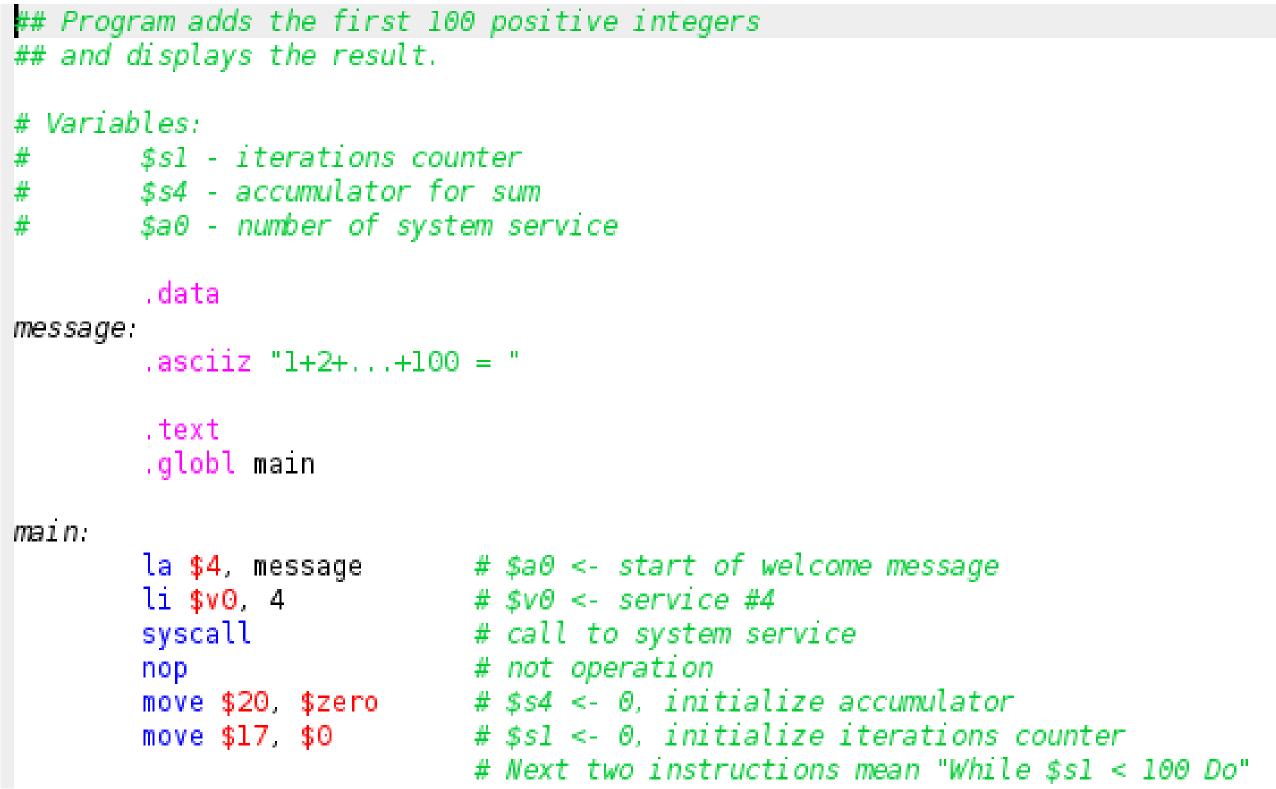
15CSE 381 Computer Organization and Architecture

