**Logo

Description automatically generated San Francisco Bay University**

**EE488 - Computer Architecture**

**Homework Assignment #5**

**Due day: 3/29/2023**

**Instruction:**

1. **Push the answer sheet to GitHub in word file**
2. **Overdue homework submission could not be accepted.**
3. **Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**
4. Implement a subprogram that prompt the user for *3* numbers, finds the median (middle value) of the *3*, and returns that value to the calling program.

.data

prompt1: .asciiz "Enter first number:" # Store the prompt and output messages

prompt2: .asciiz "Enter second number:"

prompt3: .asciiz "Enter third number:"

output: .asciiz "Median of three numbers:"

.text

main:

la $a0,prompt1 # Load address of prompt1 to $a0

li $v0,4

syscall # Print the prompt message

li $v0,5 # Read integer value

syscall

move $t0,$v0 # Move read value which is first number to $t0

la $a0,prompt2 # Load address of prompt2 to $a0

li $v0,4

syscall # Print the prompt message

li $v0,5 # Read integer value

syscall

move $t1,$v0 # Move read value which is second number to $t1

la $a0,prompt3 # Load address of prompt23 to $a0

li $v0,4

syscall # Print the prompt message

li $v0,5 # Read integer value

syscall

move $t2,$v0 # Move read value which is third number to $t2

jal median # Call procedure

move $t3,$v0

la $a0,output

li $v0,4

syscall

move $a0,$t3

li $v0,1

syscall

li $v0,10 # Terminate program

syscall

median:

blt $t1,$t0,swap1 # If second number less than first branch to swap1

blt $t2,$t0,swap2 # If not then if third number less than first then branch to swap2

blt $t2,$t1,swap3 # If first number is the less then compare, third number with second, if less then branch to swap3

b ret # If the given numbers are in ascending order then do not swap

swap1:

move $t3,$t0 # Swap first and second number using temporary register $t3

move $t0,$t1

move $t1,$t3

blt $t2,$t1,swap3 # Check whether second number after swapping less than third, if yes branch to swap3

b ret # If not then branch to ret

swap2:

move $t3,$t0 # Swap first and third number using temporary register $t3

move $t0,$t2

move $t2,$t3

blt $t2,$t1,swap3 # Check whether third number after swapping less than second, if yes branch to swap3

b ret # If not then branch to ret

swap3:

move $t3,$t1 # Swap second and third number using temporary register $t3

move $t1,$t2

move $t2,$t3

b ret # branch to ret

ret:

move $v0,$t1 # Second number after arranging in ascending order will be median, so move it to $v0

jr $ra # Return

1. Implement a recursive program that takes in a number and finds the square of that number through addition. For example if the number *3* is entered, you would add *3+3+3=9*. If *4* is entered, you would add *4+4+4+4=16*. This program must be implemented using recursion to add the numbers together.

**Answer**

.data

num: .asciiz "Enter the number : " #enter number prompt

sum: .asciiz "Square = " #Result prompt

.text

#Enter number prompt display call

li $v0,4

la $a0,num

syscall

#Read entered number

li $v0,5

syscall

move $s0,$v0 #move number into s0 register

move $t0,$v0 #move number into t0 register for count

li $s1,0 #Initialize s1 for result

#loop for repeatative addition

loop:

blez $t0,exit #counter equal or less than zero exit the loop

add $s1,$s1,$s0 #repeatative addition

addi $t0,$t0,-1 #decrement count

j loop #unconditional jump

# exit and display result

exit:

li $v0,4 #Display square string prompt

la $a0,sum

syscall

li $v0,1 #result display

move $a0,$s1

syscall

li $v0,10 #end of the program

Syscall

1. Write a recursive program to calculate factorial numbers. Use the definition of factorial as *F(n) = n \* F(n-1)*
2. The following pseudo code converts an input value of a single decimal number from

into a single hexadecimal digit. Translate this pseudo code into MIPS assembly.

*main{*

*String a[16]*

*a[0] = "0x0"*

*a[1] = "0x1"*

*a[2] = "0x2"*

*a[3] = "0x3"*

*a[4] = "0x4"*

*a[5] = "0x5"*

*a[6] = "0x6"*

*a[7] = "0x7"*

*a[8] = "0x8"*

*a[9] = "0x9"*

*a[10] = "0xa"*

*a[11] = "0xb"*

*a[12] = "0xc"*

*a[13] = "0xd"*

*a[14] = "0xe"*

*a[15] = "0xf"*

*int i = prompt("Enter a number from 0 to 15 ")*

*print("your number is " + a[i])*

*}*

***Answer***

.text

.globl main

main:

li $v0,4

la $a0, prompt

syscall

li $v0,5

syscall

move $t0, $v0

li $v0,4

la $a0,message

syscall

la $t3, a

sll $t2, $t0, 2

add $t1, $t2, $t3

li $v0,4

la $a0, ($t1)

syscall

.data

prompt: .asciiz "Enter a number from 0 to 15: "

message: .asciiz "your number is "

a: .asciiz "0x0","0x1","0x2","0x3","0x4","0x5","0x6","0x7","0x8","0x9","0xa","0xb","0xc","0xd","0xe","0xf"

1. The following pseudo code program calculates the Fibonacci numbers from *1…n*, and stores them in an array. Translate this pseudo code into MIPS assembly, and use the PrintIntArray subprogram to print the results.

*main{*

*int size = PromptInt(“Enter a max Fibonacci number to calc: “)*

*int Fibonacci[size]*

*Fibonacci[0] = 0*

*Fibonacci[1] = 1*

*for (int i = 2; i < size; i++){*

*Fibonacci[i] = Fibonacci[i-1] + Fibonacci[i-2]*

*}*

*PrintIntArray(Fibonacci, size)*

*}*

**Answer**

.data

message: .asciiz "Enter a max Fibonacci number to calc:"

.text

main:

la $a0,message # Load address of message to $a0

li $v0,4

syscall # Print the prompt message

li $v0,5 # Read integer value

syscall

move $t0,$v0 # Move read value which is number of terms of fibonacci series

li $a0,0

li $v0,1

syscall # Print first term which is 0

li $a0,32

li $v0,11

syscall # Print character 32 which is ascii of space

li $a0,1

li $v0,1

syscall # Print second term which is 1

li $a0,32

li $v0,11

syscall # Print character 32 which is ascii of space

jal printarray # Call procedure

li $v0,10 # Terminate program

syscall

printarray:

addi $sp,$sp,-4 # Store $t0 in stack

sw $t0,0($sp)

li $t1,2 # Load 2 to $t1

blt $t1,$t0,loop # If $t1 less than size of array then branch to loop

ret: # Return to main

jr $ra

loop:

li $t2,0 # Load first two terms to $t2 and $t3 initially

li $t3,1

fibonacci:

add $t4,$t3,$t2 # Add previous two terms to get next term

move $a0,$t4 # Move the next term obtained to $a0

li $v0,1

syscall # Print next term

li $a0,32

li $v0,11

syscall # Print character 32 which is ascii of space

move $t2,$t3 # Update $t2 and $t3 as previous two terms

move $t3,$t4

addi $t1,$t1,1 # Increment $t1

blt $t1,$t0,fibonacci # If $t1 less than size of array then branch to fibonacci

j ret # After all terms printed return