CS224

Spring 2017

Lab 3

Berat Biçer

21503050

Section 05

.data

out: .asciiz "\nEnter an integer: "

out2: .asciiz "\nEnter 1 for BubbleSort"

out3: .asciiz "\nEnter 2 for MergeSort"

out4: .asciiz "\nEnter 3 for PrintArray"

out5: .asciiz "\nEnter 4 for exit"

endl: .asciiz "\n"

space: .asciiz " "

.text

# Instructions

li $v0, 4

la $a0, out

syscall

# Get array size

li $v0, 5

syscall

move $s0, $v0 # save size

move $a0, $s0

jal CreateArray

move $s1, $v0 # base address

addi $s2, $0, 1

addi $s3, $0, 2

addi $s4, $0, 3

addi $s5, $0, 4

mainLoop:

# Instructions

li $v0, 4

la $a0, out2

syscall

# Instructions

li $v0, 4

la $a0, out3

syscall

# Instructions

li $v0, 4

la $a0, out4

syscall

# Instructions

li $v0, 4

la $a0, out5

syscall

li $v0, 4

la $a0, endl

syscall

# Get input

li $v0, 5

syscall

beq $v0, $s2, label1

beq $v0, $s3, label2

beq $v0, $s4, label3

beq $v0, $s5, label4

j mainLoop

label1:

move $a0, $s1

move $a1, $s0

jal BubbleSort

j mainLoop

label2:

move $a0, $s1

move $a2, $s0

sll $t0, $s0, 2

add $a1, $s1, $t0

jal MergeSort

move $s1, $v0

j mainLoop

label3:

move $a0, $s1

move $a1, $s0

jal PrintArray

j mainLoop

# End of main loop

label4: # Exit

li $v0, 10

syscall

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Functions

SpecialCase: # a0: input, v0: return in binary

move $t0, $a0

label0:

sll $t0, $t0, 1

srl $t0, $t0, 1

beq $t0, 0, special

move $t0, $a0

labelDenormalized:

sll $t0, $t0, 9

srl $t0, $t0, 9

bne $t0, 0, denormalizedPart2

j labelInfinity

denormalizedPart2:

move $t0, $a0

sll $t0, $t0, 1

srl $t0, $t0, 24

beq $t0, 0, special

labelInfinity:

move $t0, $a0

sll $t0, $t0, 1

srl $t0, $t0, 24

beq $t0, 255, infinityPart2

j labelNaN

infinityPart2:

move $t0, $a0

sll $t0, $t0, 9

srl $t0, $t0, 9

beq $t0, $t0, special

labelNaN:

move $t0, $a0

sll $t0, $t0, 1

srl $t0, $t0, 24

beq $t0, 255, NaNPart2

j normalized

NaNPart2:

move $t0, $a0

sll $t0, $t0, 9

srl $t0, $t0, 9

bne $t0, 0, special

j normalized

special:

addi $v0, $0, 1

j SpecialCaseExit

normalized:

addi $v0, $0, 0

SpecialCaseExit:

jr $ra

# End of SpecialCase

#///////////////////////////////////////////////////////////////////////////

RandomFP: # v0: return integer value

addi $sp, $sp, -4

sw $ra, ($sp)

randomLoop:

li $v0, 41

syscall

addi $sp, $sp, -4

sw $a0, ($sp)

jal SpecialCase

beq $v0, 0, normalValue

addi $sp, $sp, 4

j randomLoop

normalValue:

lw $v0, ($sp)

lw $ra, 4($sp)

addi $sp, $sp, 8

jr $ra

# End of RandomFP

#///////////////////////////////////////////////////////////////////////////

CreateArray: # a0: number of elements, #v0: base address

sll $t0, $a0, 2

add $t0, $t0, $v0

move $t1, $a0

move $a0, $t0

li $v0, 9

syscall

move $a0, $t1

addi $sp, $sp, -8

sw $ra, 4($sp)

sw $v0, ($sp)

arrayLoop:

