Group ID - 65

Member #1 : Gautam Choudhary [15114027]

Member #2 : Sandeep Pal [15114063]

B. Tech. in Comp. Sci & Enginn.

IIT, Roorkee

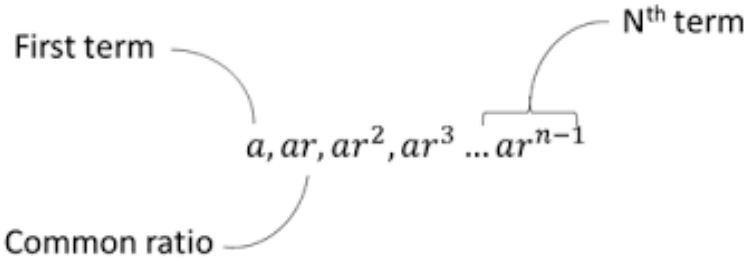
Project : Sum of 'N' consecutive terms of a GP with given 'F' and 'R'.

Submitted to : Dr. Sudip Roy

CSN-221: Computer Architecture and Microprocessors

Department of Computer Science and Engineering

Indian Institute of Technology Roorkee



A geometric progression is represented as above in most general form.

Problem Number : 9

Problem Statement : Calculate the sum of N consecutive terms of a geometric progression (GP) series with first term F and common ratio R. It should read the values "N", "F" and "R" from the user.

About Project

The project contains a

- gp_sum.c file - The file contains the c code for summation of 'n' terms of a GP.
- gp_sum.s file - The file contains the assembly code of gp_sum.c file.
- gp_sum.asm file - The file contains the assembly code.

Basic Idea

The *first term* of geometric progression is **F** and *common ratio* is **R**. Any 'k' th term of the geometric progression can be represented as-

$$T(K) = F * (R^{K-1})$$

Initialize sum to 0 and set temp variable to F.

In each iteration add temp to sum and multiply R to temp variable to generate next term of the geometric progression.

The sum obtained at the end of 'N' iterations is the sum of given GP.

C code (gp_sum.c)

This is the C code for generating the sum of 'N' terms of a GP -

```c
// GroupID-65 (15114063_15114027) - Sandeep Pal & Gautam Choudhary
// Date: November 2, 2016
// gp_sum.c The program takes input the first term, common ratio and
//            number of terms of GP and prints the SUM of GP
#include <stdio.h>

int main(){
   // Defining the Variables
    /** n - Number of terms
    ** f - First term of GP
    ** r - Common Ratio of GP
    */
    int f, r, n;

   // Taking input from the User (in order)
    printf("Enter 'First' term of GP, f:   ");
    scanf("%d", &f);

    printf("Enter 'Common Ratio' of GP, r: ");
    scanf("%d", &r);

    printf("Enter the number of terms, n:  ");
    scanf("%d", &n);


   // Temporary Variables
    int term = f;
    int sum = 0;
    int i;

   // Algorithm to calculate SUM of N terms of a GP
    for(i=0; i<n; i++){
        sum = sum + term;
        term = term*r ;
    }

   // Print the OUTPUT i.e. SUM
    printf("The sum of GP is: %d\n", sum);


    return 0;
}
```

Assembly Language Program (gp_sum.s file)

Using the command " ***gcc -S gp_sum.c*** " in command prompt terminal, we got the following assembly language program file as ***gp_sum.s*** –

```
    .file   "gp_sum.c"
    .section    .rodata
    .align 8
.LC0:
    .string "Enter 'First' term of GP, f:   "
.LC1:
    .string "%d"
    .align 8
.LC2:
    .string "Enter 'Common Ratio' of GP, r: "
    .align 8
.LC3:
    .string "Enter the number of terms, n:  "
.LC4:
    .string "The sum of GP is: %d\n"
    .text
    .globl  main
    .type   main, @function
main:
.LFB0:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    subq    $32, %rsp
    movl    $.LC0, %edi
    movl    $0, %eax
    call    printf
    leaq    -24(%rbp), %rax
    movq    %rax, %rsi
    movl    $.LC1, %edi
    movl    $0, %eax
    call    __isoc99_scanf
```

```
36      movl    $.LC2, %edi
37      movl    $0, %eax
38      call    printf
39      leaq    -20(%rbp), %rax
40      movq    %rax, %rsi
41      movl    $.LC1, %edi
42      movl    $0, %eax
43      call    __isoc99_scanf
44      movl    $.LC3, %edi
45      movl    $0, %eax
46      call    printf
47      leaq    -16(%rbp), %rax
48      movq    %rax, %rsi
49      movl    $.LC1, %edi
50      movl    $0, %eax
51      call    __isoc99_scanf
52      movl    -24(%rbp), %eax
53      movl    %eax, -12(%rbp)
54      movl    $0, -8(%rbp)
55      movl    $0, -4(%rbp)
56      jmp .L2
57  .L3:
58      movl    -12(%rbp), %eax
59      addl    %eax, -8(%rbp)
60      movl    -20(%rbp), %eax
61      movl    -12(%rbp), %edx
62      imull   %edx, %eax
63      movl    %eax, -12(%rbp)
64      addl    $1, -4(%rbp)
65  .L2:
66      movl    -16(%rbp), %eax
67      cmpl    %eax, -4(%rbp)
68      jl  .L3
69      movl    -8(%rbp), %eax
70      movl    %eax, %esi
71      movl    $.LC4, %edi
72      movl    $0, %eax
73      call    printf
74      movl    $0, %eax
75      leave
76      .cfi_def_cfa 7, 8
77      ret
78      .cfi_endproc
79  .LFE0:
80      .size   main, .-main
81      .ident  "GCC: (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4"
82      .section    .note.GNU-stack,"",@progbits
83
```

