Mini-Programming Project Solution

```
# mini-project part one
#-----------------------------------------------------------------------------
#      MIPS assembly for a simple adding machine
#-----------------------------------------------------------------------------
# Title:       adding machine
# Filename:    part1.s
# Author:      Liqiang Zhang
# Date:        Jan. 2015
# Description: a simple adding machine program that repeatedly reads in
#              integers and adds them into a running sum.
# Input:       a series of integers
# Output:      the sum of the integers
#-----------------------------------------------------------------------------
#


################## data segment ######################
       .data
       .globl  inputmsg
       .globl  outputmsg
inputmsg:      .asciiz "\nPlease enter a number (0 to stop):\n"
outputmsg:     .asciiz "\nThe sum of the numbers is: "


################## code segment ######################
.text
.globl main
main:                                       #main has to be a global label
               addu    $s7, $0, $ra         #save the return address in a global register

               li      $s1, 0               #we use s1 to hold the sum, initialize it to be zero here.

Loop:
               li      $v0, 4               #print_str (system call 4)
               la      $a0, inputmsg        #takes the address of string as an argument
               syscall

               li      $v0, 5               #read_int (system call 5)
               syscall

               beq     $v0, $0, Exit
               add     $s1, $s1, $v0
               j       Loop

Exit:
               li      $v0, 4               #print_str (system call 4)
               la      $a0, outputmsg       #takes the address of string as an argument
               syscall

               li      $v0, 1               #print_int (system call 1)
               add     $a0, $0, $s1
               syscall

                                            #Usual stuff at the end of the main
               addu    $ra, $0, $s7         #restore the return address
               jr      $ra                  #return to the main program
               add     $0, $0, $0           #nop
```

```
# mini-project part two
#------------------------------------------------------------------------
#       MIPS assembly for printing first 100 prime numbers
#------------------------------------------------------------------------
# Title:       Printing prime numbers
# Filename:    part2.s
# Author:      Liqiang Zhang
# Date:        Jan. 2015
# Description: the program finds and prints first 100 prime numbers.
# Input:       None
# Output:      the first 100 prime numbers
#------------------------------------------------------------------------
#
#
##include <iostream>
#using namespace std;
#
#bool test_prime(int);
#
#int main()
#{
#       int i = 2;
#       int n = 0;
#       while (n < 100)
#       {
#               if (test_prime(i))
#               {
#                       cout << i << " is a prime" << endl;
#                       n++;
#               }
#               i++;
#       }
#
#       return 0;
#}
#
#bool test_prime(int n)
#{
#       for (int factor = 2; factor < n; factor++)
#               if ((n % factor)==0) return false;
#
#       return true;
#}


################## data segment ######################
        .data
        .globl  outputmsg
outputmsg:      .asciiz " is a prime\n"



################## code segment ######################
        .text
        .globl main
main:                                       #main has to be a global label
                addu    $s7, $0, $ra        #save the return address in a global register

                li $s1, 2                   #start with the first prime number 2
                li $s2, 0                   #to count the number of prime numbers that we print out
                li $s3, 100                 #the total number of prime numbers that we want to print out

Loop1:
                add $a0, $zero, $s1         #call function test_prime
                jal     test_prime

                addi $s1, $s1, 1            #move to the next number

                beq $v0, $zero, CheckLoop1  #if n is not a prime, skip the printout

                addi $s2, $s2, 1
                li $v0, 1                   #print out "n "
                syscall

                li $v0, 4                   #print out " is a prime\n"
                la $a0, outputmsg
                syscall

CheckLoop1:
```

```
                slt $s4, $s2, $s3             #check if s2 is less than 100
                bne $s4, $zero, Loop1         #if s2 < 100, go back to Loop1

                                              #Usual stuff at the end of the main
                addu    $ra, $0, $s7          #restore the return address
                jr      $ra                   #return to the main program
                add     $0, $0, $0            #nop

        .globl  test_prime
test_prime:
                li $t0, 2
                add $t1, $a0, $zero
                li $v0, 0

Loop2:          slt $t3, $t0, $t1
                beq $t3, $zero, Exit

                div $a0, $t0
                mfhi $t2

                beq $t2, $zero, NotPrime
                addi $t0, $t0, 1
                j Loop2

Exit:
                addi $v0, $v0, 1
NotPrime:
                jr $ra
```