Lab 2

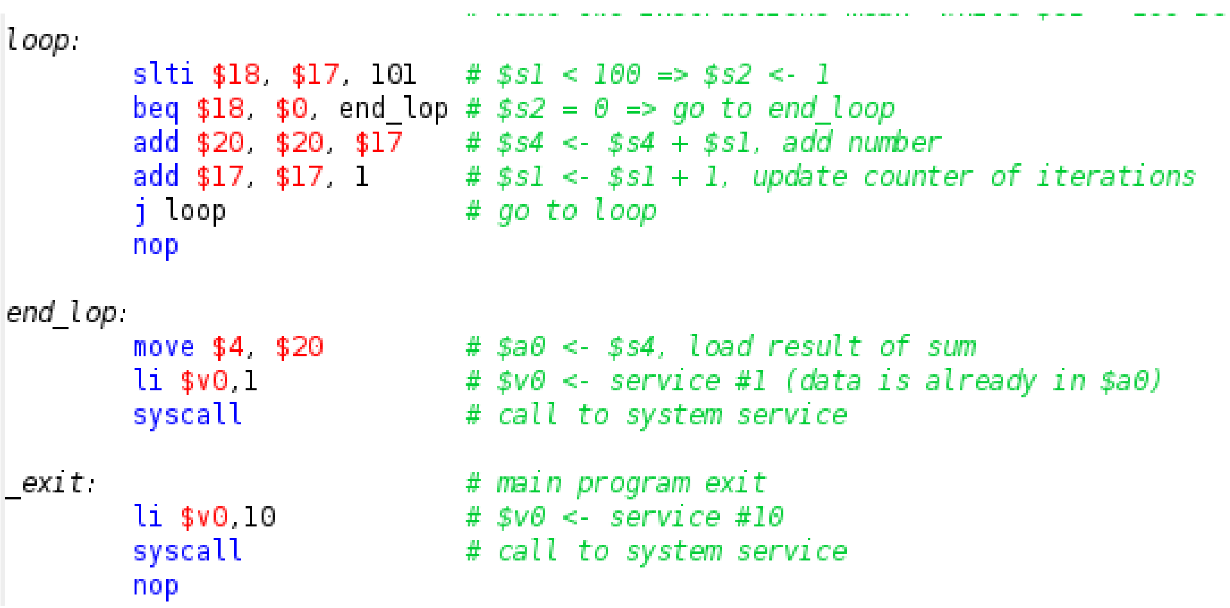
5th July 2019

1 Practice Programs

1.1 Sum of first 100 natural numbers

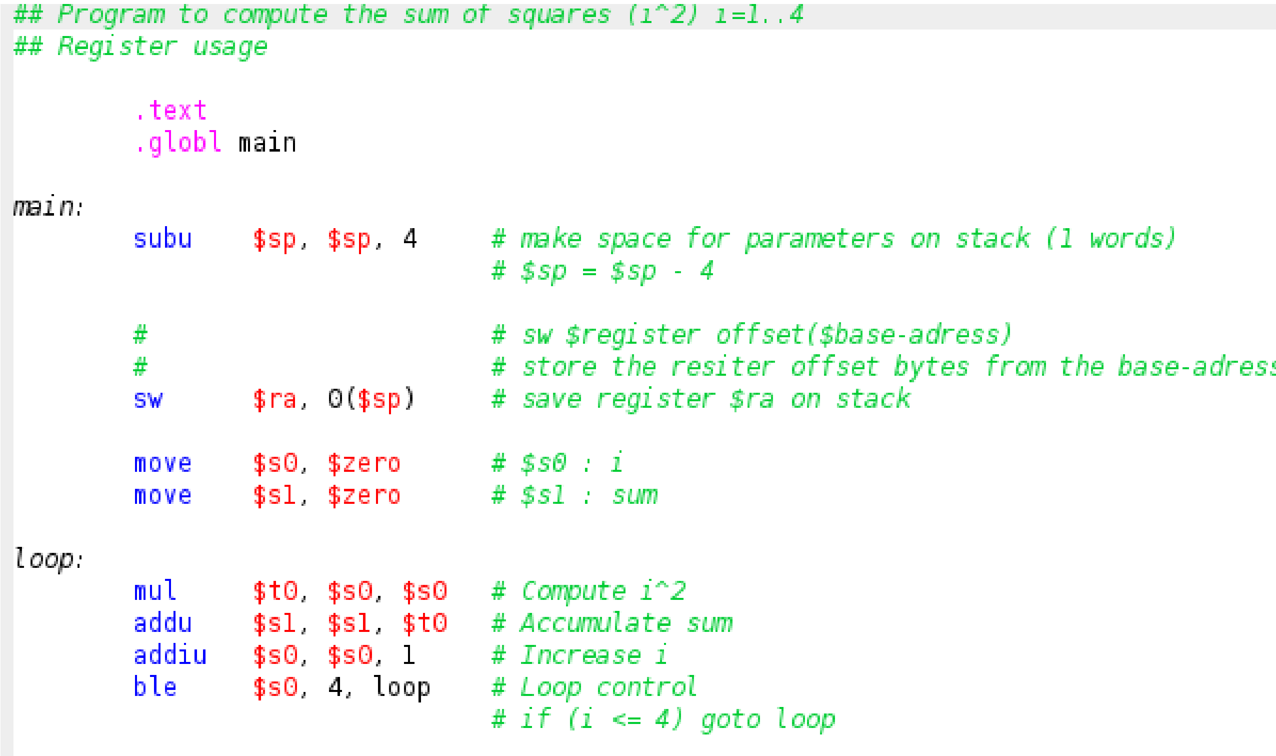


```
## Program adds the first 100 positive integers
## and displays the result.

# Variables:
#       $s1 - iterations counter
#       $s4 - accumulator for sum
#       $a0 - number of system service

        .data
message:
        .asciiz "1+2+...+100 = "

        .text
        .globl main

main:
        la $4, message       # $a0 <- start of welcome message
        li $v0, 4            # $v0 <- service #4
        syscall              # call to system service
        nop                  # not operation
        move $20, $zero      # $s4 <- 0, initialize accumulator
        move $17, $0         # $s1 <- 0, initialize iterations counter
                             # Next two instructions mean "While $s1 < 100 Do"
```

