|  |  |
| --- | --- |
|  | **Computer Organization & Assembly Language**  **BSCS 3rd**  **Department of Computer Science**  **Bahria University, Lahore Campus** |

**Quiz: 4**

Date: Week 13, 14th June 2023

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Roll No: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation of CLO** | **Question Number** | **Marks** | **Obtained Marks** |
| **CLO 1,2,3: CLO statement**  *Simulate the internal representation of data, and show how data is stored and accessed in, I/O modules, and the interconnecting components of the computer systems* | 1 | 2.5 |  |
|  |  |  |
| **Total Marks** | | **2.5** |  |

**Question 1. Write assembly code of the bellow pseudo codes**

**a.**

The pseudo code for this subprogram follows.

Subprogram PrintIntArray(array, size)

{

print("[")

for (int i = 0; i < size; i++)

{

print("," + array[i])

}

print("]")

}

**Sol**

.text

.globl main

main:

la $a0, array\_base

lw $a1, array\_size

jal PrintIntArray

jal Exit

.data

array\_size: .word 5

array\_base:

.word 12

.word 7

.word 3

.word 5

.word 11

.text

# Subprogram: PrintIntArray

# Purpose: print an array of ints

# inputs: $a0 - the base address of the array

# $a1 - the size of the array

#

PrintIntArray:

addi $sp, $sp, -16 # Stack record

sw $ra, 0($sp)

sw $s0, 4($sp)

sw $s1, 8($sp)

sw $s2, 12($sp)

move $s0, $a0 # save the base of the array to $s0

# initialization for counter loop

# $s1 is the ending index of the loop

# $s2 is the loop counter

move $s1, $a1

move $s2, $zero

la $a0 open\_bracket # print open bracket

jal PrintString

loop:

# check ending condition

sge $t0, $s2, $s1

bnez $t0, end\_loop

sll $t0, $s2, 2 # Multiply the loop counter by

# by 4 to get offset (each element

# is 4 big).

add $t0, $t0, $s0 # address of next array element

lw $a1, 0($t0) # Next array element

la $a0, comma

jal PrintInt # print the integer from array

addi $s2, $s2, 1 #increment $s0

b loop

end\_loop:

li $v0, 4 # print close bracket

la $a0, close\_bracket

syscall

lw $ra, 0($sp)

lw $s0, 4($sp)

lw $s1, 8($sp)

lw $s2, 12($sp) # restore stack and return

addi $sp, $sp, 16

jr $ra

.data

open\_bracket: .asciiz "["

close\_bracket: .asciiz "]"

comma: .asciiz ","

.include "utils.asm"

b.

Pseudo code for this algorithm follws.

for (int i = 0; i < size-1; i++)

{

for (int j = 0; j < ((size-1)-i); j++)

{

if (data[j] > data[j+1])

{

swap(data, j, j+1)

}

}

}

swap(data, i, j)

int tmp = data[i];

data[i] = data[j];

data[j] = tmp;

}

**Sol**

.text

.globl main

main:

la $a0, array\_base

lw $a1, array\_size

jal PrintIntArray

la $a0, array\_base

lw $a1, array\_size

jal BubbleSort

jal PrintNewLine

la $a0, array\_base

lw $a1, array\_size

jal PrintIntArray

jal Exit

.data

array\_size: .word 8

array\_base:

.word 55

.word 27

.word 13

.word 5

.word 44

.word 32

.word 17

.word 36

.text

# Subproram: Bubble Sort

# Purpose: Sort data using a Bubble Sort algorithm

# Input Params: $a0 - array

# $a1 - array size

# Register conventions:

# $s0 - array base

# $s1 - array size

# $s2 - outer loop counter

# $s3 - inner loop counter

BubbleSort:

addi $sp, $sp -20 # save stack information

sw $ra, 0($sp)

sw $s0, 4($sp) # need to keep and restore save registers

sw $s1, 8($sp)

sw $s2, 12($sp)

sw $s3, 16($sp)

move $s0, $a0

move $s1, $a1

addi $s2, $zero, 0 #outer loop counter

OuterLoop:

addi $t1, $s1, -1

slt $t0, $s2, $t1

beqz $t0, EndOuterLoop

addi $s3, $zero, 0 #inner loop counter

InnerLoop:

addi $t1, $s1, -1

sub $t1, $t1, $s2

slt $t0, $s3, $t1

beqz $t0, EndInnerLoop

sll $t4, $s3, 2 # load data[j]. Note offset is 4 bytes

add $t5, $s0, $t4

lw $t2, 0($t5)

addi $t6, $t5, 4 # load data[j+1]

lw $t3, 0($t6)

sgt $t0, $t2, $t3

beqz $t0, NotGreater

move $a0, $s0

move $a1, $s3

addi $t0, $s3, 1

move $a2, $t0

jal Swap # t5 is &data[j], t6 is &data[j=1]

NotGreater:

addi $s3, $s3, 1

b InnerLoop

EndInnerLoop:

addi $s2, $s2, 1

b OuterLoop

EndOuterLoop:

lw $ra, 0($sp) #restore stack information

lw $s0, 4($sp)

lw $s1, 8($sp)

lw $s2, 12($sp)

lw $s3, 16($sp)

addi $sp, $sp 20

jr $ra

# Subprogram: swap

# Purpose: to swap values in an array of integers

# Input parameters: $a0 - the array containing elements to swap

# $a1 - index of element 1

# $a2 - index of elelemnt 2

# Side Effects: Array is changed to swap element 1 and 2

Swap:

sll $t0, $a1, 2 # calcualate address of element 1

add $t0, $a0, $t0

sll $t1, $a2, 2 # calculate address of element 2

add $t1, $a0, $t1

lw $t2, 0($t0) #swap elements

lw $t3, 0($t1)

sw $t2, 0($t1)

sw $t3, 0($t0)

jr $ra

# Subprogram: PrintIntArray

# Purpose: print an array of ints

# inputs: $a0 - the base address of the array

# $a1 - the size of the array

#

PrintIntArray:

addi $sp, $sp, -16 # Stack record

sw $ra, 0($sp)

sw $s0, 4($sp)

sw $s1, 8($sp)

sw $s2, 12($sp)

move $s0, $a0 # save the base of the array to $s0

# initialization for counter loop

# $s1 is the ending index of the loop

# $s2 is the loop counter

move $s1, $a1

move $s2, $zero

la $a0 open\_bracket # print open bracket

jal PrintString

loop:

# check ending condition

sge $t0, $s2, $s1

bnez $t0, end\_loop

sll $t0, $s2, 2 # Multiply the loop counter by

# by 4 to get offset (each element

# is 4 big).

add $t0, $t0, $s0 # address of next array element

lw $a1, 0($t0) # Next array element

la $a0, comma

jal PrintInt # print the integer from array

addi $s2, $s2, 1 #increment $s0

b loop

end\_loop:

li $v0, 4 # print close bracket

la $a0, close\_bracket

syscall

lw $ra, 0($sp)

lw $s0, 4($sp)

lw $s1, 8($sp)

lw $s2, 12($sp) # restore stack and return

addi $sp, $sp, 16

jr $ra

.data

open\_bracket: .asciiz "["

close\_bracket: .asciiz "]"

comma: .asciiz ","

.include "utils.asm"