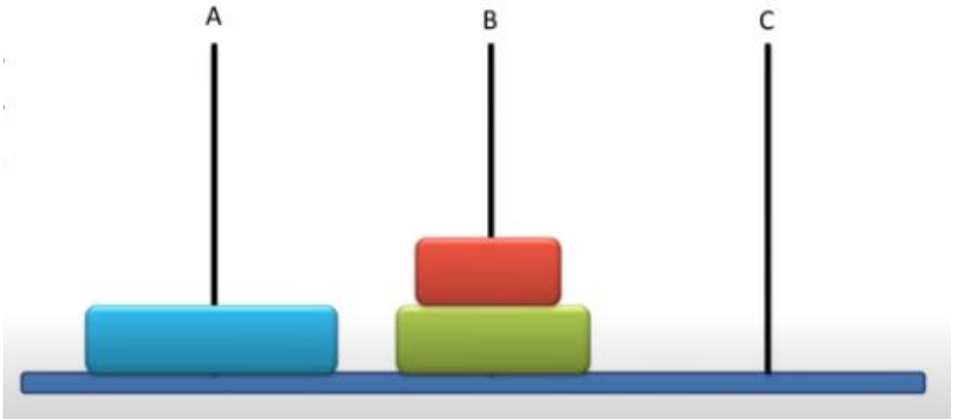
1) **Step 01:** Move top most disc to rod ‘C’.



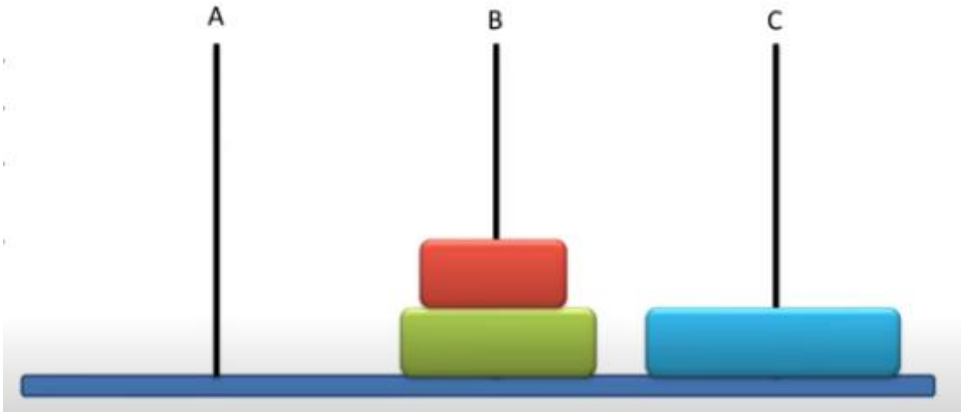
2) **Step 02:** Move second disc to rod ‘B’



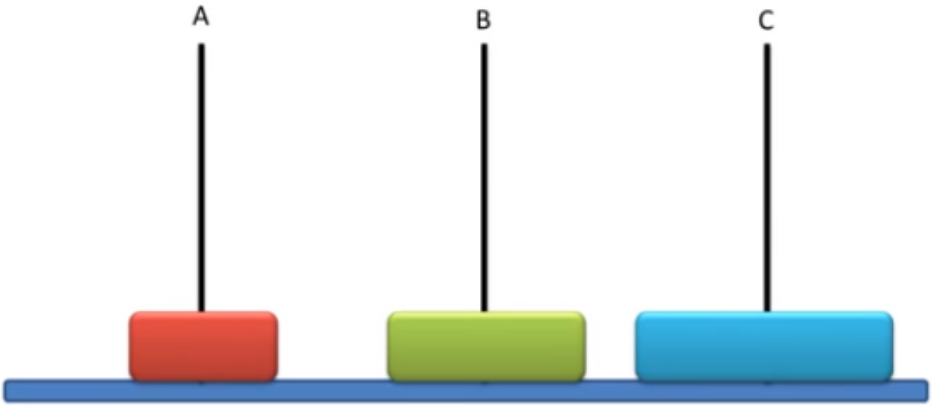
3) **Step 03:** move disc 1 to rod ‘B’