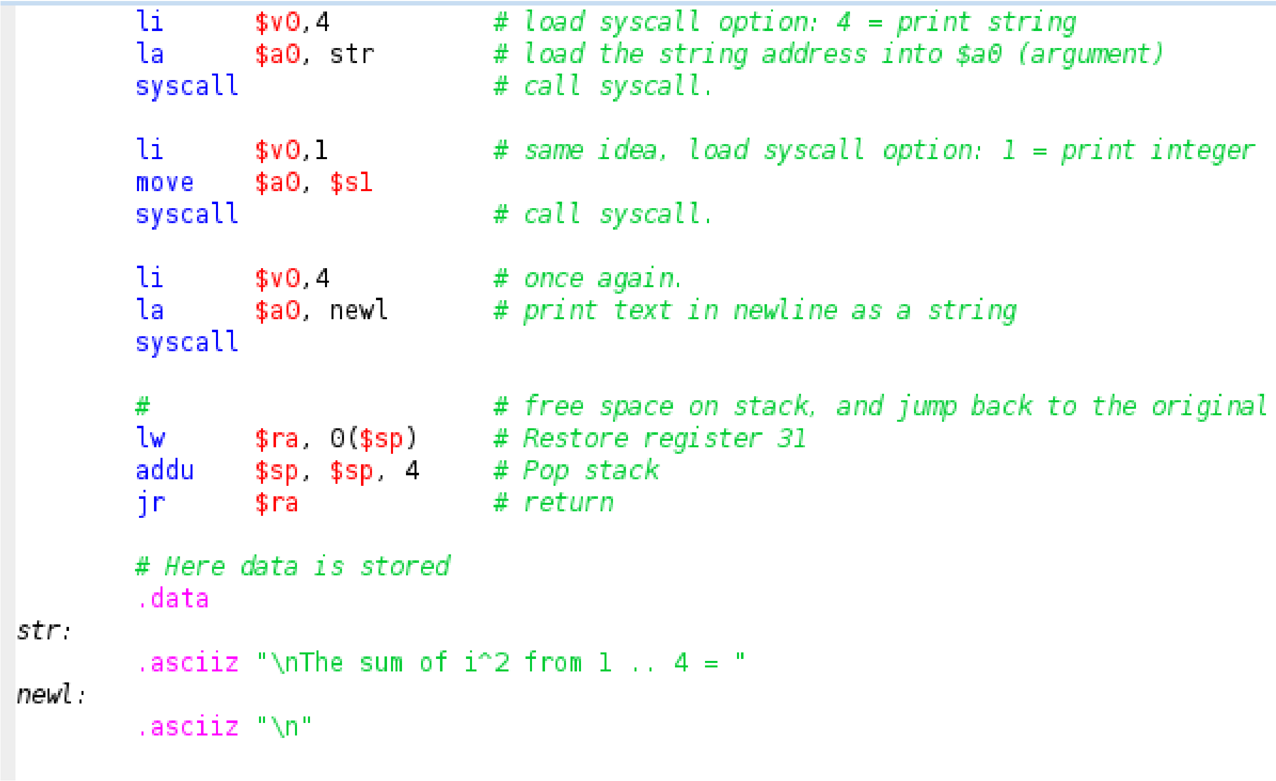


```
loop:
        slti $18, $17, 101   # $s1 < 100 => $s2 <- 1
        beq $18, $0, end_lop # $s2 = 0 => go to end_loop
        add $20, $20, $17    # $s4 <- $s4 + $s1, add number
        add $17, $17, 1      # $s1 <- $s1 + 1, update counter of iterations
        j loop               # go to loop
        nop

end_lop:
        move $4, $20         # $a0 <- $s4, load result of sum
        li $v0,1             # $v0 <- service #1 (data is already in $a0)
        syscall              # call to system service

_exit:                       # main program exit
        li $v0,10            # $v0 <- service #10
        syscall              # call to system service
        nop
```

