INDIAN INSTITUTE OF TECHNOLOGY PATNA

CS210-Computer Architecture

**Lab 5: MARS MIPS simulator**

**Execute the sample codes of MIPS as an exercise and record the outputs in a word file:**

1. Code 1:

main: addi $t1, $zero, 97

# sample1.asm

Examine the conent of the register

Your Answer: $0 = 0x00000000            $t1 = 0x00000061

Array Declaration with Initialization

.data

data1: .byte 1,2,3,4,5,6

vowels: .byte 'a', 'e', 'i', 'o', 'u'

pow2: .word 1, 2, 4, 8, 16, 32, 64, 128

Examine the content of the memory (data segment) starting at **0x 1001 0000**

Your Answer:0x04030201    0x00756f69      0x00000002     0x00000008 0x00000010

1. Code 2:

.data

msg: .asciiz "Hello"

.text

main: li $v0, 4

la $a0, msg

syscall

# sample2.asm

**Your Answer:**

Figure out where and how is the string “Hello” stored? Write out the ASCII values,

in hexadecimal form, of the characters ‘H’, ‘e’, ‘l’ and ‘o’ below:

‘H’: **0x48**‘e’: **0x65**‘l’: **ox6c**‘o’: **ox6f**

1. Code 3:

# Data stored in memory

.data

value: # a value stored in memory

.word 7

result: # where result is stored

.word 0

# The program

.text

lw $s0, value

move $s1, $s0

sll $s0, $s0, 2

add $s0, $s0, $s1

sw $s0, result

ANS : ----->   $at : 268500992

                      $s0 : 0x00000007

                      $s01: 0x00000007

                      $s0: 0x0000001c

                      $s0: 0x00000023

$at: 0x10010000

0x10010000

1. Code 4:

# Sample MIPS program that writes to a new file.

# by Kenneth Vollmar and Pete Sanderson

.data

fout: .asciiz "testout.txt" # filename for output

buffer: .asciiz "The quick brown fox jumps over the lazy dog."

.text

# Open (for writing) a file that does not exist

li $v0, 13 # system call for open file

la $a0, fout # output file name

li $a1, 1 # Open for writing (flags are 0: read, 1: write)

li $a2, 0 # mode is ignored

syscall # open a file (file descriptor returned in $v0)

move $s6, $v0 # save the file descriptor

# Write to file just opened

li $v0, 15 # system call for write to file

move $a0, $s6 # file descriptor

la $a1, buffer # address of buffer from which to write

li $a2, 44 # hardcoded buffer length

syscall # write to file

# Close the file

li $v0, 16 # system call for close file

move $a0, $s6 # file descriptor to close

syscall # close file

$vo : 0x0000000d

$at : 0x10010000

$a0 : 0x10010000

$a1 : 0x00000001

$a2:0x00000000

$

1. Code 5

.data

msg1: .asciiz "Enter the first number: "

msg2: .asciiz "\nEnter the second number: "

result: .asciiz "\nThe result of addition is: "

.text

li $v0,4

la $a0,msg1

syscall

li $v0,5

syscall

move $t1,$v0

li $v0,4

la $a0,msg2

syscall

li $v0,5

Syscall

move $t2,$v0

Add $t3,$t1,$t2

li $v0,4

la $a0,msg3

syscall

li $v0,1

move $a0,$t3

syscall

li $v0,10

syscall

ANS : $v0 : 0x00000001

$at : 0x10010000

$a0 : 0x10010000

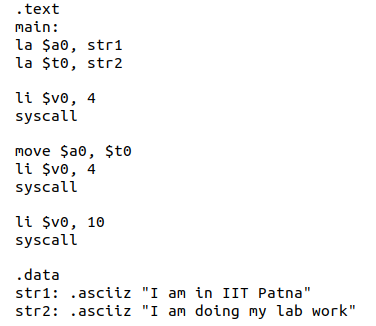
$v0: 0x00000005

**Show the usage of each system call using codes of your own. The usage of system call codes 4 and 10 are shown in the sample code below.**

MIPS provides a small set of operating-system-like services through the system call (syscall) instruction. To request a service, a program loads the system call code (see Table 1) into register $v0 and the arguments into registers $a0$\ldots$$a3 (or $f12 for floating point values). System calls that return values load their result in register $v0 (or $f0 for floating point results).



