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

# 

# **EE488 - Computer Architecture**

# **HW Assignment 5**

**NAME : S A SABBIRUL MOHOSIN NAIM**

**ID : 20176**

**QUESTION ANS : 01**

**.data**

**prompt: .asciiz "Enter a number: "**

**newline: .asciiz "\n"**

**.text**

**.globl main**

**main:**

**# Allocate space on stack for 3 numbers**

**addi $sp, $sp, -12**

**# Input first number**

**li $v0, 4**

**la $a0, prompt**

**syscall**

**li $v0, 5**

**syscall**

**sw $v0, 0($sp)**

**# Input second number**

**li $v0, 4**

**la $a0, prompt**

**syscall**

**li $v0, 5**

**syscall**

**sw $v0, 4($sp)**

**# Input third number**

**li $v0, 4**

**la $a0, prompt**

**syscall**

**li $v0, 5**

**syscall**

**sw $v0, 8($sp)**

**# Load numbers into registers for sorting**

**lw $t0, 0($sp)**

**lw $t1, 4($sp)**

**lw $t2, 8($sp)**

**# Sort the numbers (using bubble sort)**

**# Compare t0 and t1**

**ble $t0, $t1, check\_t1\_t2**

**move $t3, $t0**

**move $t0, $t1**

**move $t1, $t3**

**check\_t1\_t2:**

**# Compare t1 and t2**

**ble $t1, $t2, check\_t0\_t1**

**move $t3, $t1**

**move $t1, $t2**

**move $t2, $t3**

**check\_t0\_t1:**

**# Compare t0 and t1 again**

**ble $t0, $t1, done\_sorting**

**move $t3, $t0**

**move $t0, $t1**

**move $t1, $t3**

**done\_sorting:**

**# Median is now in $t1**

**move $a0, $t1**

**li $v0, 1**

**syscall**

**# Print newline**

**li $v0, 4**

**la $a0, newline**

**syscall**

**# Exit program**

**li $v0, 10**

**syscall**

**QUESTION ANS : 02**

.data

prompt: .asciiz "Enter a number to square: "

result: .asciiz "The square is: "

newline: .asciiz "\n"

.text

.globl main

main:

# Prompt for input

li $v0, 4

la $a0, prompt

syscall

# Read integer

li $v0, 5

syscall

move $a0, $v0 # Move input to $a0 for argument

move $a1, $v0 # Copy input to $a1 as well

# Call recursive function

jal square

# Print result

move $t0, $v0 # Save result

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t0

syscall

# Print newline

li $v0, 4

la $a0, newline

syscall

# Exit program

li $v0, 10

syscall

square:

# Base case: if n == 0, return 0

beqz $a0, base\_case

# Recursive case

addi $sp, $sp, -12 # Allocate stack space

sw $ra, 8($sp) # Save return address

sw $a0, 4($sp) # Save current n

sw $a1, 0($sp) # Save original number

addi $a0, $a0, -1 # n - 1

jal square # Recursive call with n-1

lw $a1, 0($sp) # Restore original number

add $v0, $v0, $a1 # Add original number to result

lw $ra, 8($sp) # Restore return address

addi $sp, $sp, 12 # Deallocate stack space

jr $ra # Return

base\_case:

li $v0, 0 # Return 0 for base case

jr $ra

**QUESTION ANS : 03**

.data

prompt: .asciiz "Enter a number to calculate its factorial: "

result: .asciiz "The factorial is: "

newline: .asciiz "\n"

.text

.globl main

main:

# Prompt for input

li $v0, 4

la $a0, prompt

syscall

# Read integer

li $v0, 5

syscall

move $a0, $v0 # Move input to $a0 for argument

# Call recursive factorial function

jal factorial

# Print result

move $t0, $v0 # Save result

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t0

syscall

# Print newline

li $v0, 4

la $a0, newline

syscall

# Exit program

li $v0, 10

syscall

factorial:

# Base case: if n <= 1, return 1

bgt $a0, 1, recursive\_case

li $v0, 1

jr $ra

recursive\_case:

# Save registers

addi $sp, $sp, -8

sw $ra, 4($sp)

sw $a0, 0($sp)

# Calculate F(n-1)

addi $a0, $a0, -1

jal factorial

# Restore n

lw $a0, 0($sp)

# Calculate n \* F(n-1)

mul $v0, $a0, $v0

# Restore $ra and adjust stack

lw $ra, 4($sp)

addi $sp, $sp, 8

# Return

jr $ra

**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