**Bahria University, Lahore Campus**

Department of Computer Sciences

Lab Journal 012

**(Spring 2023)**

|  |  |  |
| --- | --- | --- |
| Course: | **Computer Architecture & Organization Lab** |  |
| Course Code: | CEL 221 | Max Marks: 30 |
| Faculty’s Name: | Maryam Munawar | Lab Engineer: |

Name: AFFAN AHMAD \_\_\_ Enroll No: \_03-134221-003\_

## Lab Tasks:

### Task1: 10 Minutes

The data structure for this new stringStack class is defined below.

class stringStack

{

int SIZE=7;

int elements[SIZE]; # Note that characters are stored as int

int last = SIZE-1;

push(String s) {

last = (last - s.length())-1;

elements[last] = s.length();

int i = last + 1;

for (char c in s)

{

elements[i] = c;

i = i + 1;

}

}

String pop()

{

int i = elements[last];

int j = last + 1;

last = last + i;

for ( ; j < last; j++) {

s = s + elements[j];

}

return s;

}

}

### Task2: 10 Minutes

subprogram global main()

{

register int multiplicand

register int multiplier

register int answer

m = prompt("Enter the multiplicand")

n = prompt("Enter the multiplier")

answer = Multiply(m, n)

print("The answer is: " + answer)

}

subprogram int multiply(int m, int n)

{

if (n == 1)

return m;

return m + multiply(m,n-1)

}

Solution:

.text

.globl main

main:

# register conventions

# $s0 - m

# $s1 - n

# $s2 - answer

la $a0, prompt1 # Get the multiplicand

jal PromptInt

move $s0, $v0

la $a0, prompt2 # Get the multiplier

jal PromptInt

move $s1, $v0

move $a0, $s0

move $a1, $s1

jal Multiply # Do multiplication

move $s2, $v0

la $a0, result #Print the answer

move $a1, $s2

jal PrintInt

jal Exit

Multiply:

addi $sp, $sp -8 # push the stack

sw $a0, 4($sp) #save $a0

sw $ra, 0($sp) # Save the $ra

seq $t0, $a1, $zero # if (n == 0) return

addi, $v0, $zero, 0 # set return value

bnez $t0, Return

addi $a1, $a1, -1 # set n = n-1

jal Multiply # recurse

lw $a0, 4($sp) # retrieve m

add $v0, $a0, $v0 # return m+multiply(m, n-1)

Return:

lw $ra, 0($sp) #pop the stack

addi $sp, $sp, 8

jr $ra

.data

prompt1: .asciiz "Enter the multiplicand: "

prompt2: .asciiz "Enter the multiplier: "

result: .ascii "The answer is: "

.include "utils.asm"

**ERROR FREE :**

.text

.globl main

main:

# register conventions

# $s0 - m

# $s1 - n

# $s2 - answer

la $a0, prompt1 # Get the multiplicand

li $v0 ,4

syscall

li ,$v0 5

syscall

move $s0, $v0

la $a0, prompt2 # Get the multiplier

li $v0 ,4

syscall

li ,$v0,5

syscall

move $s1, $v0

move $a0, $s0

move $a1, $s1

jal Multiply # Do multiplication

move $s2, $v0

la $a0, result #Print the answer

li $v0,4

syscall

li $v0,1

move $a0, $s2

syscall

li $v0,10

syscall

Multiply:

addi $sp, $sp -8 # push the stack

sw $a0, 4($sp) #save $a0

sw $ra, 0($sp) # Save the $ra

seq $t0, $a1, $zero # if (n == 0) return

addi, $v0, $zero, 0 # set return value

bnez $t0, Return

addi $a1, $a1, -1 # set n = n-1

jal Multiply # recurse

lw $a0, 4($sp) # retrieve m

add $v0, $a0, $v0 # return m+multiply(m, n-1)

Return:

lw $ra, 0($sp) #pop the stack

addi $sp, $sp, 8

jr $ra

.data

prompt1: .asciiz "Enter the multiplicand: "

prompt2: .asciiz "Enter the multiplier: "

result: .ascii "The answer is: "

#.include "utils.asm"

**EXPLANATION :**

In this program we take two input the first input is multiplicand like(65) and the second value the multiplier for example 3 the the multiply function is to add 65 :3 time and there is the condition in multiply function is that if the second input is equal to 1 then the function return the multiplicand and the answer is 135 (if you have any confusion go and check the error free code for the conformation).

### Task3: 10 Minutes

Write a recursive program to calculate factorial numbers. Use the definition of factorial as F(n) = n \* F(n-1)

USING STACK :

.data

msg: .asciiz "Enter a number"

answer: .asciiz "\nFactorial is: "

.text

li $v0, 4

la $a0, msg

syscall

li $v0, 5

syscall

move $a0, $v0

li $v0, 1

syscall

jal calculate\_factorial

move $a1, $v0

li $v0, 4

la $a0, answer

syscall

move $a0, $a1

li $v0, 1

syscall

li $v0, 10

syscall

calculate\_factorial:

addi $sp, $sp-4

sw $ra, ($sp)

li $v0, 1

multiply:

beq $a0, $zero, return

mul $v0, $v0, $a0

addi $a0, $a0, -1

j multiply

return:

lw $ra, ($sp)

jr $ra

ANOTHER METHOD NOT USING STACK

.data

prompt1: .asciiz "Please enter integer: "

a: .asciiz "result : "

.text

addi $t0,$zero,0

li $v0,4

la $a0,prompt1

syscall

li $v0,5

syscall

move $t1,$v0

move $t2,$t1

val:

sub $t2 ,$t2,1

beq $t2,1,r

mul $t1,$t1,$t2

b val

r :

li $v0,4

la $a0,a

syscall

li $v0 ,1

move $a0,$t1

syscall

li $v0,10

syscall

**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| a. | 10 |  |  |
| b. | 10 |  |  |
| c. | 10 |  |  |
|  |  |  |  |
| **Total** | **30** |  | **Signature** |

**Note : Attempt all tasks and get them checked by your Instructor**