The Assembly Language Program (.asm file)

- This is a description of the file showing the registers and variables used in program.

```
# GroupID-65 (15114063_15114027) - Sandeep Pal & Gautam Choudhary
# Date: November 2, 2016
# gp_sum.asm The program takes input the first term, common ratio and number of terms of GP
#            and prints the SUM of GP
#
# Registers used:
# $a0 - syscall parameter.
# $v0 - syscall paramaeter and return value.
# $t0 - used to hold current 'term' of GP
# $t1 - used to hold current 'sum' of terms of GP
# $t2 - used to hold value of counter variable 'i'
# $t3 - used to hold number of terms of GP
# $t4 - used as hold Common Ratio of GP
#
# Memory Variables used:
# f   - used to store First term of GP
# r   - used to store Common Ratio of GP
# n   - used to store Number of terms of GP
# sum - used to store SUM of terms of GP

```

- **'.data'** contains all the variable declarations**. '.asciiz'** is a kind of datatype for *string* and **'.word'** is kind of datatype of *integers* with size 4bytes i.e. 32bits.

```
.data
    ask_input_str1: .asciiz "Enter 'First' term of GP, f:   "
    ask_input_str2: .asciiz "Enter 'Common Ratio' of GP, r: "
    ask_input_str3: .asciiz "Enter the number of terms, n:  "
    tell_output_str: .asciiz "The sum of GP is: "

    f: .word 0      #First term of GP
    r: .word 0      #Common Ratio of GP
    n: .word 0      #Number of terms of GP
    sum: .word 0    #Initialize integer 'sum' with value 'zero'

```

- **'*.text'*** section includes the actual code of your program. As the comments are very descriptive itself. First of all, all the *inputs* are taken from the user and stored into *memory*.

