```
.data
array: .word 8 -3 9 7 2 -5 1 4  # array input
sizeofarray: .word 8 # size of array
.text

j MAIN  # jumping to Main label

INPUT:
addi x2,x2,-16 # allocating 2 spots in stack
sw x20, 8(x2)  # assigning register x20 to second spot in stack
sw x27, 0(x2)  # assigning register x27 to second spot in stack
la x6, array     # psuedo instruction to place array into x6 register
lw x7, sizeofarray  #loading x7 with value inside variable n
add x20, x0, x0 # x20 = x0 + x0
add x28, x0, x5 # x28 = x0 + x5

LOOPINPUT:
bge x20, x7, INPUTDONE # checking to see if x20 is greater than or equal to x7 if true then go to INPUTDONE label
lw x27, 0(x6)  # loading x27 with x6 value
sw x27, 0(x28) # storing value of x28 into x27
addi x6, x6, 4 # x6 = x6 + 4
addi x28, x28, 4 # x28 = x28 + 4
addi x20, x20, 1 # x20 = x20 + 1
jal x0, LOOPINPUT # jumping into LOOPINPUT

INPUTDONE:
lw x20, 0(x2) # loading x20 into first stack spot
lw x27, 0(x2) # loading x27 into second stack spot
jalr x0, 0(x1) # leaves to MAIN

REORDER:
addi x2, x2, -40 # x2 = x2 + (-40)
sw x27, 24(x2)   # x27 stores in 4th spot in stack
sw x8, 16(x2)    # x8 stores in 3rd spot in stack
sw x9, 8(x2)     # x9 stores in 2nd spot in stack
sw x1, 0(x2)     # x1 stores in 1st spot in stack
add x9, x9, x7   # x9 = x9 + x7
addi x8, x8, 2   # x8 = x8 + 2
div x9, x9, x8   # x9 = x9/x8
add x21, x0, x0  # x21 = x0 + x0
addi x31, x0, 4  # x31 = x0 + 4
add x27, x27, x5 # x27 = x27 + x5
sub x28, x28, x31 # x28 = x28 + x31

LOOPREORDER:
bge x21, x9, EXIT # checking to see if x21 is greater than or equal to x9 if true then go to EXITALL label
jal x1, SWAP # jumping to SWAP label
lw x1, 0(x2) # loading x1 into first stack spot
addi x21, x21, 1 # x21 = x21 + 1
addi x27, x27, 4 # x27 = x27 + 4
sub x28, x28, x31 # x28 = x28 - x31
jal x0, LOOPREORDER # jumping to LOOPREORDER label

EXIT:
lw x27, 24(x2) # loading x27 into 4th stack spot
lw x8, 16(x2)  # loading x8 into 3rd stack spot
lw x9, 8(x2)   # loading x9 into 2nd stack spot
lw x1, 0(x2)   # loading x1 into 1st stack spot
addi x2, x2, 8 # x2 = x2 + 8
jalr x0, 0(x1) # leaving to MAIN

SWAP:
addi x2, x2, -24 # x2 = x2 + (-24)
sw x14, 16(x2)   # x14 is stored in 3rd stack spot
sw x17, 8(x2)    # x17 is stored in 2nd stack spot
sw x1, 0(x2)     # x1 is stored in 1st stack spot
lw x14, 0(x27)   # loading x14 into x27
lw x17, 0(x28)   # loading x17 into x28
sw x14, 0(x28)   # storing x14 into x28
sw x17, 0(x27)   # storing x17 into x27
lw x14, 16(x2)   # loading x14 into 3rd stack spot
lw x17, 8(x2)    # loading x17 into 2nd stack spot
lw x1, 0(x2)     # loading x1 into 1st stack spot
addi x2, x2, 24  # x2 = x2 + 24
jalr x0, 0(x1)   # leaving to MAIN

OUTPUT:
addi x2, x2, -32 # x2 = x2 + (-32)
sw x20, 24(x2)   # x20 is stored in 4th stack spot
sw x10, 16(x2)   # x10 is stored in 3rd stack spot
sw x11, 8(x2)    # x11 is stored in 2nd stack spot
sw x18, 0(x2)    # x18 is stored in 1st stack spot
add x18, x5, x18 # x18 = x5 + x18
addi x10, x0, 1  # x10 = x0 + 1

LOOPOUTPUT:
bge x20, x7, DONEOUTPUT # checking to see if x20 is greater than or equal to x7 if true then go to DONEWITHOUTPUT label
addi x10, x0, 1  # x10 = x0 + 1
lw x11, 0(x18)   # loading x11 into x18
addi x18, x18, 4 # x18 = x18 + 4
ecall  # printing to console
addi x11, x0, 32 # x11 = x0 + 32
addi x10, x0, 11 # x10 = x0 + 11
ecall  # printing to console
addi x20, x20, 1 # x20 = x20 + 1
jal x0, LOOPOUTPUT # jumping to LOOPOUTPUT label

DONEOUTPUT:
lw x20, 24(x2) # x20 is stored in 4th stack spot
lw x10, 16(x2) # x10 is stored in 3rd stack spot
lw x11, 8(x2)  # x11 is stored in 2nd stack spot
lw x18, 0(x2)  # x18 is stored in 1st stack spot
jalr x0, 0(x1) # leaving to MAIN

MAIN:
li x5, 0x0fffffe8  # storing 0x0fffffe8 into x5 by pseudo instruction into memory
jal x1, INPUT      # jumping to INPUT label
jal x1, REORDER    # jumping to REORDER label
jal x1, OUTPUT     # jumping to OUTPUT label
```