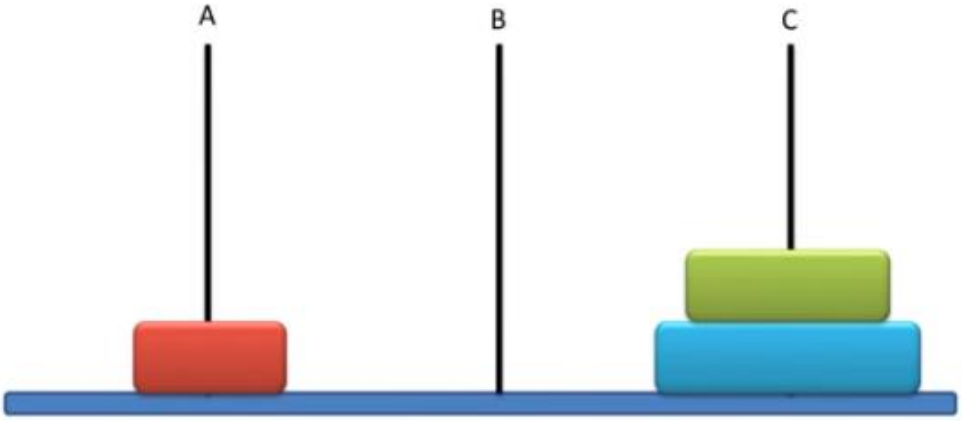
4) **Step 04:** Move disc 3 to rod ‘C’.



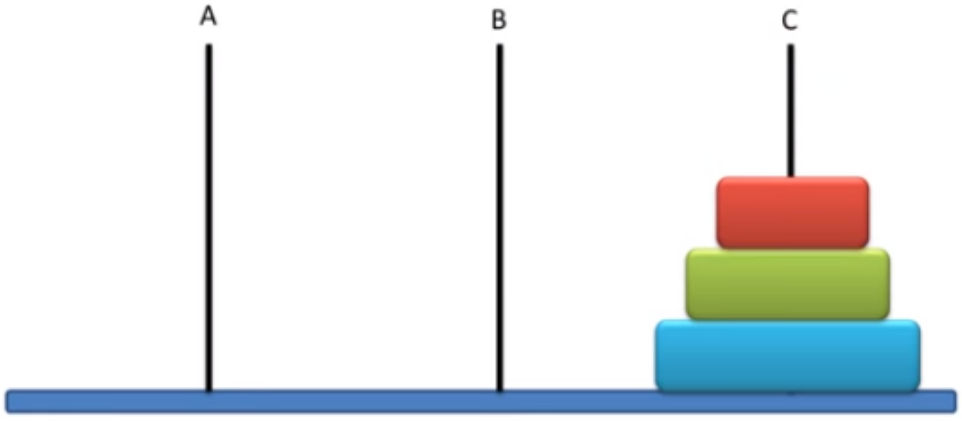
5) **Step 05:** Move disc 1 to rod ‘B’.



6) **Step 06:** Move disc 2 to rod ‘C’.



7) **Step 07:** Move disc 1 to rod ‘C’.



How This Program Works: In this program user will be asked to enter the number of disc they want to move from one tower to another. Once the number of discs are entered, the program will check whether the number of disc is zero or not, if entered number is zero than program will terminate and **"Have a good day"** message will be printed otherwise it will execute next step. After performing every step successfully, the program will terminate and output will be shown. All this work is done with the help of loop,stack, recursion and branching.

