**Logo

Description automatically generated San Francisco Bay University**

**EE488 - Computer Architecture**

**Homework Assignment #5**

**Due day: 8/7/2024**

**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.
5. 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.
6. Write a recursive program to calculate factorial numbers. Use the definition of factorial as *F(n) = n \* F(n-1)*
7. 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])*

*}*

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 1**

.data

prompt1: .asciiz "Enter first number: "

prompt2: .asciiz "Enter second number: "

prompt3: .asciiz "Enter third number: "

result: .asciiz "The median is: "

.text

main:

# Prompt for three numbers

la $a0, prompt1

li $v0, 4

syscall

li $v0, 5

syscall

move $t0, $v0

la $a0, prompt2

li $v0, 4

syscall

li $v0, 5

syscall

move $t1, $v0

la $a0, prompt3

li $v0, 4

syscall

li $v0, 5

syscall

move $t2, $v0

# Find the median

# if t0 > t1

bgt $t0, $t1, check\_t0\_t2

# else

bgt $t1, $t2, check\_t1\_t2

move $t3, $t1

j print\_result

check\_t0\_t2:

# if t0 < t2

blt $t0, $t2, check\_t0\_is\_median

# else

move $t3, $t2

j print\_result

check\_t1\_t2:

# if t2 > t0

bgt $t2, $t0, check\_t0\_t2\_again

# else

move $t3, $t0

j print\_result

check\_t0\_is\_median:

# t0 is the median

move $t3, $t0

j print\_result

check\_t0\_t2\_again:

# t2 is the median

move $t3, $t2

j print\_result

print\_result:

la $a0, result

li $v0, 4

syscall

move $a0, $t3

li $v0, 1

syscall

li $v0, 10

syscall

**ANSWER 2**

.data

input\_msg: .asciiz "Please enter an integer: "

output\_msg: .asciiz "The result is: "

newline: .asciiz "\n"

.text

.globl main

main:

# Print input message

li $v0, 4

la $a0, input\_msg

syscall

# Read integer

li $v0, 5

syscall

move $s0, $v0 # Store input in $s0

# Call square function

move $a0, $s0

jal calculate\_square

# Store result

move $s1, $v0

# Print output message

li $v0, 4

la $a0, output\_msg

syscall

# Print result

li $v0, 1

move $a0, $s1

syscall

# Print newline

li $v0, 4

la $a0, newline

syscall

# Exit program

li $v0, 10

syscall

calculate\_square:

# Initialize

move $t0, $a0 # Original number

li $t1, 0 # Counter

li $t2, 0 # Result

square\_loop:

beq $t1, $t0, square\_done

add $t2, $t2, $t0

addi $t1, $t1, 1

j square\_loop

square\_done:

move $v0, $t2

jr $ra

**ANSWER 3**

.data

input\_query: .asciiz "Input a non-negative integer: "

output\_prefix: .asciiz "Factorial result: "

newline: .asciiz "\n"

.text

.globl main

main:

# Display input query

li $v0, 4

la $a0, input\_query

syscall

# Get user input

li $v0, 5

syscall

move $s0, $v0 # Store input in $s0

# Compute factorial

move $a0, $s0

jal compute\_factorial

move $s1, $v0 # Store result in $s1

# Display output prefix

li $v0, 4

la $a0, output\_prefix

syscall

# Display factorial result

li $v0, 1

move $a0, $s1

syscall

# Print newline

li $v0, 4

la $a0, newline

syscall

# Exit program

li $v0, 10

syscall

compute\_factorial:

# Initialize

li $t0, 1 # Result

move $t1, $a0 # Counter

factorial\_loop:

beqz $t1, factorial\_done

mul $t0, $t0, $t1

addi $t1, $t1, -1

j factorial\_loop

factorial\_done:

move $v0, $t0

jr $ra

**ANSWER 4**

.data

array: .asciiz "0123456789abcdef"

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

result: .asciiz "Your number is: "

.text

main:

la $a0, prompt

li $v0, 4

syscall

li $v0, 5

syscall

move $t0, $v0

la $a0, result

li $v0, 4

syscall

la $t1, array

add $t1, $t1, $t0

lb $a0, 0($t1)

li $v0, 11

syscall

li $v0, 10

syscall

**ANSWER 5**

.data

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

space: .asciiz " "

newline: .asciiz "\n"

array: .word 0:100 # Assuming max size of 100, adjust if needed

.text

.globl main

main:

# Prompt for size

li $v0, 4

la $a0, prompt

syscall

# Read size

li $v0, 5

syscall

move $s0, $v0 # $s0 = size

# Initialize Fibonacci array

la $s1, array # $s1 = base address of array

sw $zero, 0($s1) # Fibonacci[0] = 0

li $t0, 1

sw $t0, 4($s1) # Fibonacci[1] = 1

# Calculate Fibonacci numbers

li $t0, 2 # i = 2

loop:

bge $t0, $s0, end\_loop

# Calculate Fibonacci[i] = Fibonacci[i-1] + Fibonacci[i-2]

sll $t1, $t0, 2 # $t1 = i \* 4 (offset)

add $t2, $s1, $t1 # Address of Fibonacci[i]

lw $t3, -4($t2) # Fibonacci[i-1]

lw $t4, -8($t2) # Fibonacci[i-2]

add $t5, $t3, $t4 # Fibonacci[i-1] + Fibonacci[i-2]

sw $t5, 0($t2) # Store result in Fibonacci[i]

addi $t0, $t0, 1 # i++

j loop

end\_loop:

# Call PrintIntArray

move $a0, $s1 # First argument: base address of array

move $a1, $s0 # Second argument: size

jal PrintIntArray

# Exit program

li $v0, 10

syscall

# PrintIntArray subprogram

PrintIntArray:

move $t0, $a0 # Base address of array

move $t1, $a1 # Size

li $t2, 0 # Counter

print\_loop:

bge $t2, $t1, print\_end

# Print integer

li $v0, 1

lw $a0, 0($t0)

syscall

# Print space

li $v0, 4

la $a0, space

syscall

addi $t0, $t0, 4 # Move to next element

addi $t2, $t2, 1 # Increment counter

j print\_loop

print\_end:

# Print newline

li $v0, 4

la $a0, newline

syscall

jr $ra # Return from subroutine