For example, to print two strings, use the following command:



For each of the system call code, create your own example and test the system call functionality

**Your code here:**

1. System call code 1: Print an integer:
2. # data segment

.data

num: .word 5 # integer to print

# text segment

.text

main:

li $v0, 1 # load the system call code for printing an integer into $v0

lw $a0, num # load the integer to print into $a0

syscall # make the syscall to print the integer

li $v0, 10 # load the system call code for exiting the program into $v0

syscall # make the syscall to exit the program

**Write 5 assembly level MIPS programs which illustrates some basic algorithms. Simulate using MIPS assembler. Comment each line of the code**

1) : program to find a factorial of integer

    # data segment

    .data

    num: .word 5 # integer to find the factorial of

    result: .word 1 # variable to store the result

    # text segment

    .text

    main:

        li $t0, 1 # initialize the counter to 1

        lw $t1, num # load the input number into $t1

    loop:

        mul $t2, $t0, $t1 # multiply the counter and the input number and store the result in $t2

        sw $t2, result # store the result in the 'result' variable

        addi $t0, $t0, 1 # increment the counter

        sub $t3, $t1, 1 # decrement the input number

        bne $t3, $0, loop # if the input number is not zero, go back to the loop

        li $v0, 1 # load the system call code for printing an integer

        lw $a0, result # load the result into $a0

        syscall # make the syscall to print the result

        li $v0, 10 # load the system call code for exiting the program

        syscall # make the syscall to exit the program

2) : program to find a GCD of the integer

    # data segment

    .data

    num1: .word 8 # first integer

    num2: .word 12 # second integer

    result: .word 0 # variable to store the result

    # text segment

    .text

    main:

        lw $t0, num1 # load the first integer into $t0

        lw $t1, num2 # load the second integer into $t1

    loop:

        rem $t2, $t0, $t1 # find the remainder of dividing $t0 by $t1 and store the result in $t2

        beq $t2, $0, exit # if the remainder is zero, go to the exit label

        move $t0, $t1 # move the contents of $t1 into $t0

        move $t1, $t2 # move the contents of $t2 into $t1

        b loop # go back to the loop

    exit:

        move $t2, $t0 # move the contents of $t0 into $t2

        sw $t2, result # store the result in the 'result' variable

        li $v0, 1 # load the system call code for printing an integer

        lw $a0, result # load the result into $a0

        syscall # make the syscall to print the result

        li $v0, 10 # load the system call code for exiting the program

        syscall # make the syscall to exit the program

3) : program for substraction

.data

message1: .asciiz "Enter the any number to subtract :"

message2: .asciiz "\nEnter the any number to subtract :"

n1 : .word 0

n2 : .word 0

message3: .asciiz "\nThe subtraction of the two numbers is "

.text

main:

li $v0 4 #print out message1

la $a0 message1

syscall

li $v0 5 #read message1 as number1

syscall

sw $v0 n1 #store number

li $v0 4 #print out message2

la $a0 message2

syscall

li $v0 5 #read message2 as number2

syscall

sw $v0 n2 #store number

li $v0 4

la $a0 message3

syscall

lw $t0 n1

lw $t1 n2

sub $t0, $v0, $v0 # t0 = number1 s1 - number2 s2

li $v0, 1 # print integer

move $t0, $a0 # move t0 to a0

syscall # run

4 :) program for addition

li $v0,10

li $t1, 10

add $t2, $t1, $t1

sll $t3, $t1, 2

and $t4, $t1, 0x0000FFFF

or $t5, $t1, 0x0000FFFF

5 ); PROGRAM FOR FINDING A PRIME NO.

# The number is read through the keyboard

.text

.globl main

main:

# Display message to user for a number

li $v0, 4

la $a0, prompt1

syscall

# read keyboard into $v0 (number x is upper bound number to find primes)

li $v0, 5

syscall

# move the number from $v0 to $t0

move $t0, $v0 # $t0 = n

# store 2 in $t1 and $t2

li $t1, 2 # i

li $t2, 2 # j

L3: # for (int i=2; i<n; i++)

# store 0 in $t3

li $t3, 0 # p = 0;

L2: # for (int j=2; j<i; j++)

# do div of two numbers

div $t2, $t1

# store the remainder in $t4

mfhi $t4

# branch if remainder is not 0 to L1

bne $t4, 0, L1 # if (i % j == 0)

# set $t3 as 1

li $t3, 1 # p = 1

# if p=1 break to next i

beq $t3, 1, L4

L1: # if (i % j == 0)

# add 1 to t2

addi $t2, $t2, 1 # j++

# repeat code while j < i

ble $t2, $t1, L2

# print integer function call 1

# put the answer into $a0

li $v0, 1

move $a0, $t1

syscall # System.out.println(i)

#print comma

li $v0, 4

la $a0, comma

syscall

L4:

# add 1 to t1

addi $t1, $t1, 1 # i++

# repeat code while i < n

ble $t1, $t0, L3 # for (int i=2; i<n; i++)

.data

prompt1:

.asciiz "Enter a number "

comma:

.asciiz ","

End of the Lab submit your report and code here

<https://u.pcloud.com/#page=puplink&code=CHPkZ7PpCYpJkj4R0ks0vgHWOymIoeh9k>