CS224

Section No.: 6

Spring 2020

Lab No. 2

Osama Tanveer / 21801147

a

```
	.text
	.globl __main
	.eqv shift_amount 4
__main:
	# storing word in memory
	lui $a0, 0xAA00
	ori $a0, 0x00BB
	sw $a0, int
	# shift amount
	li $a1, shift_amount
	jal shiftLeftCircular

	move $a0, $v0
	li $v0, 34
	syscall

	li $v0, 4
	la $a0, nextLine
	syscall

	lw $a0, int
	li $a1, shift_amount
	jal shiftRightCircular

	move $a0, $v0
	li $v0, 34
	syscall

	li $v0, 10
	syscall

shiftLeftCircular:
	addi $sp, $sp, -16
	sw $s0, 0($sp)
	sw $s1, 4($sp)
	sw $s2, 8($sp)
	sw $s3, 12($sp)

	move $s0, $a0
	move $s1, $a1
	# remaining bits
	li $s2, 32
	sub $s2, $s2, $s1
	srlv $s3, $s0, $s2

	sll $s0, $s0, shift_amount
	or $s0, $s0, $s3
	move $v0, $s0
	lw $s0, 0($sp)
	lw $s1, 4($sp)
	lw $s2, 8($sp)
	lw $s3, 12($sp)
	addi $sp, $sp, 16
	jr $ra

shiftRightCircular:
	addi $sp, $sp, -16
	sw $s0, 0($sp)
	sw $s1, 4($sp)
	sw $s2, 8($sp)
	sw $s3, 12($sp)

	move $s0, $a0
	move $s1, $a1
	# remaining bits
	li $s2, 32
	sub $s2, $s2, $s1
	sllv $s3, $s0, $s2

	srl $s0, $s0, shift_amount
	or $s0, $s0, $s3
	move $v0, $s0
	lw $s0, 0($sp)
	lw $s1, 4($sp)
	lw $s2, 8($sp)
	lw $s3, 12($sp)
	addi $sp, $sp, 16
	jr $ra

	.data
	int: .word
	nextLine: .asciiz "\n"
```

