HW#6 (CSC390); SOLUTION

#Q1. Consider the following two C procedures (swap and sort) and their corresponding MIPS assembly codes as shown in the figures 1 and 2, respectively. Using these two C procedures (swap and sort) write a MIPS assembly program that will sort the following array elements in the **ascending order**.
What changes would you make to sort the array elements in **descending order**?

Array= [100 50 75 -1 -50 500 20 40 40 17 19 23 5 7 -20]

```
# HW#6-Q1 Bubble Sort Algorithm
# Sort the data in an array ascending order
# Array= [100 50 75 -1 -50 500 20 40 40 17 19 23 5 7 -20]

.data

Array: .word 100,50,75,-1,-50,500,20,40,40,17,19,23,5,7,-20
.space 4
n: .word 15 # Number of elements in the Array


.text
la $a0, Array #load the base address of Array into the parameter register $a0
lw $a1, n # load no. of elements into the parameter register $a1

jal sort # Call the procedure
j END

sort:
      addi $sp, $sp, -20    # make room on stack for 5 registers
      sw $ra, 16($sp)       # save $ra on stack
      sw $s3,12($sp)        # save $s3 on stack
      sw $s2, 8($sp)        # save $s2 on stack
      sw $s1, 4($sp)        # save $s1 on stack
      sw $s0, 0($sp)        # save $s0 on stack

  # procedure body

      move $s2, $a0         # save $a0 into $s2
      move $s3, $a1         # save $a1 into $s3
      move $s0, $zero       # i = 0
for1tst: slt  $t0, $s0, $s3     # $t0 = 0 if $s0 ? $s3 (i ? n)
      beq  $t0, $zero, exit1  # go to exit1 if $s0 ? $s3 (i ? n)
      addi $s1, $s0, -1     # j = i – 1
for2tst: slti $t0, $s1, 0       # $t0 = 1 if $s1 < 0 (j < 0)
      bne  $t0, $zero, exit2  # go to exit2 if $s1 < 0 (j < 0)
      sll  $t1, $s1, 2      # $t1 = j * 4
      add  $t2, $s2, $t1    # $t2 = v + (j * 4)
```

```
        lw   $t3, 0($t2)       # $t3 = v[j]
        lw   $t4, 4($t2)       # $t4 = v[j + 1]
        slt  $t0, $t4, $t3     # $t0 = 0 if $t4 ? $t3

        #For Descending Order
        #bne  $t0, $zero, exit2  # go to exit2 if $t4 ? $t3

       #For Ascending Order
        beq  $t0, $zero, exit2  # go to exit2 if $t4 ? $t3

        move $a1, $s1          # 2nd param of swap is j
        jal  swap              # call swap procedure
        addi $s1, $s1, -1      # j –= 1
        j    for2tst           # jump to test of inner loop
exit2:   addi $s0, $s0, 1       # i += 1
        j    for1tst           # jump to test of outer loop


        exit1: lw $s0, 0($sp)  # restore $s0 from stack
        lw $s1, 4($sp)        # restore $s1 from stack
        lw $s2, 8($sp)        # restore $s2 from stack
        lw $s3,12($sp)        # restore $s3 from stack
        lw $ra,16($sp)        # restore $ra from stack
        addi $sp,$sp, 20      # restore stack pointer
        jr $ra                # return to calling routine

swap: sll $t1, $a1, 2   # $t1 = k * 4

     add $t1, $a0, $t1 # $t1 = v+(k*4)
                 #   (address of v[k])

     lw $t0, 0($t1)    # $t0 (temp) = v[k]
     lw $t2, 4($t1)    # $t2 = v[k+1]
     sw $t2, 0($t1)    # v[k] = $t2 (v[k+1])
     sw $t0, 4($t1)    # v[k+1] = $t0 (temp)
     jr $ra          # return to calling routine


END:
nop
```

Q2. Given the two object files of procedure A and procedure B, show updated address of the first few instructions of the completed executable file. Ref: pages 127-128 of your text book.

Note that the default starting address of Text segment = $0x00400000_{hex}$, Data Segment = $0x10000000_{hex}$ and \$gp = $0x10008000_{hex}$

SOLUTION:

Executable Header File		
	Text Size	370_{hex}
	Data Size	55_{hex}
Text Segment	Address	Instruction
	0040 0000_{hex}	lw \$a0, 8000_{hex}(\$gp)
	0040 0004_{hex}	lw \$a1, 8004_{hex}(\$gp)
	0040 0008_{hex}	jal 40 0100_{hex}
	….	
	0040 0120 $_{hex}$	lw \$a0, 8030_{hex}(\$gp)
	0040 0124 $_{hex}$	jal 40 0000_{hex}
	0040 0128 $_{hex}$	sw \$a1, 8034_{hex}(\$gp)
	….	…..
Data Segment	Address	
	1000 0000 $_{hex}$	(X1)
	1000 0004 $_{hex}$	(X2)
	…….	….
	1000 0030 $_{hex}$	(Y1)
	1000 0034 $_{hex}$	(Y2)
	……..	….