**Code**:

.text

#-------------------------------------------

# Procedure: quicksort

# Argument:

# $a0: Base address of the array

# $a1: Number of array element - length

# Notes: Implement quicksort, base array

# at $a0 will be sorted after the routine

# is done.

#-------------------------------------------

quicksort:

# Caller RTE store

#registers: a0, a1, a2, a3; s3, s4

addi $sp, $sp, -36

sw $fp, 36($sp)

sw $ra, 32($sp)

sw $a0, 28($sp)

sw $a1, 24($sp)

sw $a2, 20($sp)

sw $a3, 16($sp)

sw $s4, 12($sp)

sw $s3, 8($sp)

addi $fp, $sp, 36

#parameters a0 = base address; a2 = low; a3 = high

addi $a3, $a1, -1 #last element of the array -> high

li $a2, 0 #set first index to 0 -> low

sort:

bge $a2, $a3, done # if(low < high)

jal partition # int pi = partition(arr, low, high)

# pi will be stored in $s4

#sort before partition -- sort(arr, low, pi-1)

move $s3, $a3 #store old high value in $s3

addi $a3, $s4, -1 # high (a3) = pi - 1

jal sort #recursive call

#sort after partition -- sort(arr, pi+1, high)

move $a3, $s3 #put the original high value back

addi $a2, $s4, 1 #replace the new low value (a2) with (pi+1)

jal sort #recursive call

partition:

#parameters: arr = a0; low = a2; high = a3

#int pivot = arr[high] --> first calculate address

sll $t3, $a3, 2 # (high \* 4) to get word address

add $t3, $a0, $t3 # combine word address with base address to get actual

lw $t5, 0($t3) # get the actual pivot value

addi $t6, $a2, -1 #int i = (low - 1)

#for(int j = low; j < high; j++)

move $t7, $a2 #j = low ==> t7

for:

bge $t7, $a3, end\_for #break if j is greater than or equal to high

#if(arr[j] <= pivot) --> first get arr[j]

sll $t3, $t7, 2 # $t3 = (j \* 4)

add $t3, $a0, $t3 #actual address

lw $t4, 0($t3) # $t4 = arr[j]

bgt $t4, $t5, end\_if #leave if arr[j] > pivot

addi $t6, $t6, 1 #i++

#swap arr[i] and arr[j]

sll $t1, $t6, 2 #$t1 = (i \* 4)

add $t1, $a0, $t1 #actual adddress

lw $t2, 0($t1) #value of arr[i]

sll $t8, $t7, 2

add $t8, $a0, $t8

lw $t9, 0($t8) #value of arr[j]

sw $t2, 0($t8) #arr[i] = arr[j]

sw $t9, 0($t1) #arr[j] = temp (arr[i])

end\_if:

addi $t7, $t7, 1 #j++

j for #go back to start of for loop

end\_for:

#swap arr[i+1] and arr[high]

addi $t1, $t6, 1 # $t1 = i + 1

sll $t1, $t1, 2 # (i + 1) \* 4 => address value

add $t1, $a0, $t1 # actual address

lw $t2, 0($t1) # $t2 = value of arr[i+1]

sll $t8, $a3, 2 # $t8 = (high \* 4)

add $t8, $a0, $t8

lw $t9, 0($t8) # the value of arr[high]

sw $t2, 0($t8) #arr[i+1] = arr[high]

sw $t9, 0($t1) #arr[high] = temp (arr[i+1])

#return i + 1 ==> register $s4

addi $s4, $t6, 1 #s4 = i + 1

jr $ra #return statement

done:

# Caller RTE restore (TBD)

#registers: a0, a1, a2, a3; s3, s4

lw $fp, 36($sp)

lw $ra, 32($sp)

lw $a0, 28($sp)

lw $a1, 24($sp)

lw $a2, 20($sp)

lw $a3, 16($sp)

lw $s4, 12($sp)

lw $s3, 8($sp)

addi $fp, $sp, 36

# Return to Caller

jr $ra

\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Testing:

------------------------------------------

Array is UNSORTED

23 45 8 7 5 9 5 6 89 61

44 4 7 8 15 13 24 35 46 87

Array is SORTED

4 5 5 6 7 7 8 8 9 13

15 23 24 35 44 45 46 61 87 89

------------------------------------------

Array is UNSORTED

45 23

Array is SORTED

23 45

------------------------------------------

Array is SORTED

45

Array is SORTED

45

------------------------------------------

Array is SORTED

23 45

Array is SORTED

23 45

------------------------------------------

Array is SORTED

10 20 30 40 50 60 70 80 90 100

Array is SORTED

10 20 30 40 50 60 70 80 90 100

------------------------------------------

Array is UNSORTED

10 9 8 7 6 5 4 3 2 1

Array is UNSORTED

1 9 8 7 6 5 4 3 2 10

------------------------------------------

Array is UNSORTED

10 9 8 7 7 8 9 10

Array is SORTED

7 7 8 8 9 9 10 10

------------------------------------------

Array is UNSORTED

50 55 5 10 40 15 20

Array is SORTED

5 10 15 20 40 50 55

------------------------------------------

Array is UNSORTED

41 14 15 16 16 17 87 77

Array is SORTED

14 15 16 16 17 41 77 87

------------------------------------------

Array is SORTED

1 1 1 1 1 1 1 1 1 1

Array is SORTED

1 1 1 1 1 1 1 1 1 1

------------------------------------------

\*\*\*\* FAILED [9/10] \*\*\*\*

-- program is finished running --