```
# mini-project part three
#------------------------------------------------------------------------
#       MIPS assembly for solving the Towers of Hanoi Puzzle
#------------------------------------------------------------------------
# Title:        Towers of the Hanoi Puzzle
# Filename:     Hanoi.s
# Author:       Liqiang Zhang
# Date:         Jan. 2015
# Description: the assembly program that solves the Towers of Hanoi puzzle
# Input:        the number of disks
# Output:       the sequence of disk movement that solves the puzzle.
#------------------------------------------------------------------------
#
# The c program for solving the Towers of Hanoi Puzzle
#
# /* move n smallest disks from start to finish using extra */
# void hanoi(int n, int start, int finish, int extra)
# {
#       if(n != 0){
#               hanoi(n-1, start, extra, finish);
#               print_string(“Move disk”);
#               print_int(n);
#               print_string(“from peg”);
#               print_int(start);
#               print_string(“to peg”);
#               print_int(finish);
#               print_string(“.\n”);
#               hanoi(n-1, extra, finish, start);
#       }
# }
#
# main()
# {
#       int n;
#       print_string(“Enter number of disks>“);
#       n = read_int();
#       hanoi(n, 1, 2, 3);
#       return 0;
#}
#
#------------------------------------------------------------------------

################## data segment ######################
        .data
        .globl inputmsg
inputmsg:       .asciiz         "\nWelcome to the solver of Towers of Hanoi!\nPLease enter number of disks> "
        .globl movemsg
movemsg:        .asciiz         "\nMove disk "
        .globl frommsg
frommsg:        .asciiz         " from peg "
        .globl tomsg
tomsg:          .asciiz         " to peg "
        .globl endlinemsg
endlinemsg:     .asciiz         ".\n"
        .globl finishmsg
finishmsg:      .asciiz         "\nThe job is done!\n"

################## code segment ######################
.text
.globl main
main:
        addu    $s7, $0, $ra    # save the return adderss in a global register

        li      $v0, 4          # pring_string (system call 4)
        la      $a0, inputmsg   # take the address of the string as argument
        syscall
        li      $v0, 5
        syscall                 # read input to $v0
        add     $a0, $0, $v0    # copy the input (n) to #a0
        addi    $a1, $0, 1      # start = 1
        addi    $a2, $0, 2      # finish = 2
        addi    $a3, $0, 3      # extra = 3
        jal     hanoi

        li      $v0, 4
        la      $a0, finishmsg # print finish message
        syscall

        add     $v0, $0, $0
```

```
        addu    $ra, $0, $s7
        jr      $ra
        add     $0, $0, $0

        .globl hanoi
hanoi:
        addi    $sp, $sp, -24
        sw      $ra, 20($sp)
        sw      $s0, 16($sp)
        sw      $a0, 12($sp)
        sw      $a1, 8($sp)
        sw      $a2, 4($sp)
        sw      $a3, 0($sp)

        beq     $a0, $0, end   # if $a0 == 0, go to end

        add     $s0, $a0, $0   # copy $a0 to $s0 (the value of n)

        addi    $a0, $a0, -1   # $a0 = n - 1
        add     $t0, $a2, $0
        add     $a2, $a3, $0   # $a2 = extra
        add     $a3, $t0, $0   # $a3 = finish

        jal     hanoi


        li      $v0, 4
        la      $a0, movemsg
        syscall                # print_string("Move disk ")

        li      $v0, 1
        add     $a0, $s0, $0
        syscall                # print_int(n)

        li      $v0, 4
        la      $a0, frommsg
        syscall                # print_string("from peg ")

        li      $v0, 1
        add     $a0, $a1, $0
        syscall                # print_int(start)

        li      $v0, 4
        la      $a0, tomsg
        syscall                # print_string(" to peg ")

        li      $v0, 1
        add     $a0, $a3, $0
        syscall                # print_int(finsh)

        li      $v0, 4
        la      $a0, endlinemsg
        syscall                # print_string(".\n")

        addi    $a0, $s0, -1   # $a0 = n - 1
        add     $t0, $a1, $0
        add     $a1, $a2, $0   # $a1 = extra
        add     $a2, $a3, $0   # $a2 = finish
        add     $a3, $t0, $0   # $a3 = start

        jal     hanoi

end:
        lw      $a3, 0($sp)
        lw      $a2, 4($sp)
        lw      $a1, 8($sp)
        lw      $a0, 12($sp)
        lw      $s0, 16($sp)
        lw      $ra, 20($sp)
        addi    $sp, $sp, 24
        jr       $ra           # return
```