addi $sp, $sp, -8

sw $v0, 4($sp)

sw $a0, ($sp)

jal RandomFP

move $t0, $v0

lw $v0, 4($sp)

lw $a0, ($sp)

sw $t0, ($v0)

addi $sp, $sp, 8

addi $v0, $v0, 4

addi $a0, $a0, -1

bne $a0, 0, arrayLoop

# End of arrayLoop

lw $v0, ($sp)

lw $ra, 4($sp)

jr $ra

# End of CreateArray

#///////////////////////////////////////////////////////////////////////////

CompareFP: # a0: 1st, a1: 2nd; v0: greater, v1: lesser

move $t0, $a0

move $t1, $a1

CaseMSB:

srl $t0, $t0, 31

srl $t1, $t1, 31

bgt $t0, $t1, fLTs

blt $t0, $t1, fGTs

move $t2, $t0 # MSB saved

CaseExponent:

move $t0, $a0

move $t1, $a1

sll $t0, $t0, 1

sll $t1, $t1, 1

srl $t0, $t0, 24

srl $t1, $t1, 24

beq $t2, 0, PosExp

NegExp:

bgt $t0, $t1, fLTs

blt $t0, $t1, fGTs

j CaseFraction

PosExp:

bgt $t0, $t1, fGTs

blt $t0, $t1, fLTs

CaseFraction:

move $t0, $a0

move $t1, $a1

sll $t0, $t0, 9

sll $t1, $t1, 9

srl $t0, $t0, 9

srl $t1, $t1, 9

beq $t2, 0, PosFrac

NegFrac:

bgt $t0, $t1, fLTs

blt $t0, $t1, fGTs

j equal

PosFrac:

bgt $t0, $t1, fGTs

blt $t0, $t1, fLTs

j equal

fLTs:

equal:

move $v0, $a1

move $v1, $a0

j CompareFPexit

fGTs:

move $v0, $a0

move $v1, $a1

CompareFPexit:

jr $ra

# End of CompareFP

#///////////////////////////////////////////////////////////////////////////

BubbleSort: # BubbleSort function for an integer array

# a0: base, $a1: size

addi $t6, $0, 1

bge $t6, $a1, BubbleSortEnd

# Final address

sll $t0, $a1, 2

add $t0, $a0, $t0

addi $t0, $t0, -4

addi $t4, $0, 0

addi $t5, $a1, -1

move $t1, $a0 # base address copy

bsStart:

sortPass:

addi $sp, $sp, -24

sw $t0, 20($sp)

sw $t1, 16($sp)

sw $t2, 12($sp)

sw $ra, 8($sp)

sw $a1, 4($sp)

sw $a0, ($sp)

lw $a0, ($t1) # base item

lw $a1, 4($t1) # next item

jal CompareFP

lw $a0, ($sp)

lw $a1, 4($sp)

lw $ra, 8($sp)

lw $t2, 12($sp)

lw $t1, 16($sp)

lw $t0, 20($sp)

addi $sp, $sp, 24

sw $v1, ($t1)

sw $v0, 4($t1)

addi $t1, $t1, 4 # base + 4

bne $t1, $t0, sortPass # base address copy + 4

# End of sortPass

move $t1, $a0 # base address copy

addi $t0, $t0, -4 # final - 4

addi $t4, $t4, 1

bne $t4, $t5, sortPass

BubbleSortEnd:

jr $ra

# End of BubbleSort

#///////////////////////////////////////////////////////////////////////////

MergeSort: # a0: base, a1: last, a2: size

ble $a2, 1, MergeSortExit

addi $sp, $sp, -16

sw $ra, 12($sp)

sw $a2, 8($sp)

sw $a1, 4($sp)

sw $a0, ($sp)

srl $t0, $a2, 1 # newSize

sll $1, $t0, 2 # space

add $t2, $t1, $a0 # newLast

move $a2, $t0

move $a1, $t2

jal MergeSort

lw $a0, ($sp)

lw $a1, 4($sp)

lw $a2, 8($sp)