Algorithm:

```
#################################################################
#CAO Project
#Implementation of Tower of Hanoi using MIPS
#Tower of Hanoi consist of three towers and n number of disk
#Rules of game
#Rule-1: One disc can be moved at one time
#Rule-2: Smaller disk cannot be placed below larger disk
#Number of moves can be obtained by formula 2*moves(a-1)+1
#################################################################


#.data section
#Initialization/Declaration of variables


.data
d:.asciiz "\n====================================="
TOH:.asciiz "\n|          TOWER OF HANOI          |"
d1:.asciiz "\n====================================="
prompt:.asciiz "\n\nEnter number of disk:"
Part1:.asciiz "\nMove Disk from tower "
Part2:.asciiz" to "
```

```
Part3:.asciiz "tower "
newline:.asciiz "\n"
bye:.asciiz "\nHave a good day!"

#.text section
#Coding of main program

.text
.globl main

main:

#Print string message
li $v0,4
la $a0,d
syscall

#Print string message
li $v0,4
la $a0,TOH
syscall

#Print string message
li $v0,4
la $a0,d1
syscall

#Print string message
```

```
li $v0,4
la $a0,prompt
syscall

#Read integer value
li $v0,5
syscall

#Branch to label bye if $v0 <= zero
ble $v0,0,Bye

#move the value of $v0 to $s0
move $s0,$v0

li $s1,1    #Load immidiate $s1 with 1(i.e for Tower 1)
li $s2,2    #Load immidiate $s1 with 2(i.e for Tower 2)
li $s3,3    #Load immidiate $s1 with 3(i.e for Tower 3)

#Jump and link label Tower of Hanoi
jal Tower_of_Hanoi

#Unconditional branch to label exit
b exit

#Tower of Hanoi label
Tower_of_Hanoi:

#branch to label loop if value of #s0 is not equals to 1
```

```
bne $s0,1,loop

#print string message
li $v0,4
la $a0,Part1
syscall

#print int value
li $v0,1
add $a0,$s1,$zero
syscall

#print string message
li $v0,4
la $a0,Part2
syscall

#print string message
li $v0,4
la $a0,Part3
syscall

#print int value
li $v0,1
add $a0,$s3,$zero
syscall

#print string message
```

```
li $v0,4
la $a0,newline
syscall

#jumpt to register ra
jr $ra

#loop label
loop:

#Initial value of stack
addiu $sp,$sp,-20

#Storing in stack
sw $ra, 16($sp)     #Store return register
sw $s3, 12($sp)     #Store tower 3
sw $s2, 8($sp)      #Store tower 2
sw $s1, 4($sp)      #Store tower 1
sw $s0, 0($sp)      #Store number of disks

#Recursive call 1
#Swapping of tower 2 and 3
move $t0,$s2
move $s2,$s3
move $s3,$t0

addiu $s0,$s0,-1
```

```
jal Tower_of_Hanoi

lw $ra,16($sp)     #Load return address
lw $s3,12($sp)     #Load tower 3
lw $s2,8($sp)      #Load tower 2
lw $s1,4($sp)      #Load tower 1
lw $s0,0($sp)      #Load number of disks

#print string message
li $v0,4
la $a0,Part1
syscall

#print int value
li $v0,1
add $a0,$s1,$zero
syscall

#print string message
li $v0,4
la $a0,Part2
syscall

#print string message
li $v0,4
la $a0,Part3
syscall
```

```
#print int value
li $v0,1
add $a0,$s3,$zero
syscall

#print string message
li $v0,4
la $a0,newline
syscall

#Recursive call 2
#Swapping of tower 2 and 1
move $t0,$s2
move $s2,$s1
move $s1,$t0

addiu $s0,$s0,-1

jal Tower_of_Hanoi

lw $ra,16($sp)

addiu $sp,$sp,20

jr $ra

#bye label
Bye:
```

```
#print string message
li $v0,4
la $a0,bye
syscall

#unconditional branch to exit label
b exit

#exit label
exit:
li $v0,10
syscall
```

Output:

```
=======================================
|           TOWER OF HANOI            |
=======================================

Enter number of disk:3

Move Disk from tower 1 to tower 3

Move Disk from tower 1 to tower 2

Move Disk from tower 3 to tower 2

Move Disk from tower 1 to tower 3

Move Disk from tower 2 to tower 1

Move Disk from tower 2 to tower 3

Move Disk from tower 1 to tower 3

-- program is finished running --
```

```
=======================================
|           TOWER OF HANOI            |
=======================================

Enter number of disk:0

Have a good day!
-- program is finished running --
```