# Laboratory 8

Title of the Laboratory Exercise: Sorting

1. Introduction and Purpose of Experiment

Students will create assembly code with sorting techniques and nested loops

1. Aim and Objectives

Aim

To develop assembly language program to perform sorting using nested loop structures

Objectives

At the end of this lab, the student will be able to

* + use nested loops in assembly
  + perform sorting in ascending/ descending order
  + Build complex looping logic in assembly language

1. Experimental Procedure

1. Write algorithm to solve the given problem

2. Translate the algorithm to assembly language code

3. Run the assembly code in GNU assembler

4. Create laboratory report documenting the work

1. Questions

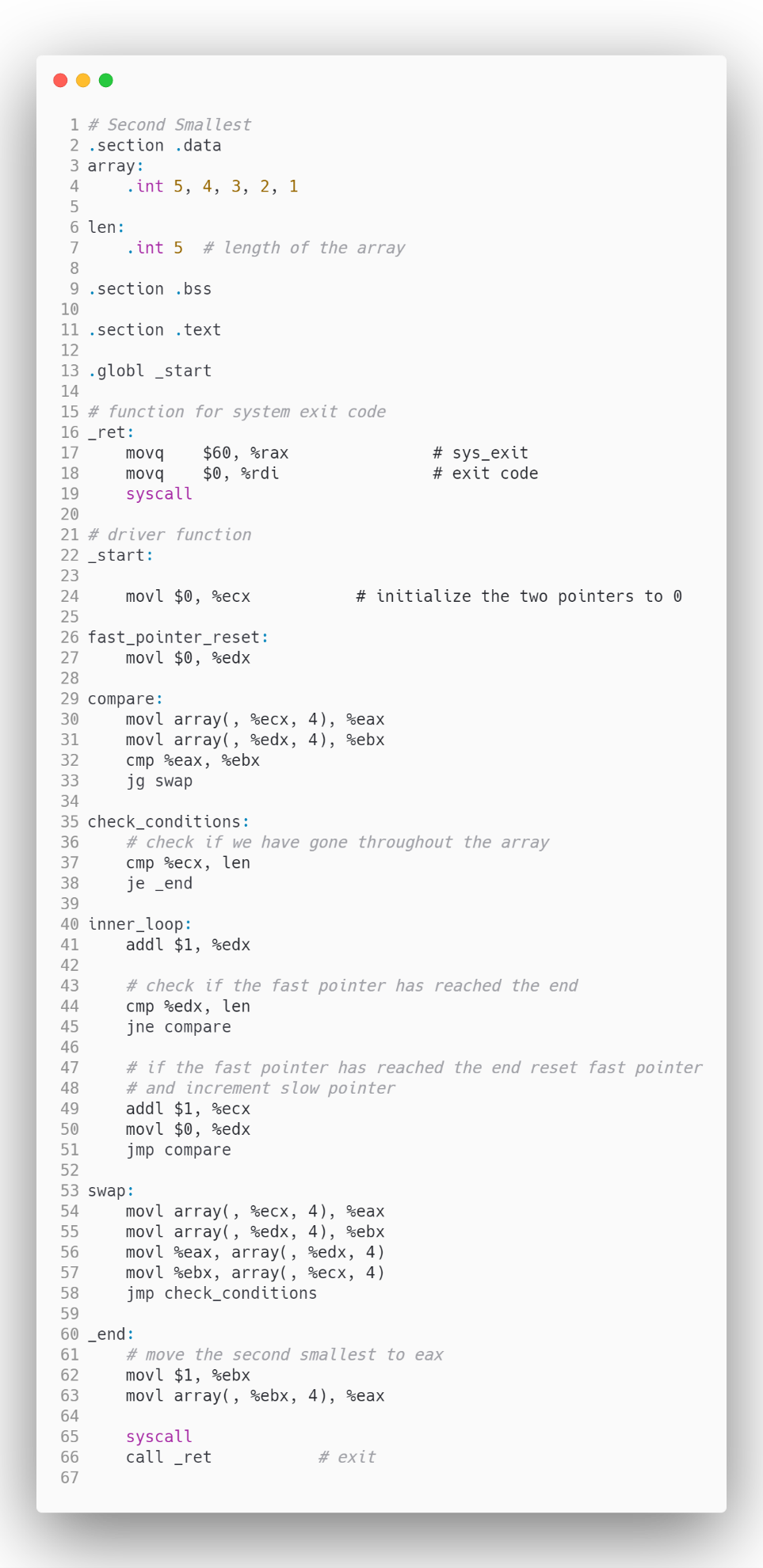
Develop an assembly language program to perform the following

* + 1. Arrange an array of ’n’ numbers in ascending order
    2. Arrange an array of ’n’ numbers in descending order
    3. Determine the second smallest number in an array.

1. Calculations/Computations/Algorithms







1. Presentation of Results

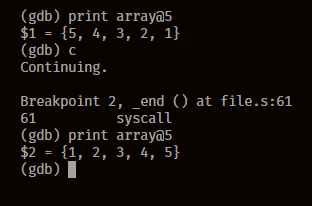


Figure 0‑1 Sort Array in Ascending Order

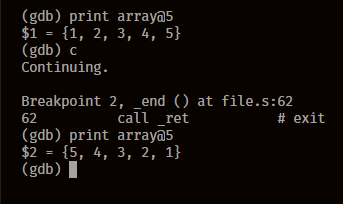


Figure 0‑2 Sort Array in Descending Order

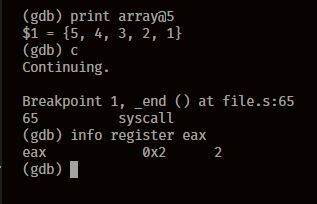


Figure 0‑3 Find Second Smallest Element

1. Analysis and Discussions

The Array here is sorted using various conditional statements by combination of compare and jump instruction, to sort them we are using insertion sort algorithm, but a little unoptimized version of it, we keep on swapping the current element with every element that is smaller than it, we keep on doing this until we reach the end of the array, the time complexity for such a sorting algorithm is .

1. Conclusions

Arrays can be sorted by using a combination of jump, compare and labels. These have to be carefully designed as to avoid infinite loops and array index out of bounds errors, which causes segmentation faults.

1. Comments

1. Limitations of Experiments

Complex Sorting Algorithms such as Tim Sort, Radix Sort, Optimized Quick Sort, are very complex to implement in Assembly although they would provide performance benefits.

2. Limitations of Results

The code written for sorting the elements is unoptimized and would perform worse when the data given to it is in huge amount.

3. Learning happened

Elements of an array can be sorted in Assembly, though with quite a lot of written code.

4. Recommendations

The loop statements should be carefully written to avoid infinite loops.

Signature and date Marks