1.2 Sum of squares of first 4 numbers



```
## Program to compute the sum of squares (i^2) i=1..4
## Register usage

        .text
        .globl main

main:
        subu    $sp, $sp, 4     # make space for parameters on stack (1 words)
                                # $sp = $sp - 4

        #                       # sw $register offset($base-adress)
        #                       # store the resiter offset bytes from the base-adress
        sw      $ra, 0($sp)     # save register $ra on stack

        move    $s0, $zero      # $s0 : i
        move    $s1, $zero      # $s1 : sum

loop:
        mul     $t0, $s0, $s0   # Compute i^2
        addu    $s1, $s1, $t0   # Accumulate sum
        addiu   $s0, $s0, 1     # Increase i
        ble     $s0, 4, loop    # Loop control
                                # if (i <= 4) goto loop
```



```
        li      $v0,4               # load syscall option: 4 = print string
        la      $a0, str            # load the string address into $a0 (argument)
        syscall                     # call syscall.

        li      $v0,1               # same idea, load syscall option: 1 = print integer
        move    $a0, $s1
        syscall                     # call syscall.

        li      $v0,4               # once again.
        la      $a0, newl           # print text in newline as a string
        syscall

        #                           # free space on stack, and jump back to the original
        lw      $ra, 0($sp)         # Restore register 31
        addu    $sp, $sp, 4         # Pop stack
        jr      $ra                 # return

        # Here data is stored
        .data
str:
        .asciiz "\nThe sum of i^2 from 1 .. 4 = "
newl:
        .asciiz "\n"
```