b

```
		.text
	.globl __main

__main:
	jal createArray
	move $a0, $v0
	move $a1, $v1

	# a0 - starting address of array
	# a1 - size of array
	jal arrayOperations

	# Ending program
	li $v0, 10
	syscall
createArray:
	addi $sp, $sp, -16
	sw $s0, 0($sp)
	sw $s1, 4($sp)
	sw $s2, 8($sp)
	sw $s3, 12($sp)
	# inputting size
	la $a0, promptToEnterSize
	li $v0, 4
	syscall

	li $v0, 5
	syscall
	move $s1, $v0 #s1 contains size

	# declaring spcae on heap
	move $a0, $v0
	li $v0, 9
	syscall
	move $s0, $v0 # s0 contains the starting address

	# inputting values
	la $a0, promptToEnterValues
	li $v0, 4
	syscall

	# i
	li $s2, 0

	# addressing memory
	move $s3, $s0

forToInput:
	beq $s2, $s1, inputsTaken
	li $v0, 5
	syscall
	sw $v0, 0($s3)
	addi $s3, $s3, 4
	addi $s2, $s2, 1
	j forToInput

inputsTaken:
	move $v0, $s0
	move $v1, $s1

	lw $s0, 0($sp)
	lw $s1, 4($sp)
	lw $s2, 8($sp)
	lw $s3, 12($sp)
	addi $sp, $sp, 16
	jr $ra

# a0 - starting address
# a1 - size
arrayOperations:
	move $s0, $a0
	move $s1, $a1

	# MINIMUM
	addi $sp, $sp, -36
	sw $s0, 0($sp)
	sw $s1, 4($sp)
	sw $s2, 8($sp)
	sw $s3, 12($sp)
	sw $s4, 16($sp)
	sw $s5, 20($sp)
sw $s6, 24($sp)
sw $s7, 28($sp)
sw $ra, 32($sp)

# s0 - starting address, s1 - size , s2 - i
move $s0, $a0
move $s1, $a1
#addi $s1, $s1, 1
addi $s2, $zero, 1
lw $s3, 0($s0) # first number
addi $s0, $s0, 4

# storing ra to return to main on stack

jal min
lw $ra, 32($sp)

# priting minimum
la $a0, minStatement
li $v0, 4
syscall

move $a0, $s3
li $v0, 1
syscall

la $a0, nextLine
li $v0, 4
syscall

lw $s0, 0($sp)
lw $s1, 4($sp)
lw $s2, 8($sp)
lw $s3, 12($sp)
lw $s4, 16($sp)
lw $s5, 20($sp)
lw $s6, 24($sp)
lw $s7, 28($sp)
addi $sp, $sp, 36

# MAXIMUM
addi $sp, $sp, -36
sw $s0, 0($sp)
sw $s1, 4($sp)
sw $s2, 8($sp)
sw $s3, 12($sp)
sw $s4, 16($sp)
sw $s5, 20($sp)
sw $s6, 24($sp)
sw $s7, 28($sp)
sw $ra, 32($sp)

lw $s3, 0($s0) # s3 contains maximum
li $s5, 1 # i = 1
addi $s0, $s0, 4
jal max
la $a0, maxStatement
li $v0, 4
syscall

move $a0, $s3
li $v0, 1
syscall

la $a0, nextLine
li $v0, 4
syscall


lw $s0, 0($sp)
lw $s1, 4($sp)
lw $s2, 8($sp)
lw $s3, 12($sp)
lw $s4, 16($sp)
lw $s5, 20($sp)
lw $s6, 24($sp)
lw $s7, 28($sp)
lw $ra, 32($sp)
addi $sp, $sp, 36

# SUM
addi $sp, $sp, -36
sw $s0, 0($sp)
sw $s1, 4($sp)
sw $s2, 8($sp)
sw $s3, 12($sp)
sw $s4, 16($sp)
sw $s5, 20($sp)
sw $s6, 24($sp)
sw $s7, 28($sp)
sw $ra, 32($sp)

li $s2, 0 # i = 0
addi $s4, $s4, 0 # sum
jal sum

# printing sum
la $a0, sumStatement
li $v0, 4
syscall

move $a0, $s4
li $v0, 1
syscall

la $a0, nextLine
li $v0, 4
syscall

lw $s0, 0($sp)
lw $s1, 4($sp)
lw $s2, 8($sp)
lw $s3, 12($sp)
lw $s4, 16($sp)
lw $s5, 20($sp)
lw $s6, 24($sp)
lw $s7, 28($sp)
lw $ra, 32($sp)
addi $sp, $sp, 36

# PALINDROME
sw $s0, 0($sp)
sw $s1, 4($sp)
sw $s2, 8($sp)
sw $s3, 12($sp)
sw $s4, 16($sp)
sw $s5, 20($sp)
sw $s6, 24($sp)
sw $s7, 28($sp)
```

```
	sw $ra, 32($sp)

	# s0 start address
	# s3 contains last element address
	sll $s2, $s1, 2
	add $s2, $s2, $s0
	addi $s2, $s2, -4
	# for mid index
	li $s3, 2
	div $s1, $s3
	mflo $s3

	# i = 0
	li $s4, 0

	jal palindrome
	move $s7, $v0
	bne $s7, $zero, isPalindrome
	beq $s7, $zero, isNotPalindrome
jumpAfterPrinting:
	lw $s0, 0($sp)
	lw $s1, 4($sp)
	lw $s2, 8($sp)
	lw $s3, 12($sp)
	lw $s4, 16($sp)
	lw $s5, 20($sp)
	lw $s6, 24($sp)
	#lw $s7, 28($sp)
	lw $ra, 32($sp)
	addi $sp, $sp, 36

	jr $ra # return to main

isPalindrome:
	la $a0, isPal
	li $v0, 4
	syscall
	j jumpAfterPrinting
isNotPalindrome:
	la $a0, isNotPal
	li $v0, 4
	syscall
	j jumpAfterPrinting
```

```
palindrome:
        beq $s4, $s3, palindromeDone
        lw $s5, 0($s0)
        lw $s6, 0($s2)
        seq $v0, $s5, $s6
        addi $s0, $s0, 4
        addi $s2, $s2, -4
        addi $s4, $s4, 1
        j palindrome
palindromeDone:
        jr $ra
sum:
        beq $s2, $s1, sumDone
        lw $s3, 0($s0)
        add $s4, $s4, $s3
        addi $s0, $s0, 4
        addi $s2, $s2, 1
        j sum
sumDone:
        jr $ra
max:
        beq $s5, $s1, maxDone
        lw $s4, 0($s0)
        sgt $s6, $s4, $s3
        addi $s0, $s0, 4
        addi $s5, $s5, 1
        bne $s6, $zero, updateMax
        j max
updateMax:
        move $s3, $s4
        j max
maxDone:
        jr $ra
min:
        beq $s2, $s1, minDone
        lw $s4, 0($s0)
        slt $s5, $s3, $s4
        move $s7, $s3
        addi $s0, $s0, 4
        addi $s2, $s2, 1
        beq $s5, $zero, swapForMin
        j min
swapForMin:
        move $s3, $s4
        j min
minDone:
        jr $ra

        .data
        promptToEnterSize: .asciiz "Enter the size of the array: "
        promptToEnterValues: .asciiz "Enter the values: "
        minStatement: .asciiz "The minimum is "
        maxStatement: .asciiz "The maxmimum is "
        sumStatement: .asciiz "The sum is "
        nextLine: .asciiz "\n"
        isPal: .asciiz "The array is a palindrome."
        isNotPal: .asciiz "The array is not a palindrome."
```