```
.extern csd_main

.align 8

// Our interrupt vector table
csd_entry:
        b csd_reset
        b .
        b .
        b .
        b .
        b .
        b csd_irq
        b .

.global main

csd_reset:

main:

        // sorting algorithm: selection sort (compare each value and select the
smallest)

        ldr r0, =Input_data // store Input_data memory address in r0
        ldr r1, =Output_data // store Output_data memory address in r1
        mov r2, #-4 // r2 stands for start position

first_loop:

        add r2, r2, #4 // assign r2 the first start position
        mov r3, r2 // r3 stands for the position of the smallest value
        mov r4, r3 // r4 stands for compared position

second_loop:

        add r4, r4, #4 // r4 stands for the next position of r2
        ldr r5, [r0, r3] // load the word which needs to be compared into r5
        ldr r6, [r0, r4] // load the next word which needs to be compared into r6

        cmp r5, r6 // compare

        movgt r3, r4
        // if r5 is greater than r6 (which means r6 is smaller) move value of r4 to
r3
        // it means the smallest value has been changed

        cmp r4, #124 // check the remaining number of comparison in
second_loop
        bne second_loop // if times are remaining, go to second_loop label and
continue comparison

    // if second_loop ends then store the smallest value into Output_data

        ldr r7, [r0, r3] // load the smallest word into r7
        str r7, [r1, r2] // store the word to Output_data

        // also move the value (which is not the smallest) to Input_data at empty
position

        ldr r7, [r0, r2] // load the not-smallest value into r7
        str r7, [r0, r3] // fill the empty place in Input_data (which was filled with
the smallest)

        cmp r2, #120 // check the remaining number of comparison in first_loop
```

```
        bne first_loop // if times are remaining, go to first_loop lable and continue switching

        // after escaping from the first_loop, only one value (the biggest) left in Input_data

        ldr r7, [r0, #124] // load the biggest value into r7
        str r7, [r1, #124] // fill the list completely (store the biggest value to the list)

        b csd_main // go to csd_main which leads to program ends

.data
.align 4

Input_data: .word 2, 0, -7, -1, 3, 8, -4, 10
                  .word -9, -16, 15, 13, 1, 4, -3, 14
                  .word -8, -10, -15, 6, -13, -5, 9, 12
                  .word -11, -14, -6, 11, 5, 7, -2, -12

Output_data: .word 0, 0, 0, 0, 0, 0, 0, 0
                   .word 0, 0, 0, 0, 0, 0, 0, 0
                   .word 0, 0, 0, 0, 0, 0, 0, 0
                   .word 0, 0, 0, 0, 0, 0, 0, 0

// Normal Interrupt Service Routine
csd_irq:
        b .
```