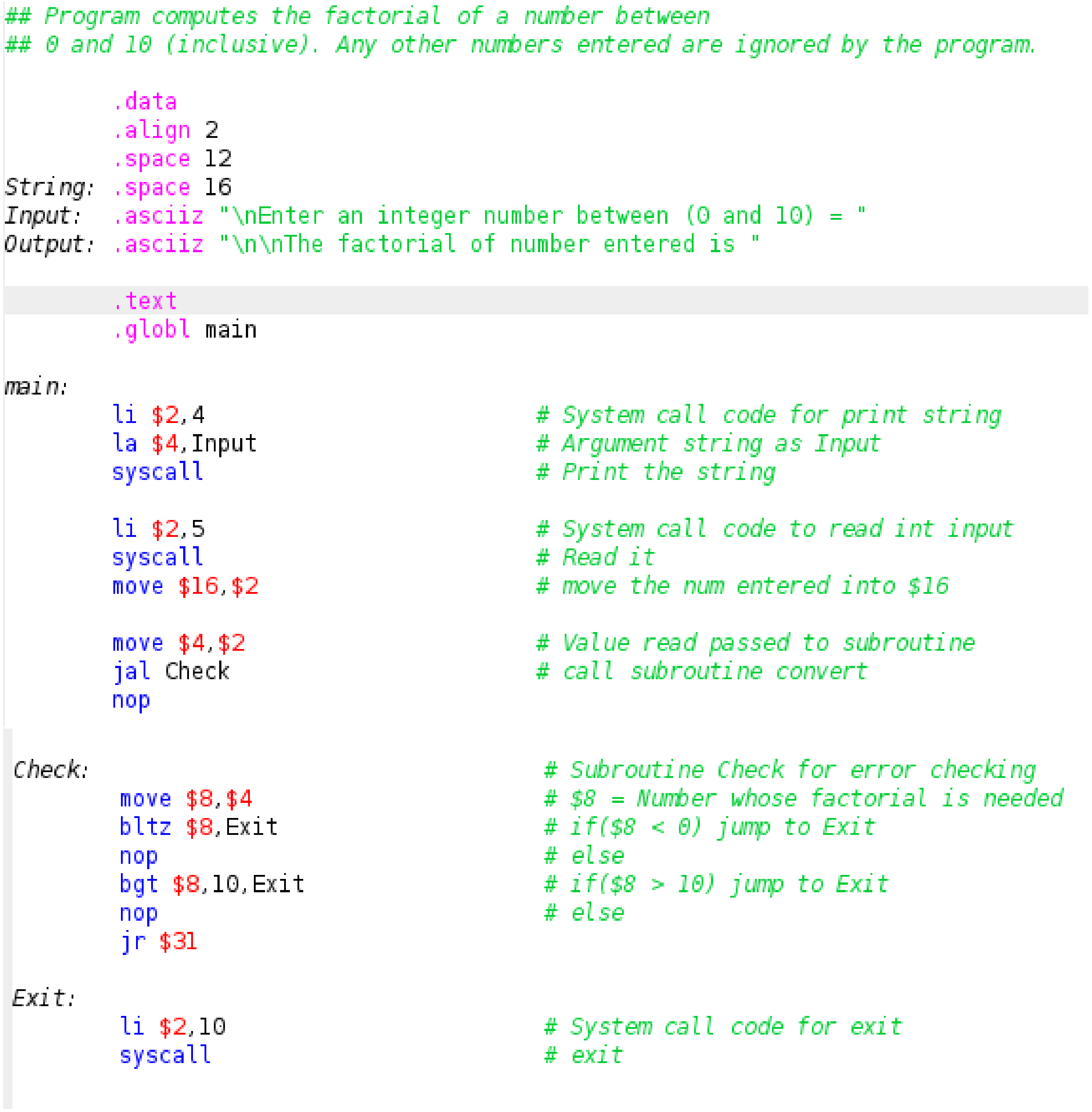
2 Questions

1. We have familiarized sum of first 100 natural numbers. Now write a program to add first N natural numbers. The program should ask the value of N.