srl $t0, $a2, 1 # newSize

sll $t1, $t0, 2 # space

add $t2, $t1, $a0 # base = base + space

addi $t2, $t2, 4 # newBase = base + 4

sub $t0, $a2, $t0 # newSize = size - newSize

move $a0, $t2

move $a2, $t0

jal MergeSort

lw $a0, ($sp)

lw $a1, 4($sp)

lw $a2, 8($sp)

srl $t0, $a2, 1 # newSize1

sub $t1, $a2, $t0 # newSize2

sll $t2, $t0, 2 # space

add $t3, $a0, $t2

addi $t3, $t3, 4 # newBase

move $a1, $t0 # newSize1

move $a2, $t3 # newBase

move $a3, $t1 # newSize2

jal Merge

lw $ra, 12($sp)

addi $sp, $sp, 16

MergeSortExit:

move $v0, $a0

jr $ra

#///////////////////////////////////////////////////////////////////////////

Merge: # a0: FirstBase, a1: FirstSize, a2: SecondBase, a3: SecondSize, v0: new Base

ble $a1, 0, DirectExit # FirstSize

ble $a3, 9, DirectExit # SecondSize

else:

addi $sp, $sp, -8

sw $ra, 4($sp)

add $t0, $a1, $a3

li $v0, 9

la $a0, ($t0)

syscall

sw $v0, ($sp) # NewBase

MergeLoop:

addi $sp, $sp, -20

sw $v0, 16($sp)

sw $a0, 12($sp)

sw $a1, 8($sp)

sw $a2, 4($sp)

sw $a3, ($sp)

lw $t0, ($a0) # FirstItem

lw $t1, ($a2) # SecondItem

move $a0, $t0

move $a1, $t1

addi $sp, $sp, -8

sw $t0, 4($sp)

sw $t1, ($sp)

jal CompareFP

lw $t1, ($sp)

lw $t0, 4($sp)

lw $a3, 8($sp)

lw $a2, 12($sp)

lw $a1, 16($sp)

lw $a0, 20($sp)

lw $v0, 24($sp)

addi $sp, $sp, 28

beq $v1, $t0, LoadFirst

beq $v1, $t1, LoadSecond

j ExitLoop

LoadFirst:

lw $t0, ($v1)

addi $a0, $a0, 4

addi $v0, $v0, 4

addi $a1, $a1, -1

bne $a1, 0, MergeLoop

j ExitLoop

LoadSecond:

lw $t1, ($v1)

addi $a2, $a2, 4

addi $v0, $v0, 4

addi $a3, $a3, -1

bne $a3, 0, MergeLoop

j ExitLoop

ExitLoop:

beq $a1, 0, MergeExit

CopyFA:

lw $t0, ($a0)

sw $t0, ($v0)

addi $a0, $a0, 4

addi $v0, $v0, 4

addi $a1, $a1, -1

bne $a1, 0, CopyFA

beq $a3, 0, MergeExit

CopySA:

lw $t0, ($a2)

sw $t0, ($v0)

addi $a2, $a2, 4

addi $v0, $v0, 4

addi $a3, $a3, -1

bne $a3, 0, CopySA

MergeExit:

lw $v0, ($sp) # NewBase

lw $ra, 4($sp)

DirectExit:

jr $ra

# End of Merge

#///////////////////////////////////////////////////////////////////////////

PrintArray: # a0: base, $a1: size

sll $t0, $a1, 2

add $t0, $a0, $t0 # max address

addi $sp, $sp, -4

sw $a0, ($sp)

PrintLoop:

li $v0, 4

la $a0, space

syscall

lw $a0, ($sp)

lw $t1, ($a0)

li $v0, 1

la $a0, ($t1)

syscall

lw $a0, ($sp)

addi $a0, $a0, 4

sw $a0, ($sp)

bne $a0, $t0, PrintLoop

addi $sp, $sp, 4

jr $ra

# End of printArray

#///////////////////////////////////////////////////////////////////////////