# **Date:8/07/24**

# 

# **EE488 - Computer Architecture**

# **HW Assignment 5**

**NAME : S A SABBIRUL MOHOSIN NAIM**

**ID : 20176**

**QUESTION ANS : 01 (python code)**

def find\_median():

print("Please enter three numbers.")

a = int(input("Enter the first number: "))

b = int(input("Enter the second number: "))

c = int(input("Enter the third number: "))

# Find the median using conditional logic

if a > b:

if a < c:

median = a

elif b > c:

median = b

else:

median = c

else:

if a > c:

median = a

elif b < c:

median = b

else:

median = c

return median

# Calling the function and printing the result

median\_value = find\_median()

print("The median value is:", median\_value)

**QUESTION ANS : 02 (Python code)**

def square\_recursive(n, count=None):

if count is None:

count = n # Initialize count to the value of n if it's the first call.

if count == 1:

return n # Base case: when count is 1, just return n.

else:

return n + square\_recursive(n, count - 1) # Recursive call, adding n each time and decreasing count.

def main():

number = int(input("Enter a number to square: ")) # Take user input.

result = square\_recursive(number) # Calculate the square using the recursive function.

print(f"The square of {number} is {result}.") # Output the result.

if \_\_name\_\_ == "\_\_main\_\_":

main() # Run the main function if this script is executed.

**QUESTION ANS : 03 (python code)**

def factorial\_recursive(n):

if n == 0:

return 1 # Base case: the factorial of 0 is 1

else:

return n \* factorial\_recursive(n - 1) # Recursive case: n \* factorial of (n-1)

def main():

number = int(input("Enter a number to find the factorial: ")) # Take user input

result = factorial\_recursive(number) # Calculate factorial using the recursive function

print(f"The factorial of {number} is {result}.") # Output the result

if \_\_name\_\_ == "\_\_main\_\_":

main() # Run the main function if this script is executed

**QUESTION ANS : 04**

.data

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

resultMsg: .asciiz "Your number is: "

hexDigits: .asciiz "0x00x11x22x33x44x55x66x77x88x99x0Ax0Bx0Cx0Dx0Ex0F" # Each hex digit takes three characters

.text

.globl main

main:

# Print prompt message

li $v0, 4

la $a0, prompt

syscall

# Read integer input from user

li $v0, 5

syscall

move $t0, $v0 # Move the read integer to $t0

# Calculate the address offset in the array

li $t1, 3 # Each hex digit representation is 3 characters long

mul $t0, $t0, $t1 # Offset = input number \* 3

# Load the base address of hexDigits

la $t2, hexDigits

# Calculate the exact address to read from

add $t2, $t2, $t0 # Address = base address + offset

# Print result message

li $v0, 4

la $a0, resultMsg

syscall

# Print the hex digit

li $v0, 4

move $a0, $t2

syscall

# Exit program

li $v0, 10

syscall

**QUESTION ANS : 05**

.data

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

space: .asciiz " "

newline: .asciiz "\n"

fibonacci: .word 0:100 # Allocate space for up to 100 Fibonacci numbers

.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 $t0, fibonacci

li $t1, 0

sw $t1, 0($t0) # fibonacci[0] = 0

li $t1, 1

sw $t1, 4($t0) # fibonacci[1] = 1

# Calculate Fibonacci numbers

li $t1, 2 # i = 2

fibonacci\_loop:

bge $t1, $s0, print\_array # if i >= size, exit loop

# fibonacci[i] = fibonacci[i-1] + fibonacci[i-2]

mul $t2, $t1, 4 # $t2 = i \* 4 (offset for current element)

sub $t3, $t2, 4 # $t3 = (i-1) \* 4 (offset for i-1 element)

sub $t4, $t2, 8 # $t4 = (i-2) \* 4 (offset for i-2 element)

lw $t5, fibonacci($t3)

lw $t6, fibonacci($t4)

add $t7, $t5, $t6

sw $t7, fibonacci($t2)

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

j fibonacci\_loop

print\_array:

# Call PrintIntArray subroutine

la $a0, fibonacci # array address

move $a1, $s0 # size

jal PrintIntArray

# Exit program

li $v0, 10

syscall

# PrintIntArray subroutine

PrintIntArray:

move $t0, $a0 # array address

move $t1, $a1 # size

li $t2, 0 # loop counter

print\_loop:

bge $t2, $t1, print\_end

# Print integer

li $v0, 1

lw $a0, ($t0)

syscall

# Print space

li $v0, 4

la $a0, space

syscall

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