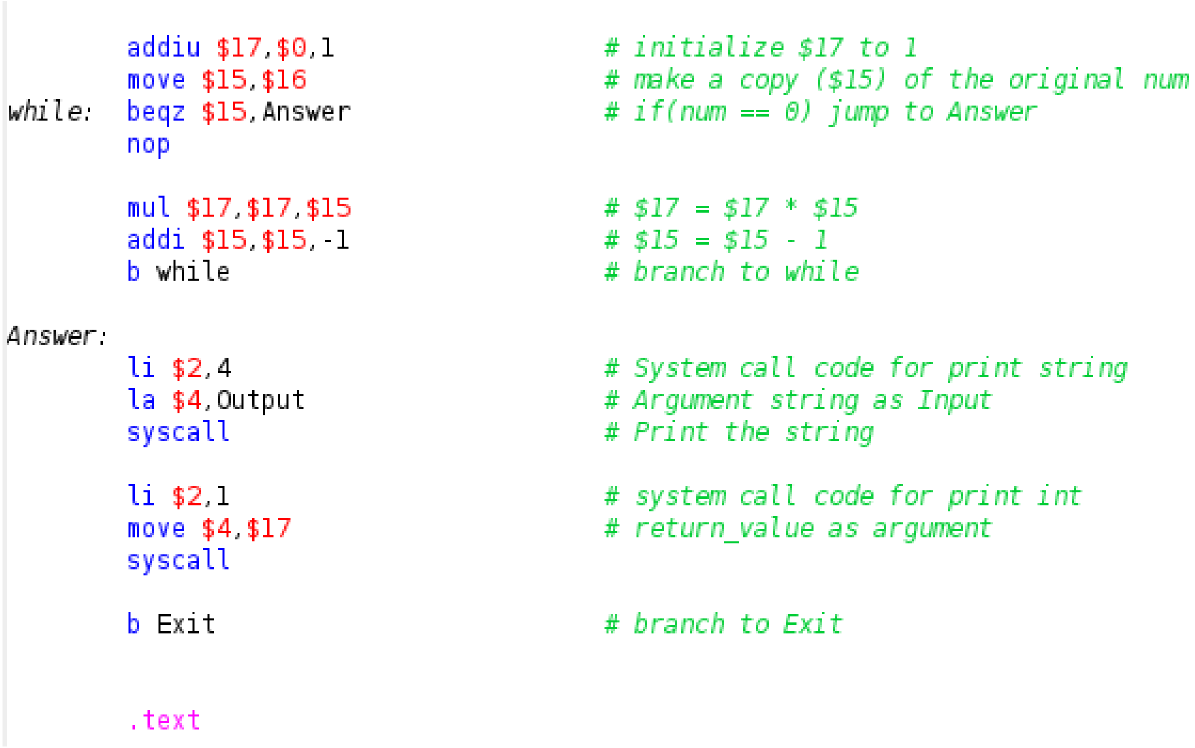
2. Write a MIPS program to find sum of squares of first N natural numbers.

3 Practice Program

3.1 Finding n factorial



```
## Program computes the factorial of a number between
## 0 and 10 (inclusive). Any other numbers entered are ignored by the program.

        .data
        .align 2
        .space 12
String: .space 16
Input:  .asciiz "\nEnter an integer number between (0 and 10) = "
Output: .asciiz "\n\nThe factorial of number entered is "

        .text
        .globl main

main:
        li $2,4                         # System call code for print string
        la $4,Input                     # Argument string as Input
        syscall                         # Print the string

        li $2,5                         # System call code to read int input
        syscall                         # Read it
        move $16,$2                     # move the num entered into $16

        move $4,$2                      # Value read passed to subroutine
        jal Check                       # call subroutine convert
        nop

Check:                                  # Subroutine Check for error checking
        move $8,$4                      # $8 = Number whose factorial is needed
        bltz $8,Exit                    # if($8 < 0) jump to Exit
        nop                             # else
        bgt $8,10,Exit                  # if($8 > 10) jump to Exit
        nop                             # else
        jr $31

Exit:
        li $2,10                        # System call code for exit
        syscall                         # exit
```



```
        addiu $17,$0,1                  # initialize $17 to 1
        move $15,$16                    # make a copy ($15) of the original num
while:  beqz $15,Answer                 # if(num == 0) jump to Answer
        nop

        mul $17,$17,$15                 # $17 = $17 * $15
        addi $15,$15,-1                 # $15 = $15 - 1
        b while                         # branch to while

Answer:
        li $2,4                         # System call code for print string
        la $4,Output                    # Argument string as Input
        syscall                         # Print the string

        li $2,1                         # system call code for print int
        move $4,$17                     # return_value as argument
        syscall

        b Exit                          # branch to Exit


        .text
```