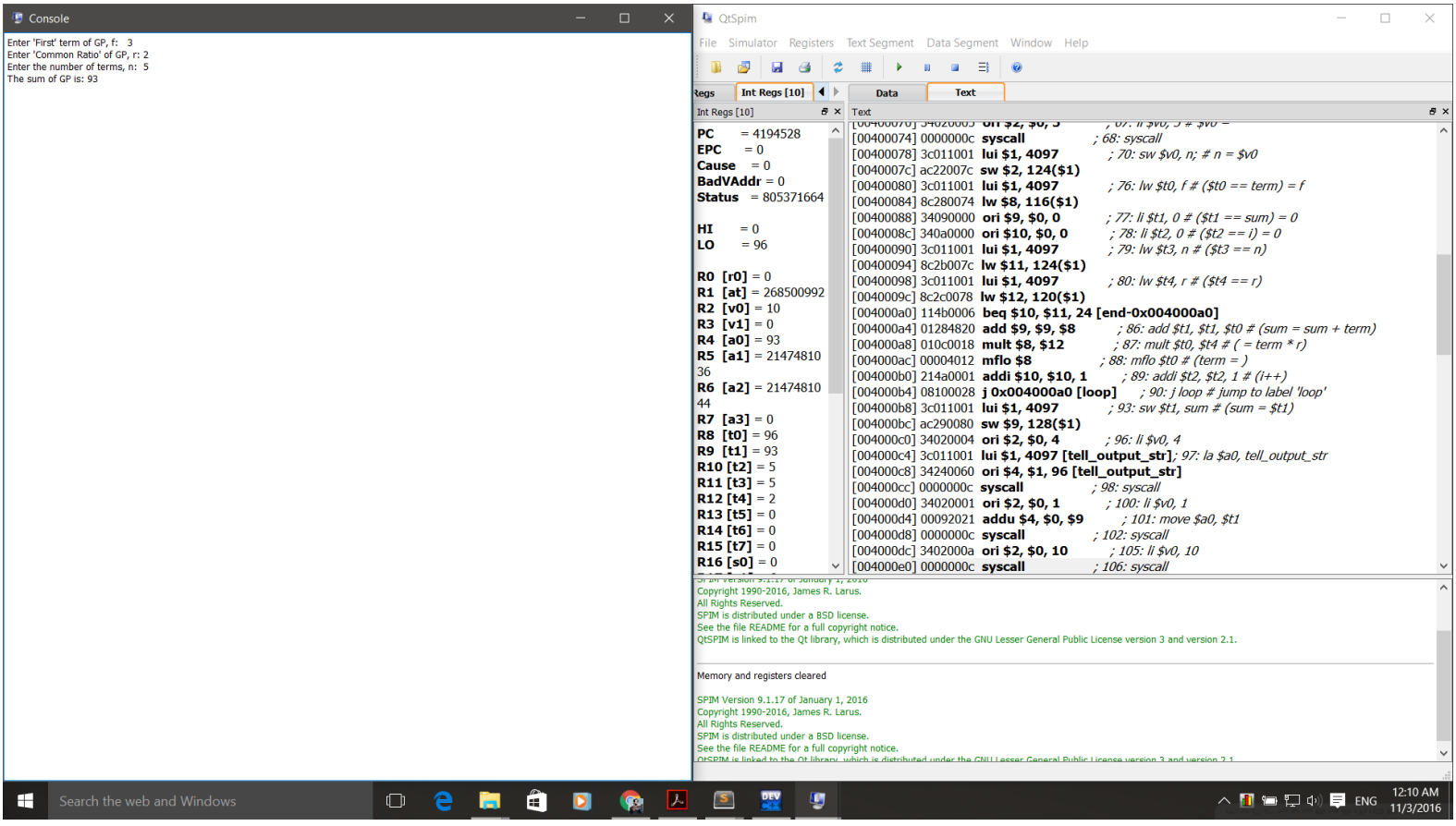
```
.text
main:
    #Prompts user to enter 'First' term of GP**************************************************
    la  $a0, ask_input_str1                         #loads the address of string to be printed in $a0
    li  $v0, 4                                  #system call code for printing string
    syscall                                     #system call
    #Takes input the 'First' term of GP
    li  $v0, 5      # $v0 = <user input>
    syscall
    #Store to memory
    sw $v0, f;      # f = $v0



    #Prompts user to enter 'Common Ratio' of GP************************************************
    la  $a0, ask_input_str2                         #loads the address of string to be printed in $a0
    li  $v0, 4                                  #system call code for printing string
    syscall                                     #system call
    #Takes input the 'Common Ratio' of GP
    li  $v0, 5      # $v0 = <user input>        #system call code for integer input
    syscall
    #Store to memory
    sw $v0, r;      # r = $v0



    #Prompts user to enter 'Number' of terms of GP*********************************************
    la  $a0, ask_input_str3                         #loads the address of string to be printed in $a0
    li  $v0, 4                                  #system call code for printing string
    syscall                                     #system call
    #Takes input the 'Number' of terms of GP
    li  $v0, 5      # $v0 = <user input>
    syscall
    #Store to memory
    sw $v0, n;      # n = $v0

```

- Now the *temporary* variables are declared. Then, a **loop** is executed in which the **term** is generated and it is added to the **sum** each time. Also, the *counter variable* checks the number of iterations are no more than the '**number**' of terms.

```
75      # Temporary variables
76      lw $t0, f   # ($t0 == term) = f
77      li $t1, 0   # ($t1 == sum) = 0
78      li $t2, 0   # ($t2 == i) = 0
79      lw $t3, n   # ($t3 == n)
80      lw $t4, r   # ($t4 == r)
81
82
83
84      # LOOP Statement
85      loop: beq $t2, $t3, end # if(i == n) jump to label 'end'
86          add $t1, $t1, $t0       # (sum = sum + term)
87          mult $t0, $t4           # (<product> = term * r)
88          mflo $t0                # (term = <product>)
89          addi $t2, $t2, 1        # (i++)
90          j loop              # jump to label 'loop'
91      end:
92          #Store result back to memory
93          sw $t1, sum         # (sum = $t1)
94
95          #Prints the 'output_str' to Console
96          li  $v0, 4
97          la  $a0, tell_output_str
98          syscall
99          #Prints 'GP - Sum' to Console
100         li  $v0, 1
101         move $a0, $t1
102         syscall
103
104         # Program exits
105         li  $v0, 10
106         syscall
```

Test Case #1

- First term, f = 3
- Common Ratio, r = 2
- Number of terms, n = 5
- @ SUM = 93



Test Case #2

- First term, f = 5
- Common Ratio, r = 3
- Number of terms, n = 4
- @ SUM = 200