```
# Title: Project 1 Part III             Filename: Project 1 Part III.s
# Author: Dan Cassidy                   Date: 2015-03-03
# Description: This program will take the given number of disks and
#   solve the Towers of Hanoi puzzle using a recursive function.
# Input: The number of disks to use in the puzzle.
# Output: The steps taken to solve the puzzle.
# Variables:
#   main: $a0 = n
#   hanoi: $a0 = n, $a1 = start, $a2 = finish, $a3 = extra
#       *note: start, finish, and extra move around some
################## Data segment #####################
.data
getDisks:   .asciiz     "Enter number of disks>"
moveDisk:   .asciiz     "Move disk "
fromPeg:    .asciiz     " from peg "
toPeg:      .asciiz     " to peg "
endLine:    .asciiz     ".\n"

################## Code segment #####################
.text
.globl  main
main:                                   #main program entry
        li      $v0, 4                  #prepare to output getDisks
        la      $a0, getDisks           #set output to getDisks
        syscall                         #output getDisks

        li      $v0, 5                  #prepare to input n
        syscall                         #input n

        addi    $a0, $v0, 0             #load n into argument 1 of hanoi
        addi    $a1, $zero, 1           #load start into argument 2 of hanoi
        addi    $a2, $zero, 2           #load finish into argument 3 of hanoi
        addi    $a3, $zero, 3           #load extra into argument 4 of hanoi

        jal     hanoi                   #call hanoi

        li      $v0, 10                 #prepare to exit program
        syscall                         #exit program



# Function: hanoi
# Description: Given inputs, will solve 'towers of hanoi' recursively
# Input:
#   $a0, holds the disk number
#   $a1, holds the number designator of the starting peg
#   $a2, holds the number designator of the final peg
#   $a3, holds the number designator of the extra peg
# Output:
#   A single step in the process of solving the puzzle.
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
hanoi:                                  #function entry
        beq     $a0, $zero, end_h       #if (n==0), jump to end_h
        addi    $sp, $sp, -20           #make room in stack
```

```
        sw      $ra, 16($sp)            #save $ra to stack
        sw      $a0, 12($sp)            #save n to stack
        sw      $a1, 8($sp)             #save start to stack
        sw      $a2, 4($sp)             #save final to stack
        sw      $a3, 0($sp)             #save extra to stack

        addi    $a0, $a0, -1            #decrement n (load n-1 into argument 1)
        addi    $t3, $a2, 0             #copy finish to a temp location
        addi    $a2, $a3, 0             #load extra into argument 3
        addi    $a3, $t3, 0             #load finish into argument 4
        jal     hanoi                   #hanoi(n-1, start, extra, finish)
        lw      $s0, 12($sp)            #restore n from stack
        lw      $a1, 0($sp)             #restore extra from stack into argument 2
        lw      $a2, 4($sp)             #restore finish from stack into argument 3
        lw      $a3, 8($sp)             #restore start from stack into argument 4
        addi    $sp, $sp, 16            #move $sp back but keep $ra saved

        li      $v0, 4                  #prepare to output moveDisk
        la      $a0, moveDisk           #set output to moveDisk
        syscall                         #output moveDisk
        li      $v0, 1                  #prepare to output n
        addi    $a0, $s0, 0             #set output to n
        syscall                         #output n
        li      $v0, 4                  #prepare to output fromPeg
        la      $a0, fromPeg            #set output to fromPeg
        syscall                         #output fromPeg
        li      $v0, 1                  #prepare to output start
        addi    $a0, $a3, 0             #set output to start
        syscall                         #output start
        li      $v0, 4                  #prepare to output toPeg
        la      $a0, toPeg              #set output to toPeg
        syscall                         #output toPeg
        li      $v0, 1                  #prepare to output final
        addi    $a0, $a2, 0             #set output to final
        syscall                         #output final
        li      $v0, 4                  #prepare to output endLine
        la      $a0, endLine            #set output to endLine
        syscall                         #output endLine

        addi    $a0, $s0, -1            #load n-1 into argument 1
        jal     hanoi                   #hanoi(n-1, extra, finish, start)
        lw      $ra, 0($sp)             #restore $ra from stack
        addi    $sp, $sp, 4             #move $sp; all memory reclaimed
end_h:  jr  $ra                         #return from function
```