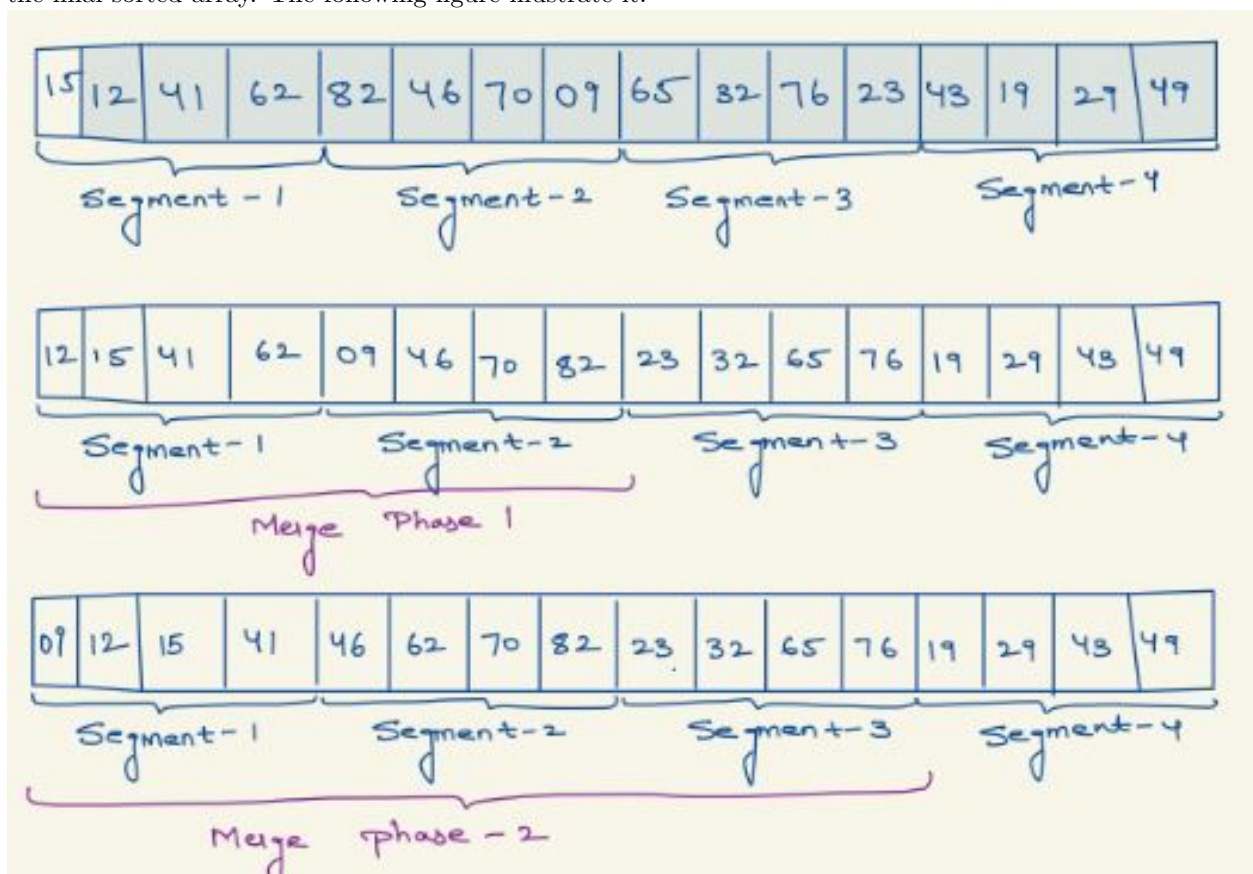# Operating System-2
Assignment 1
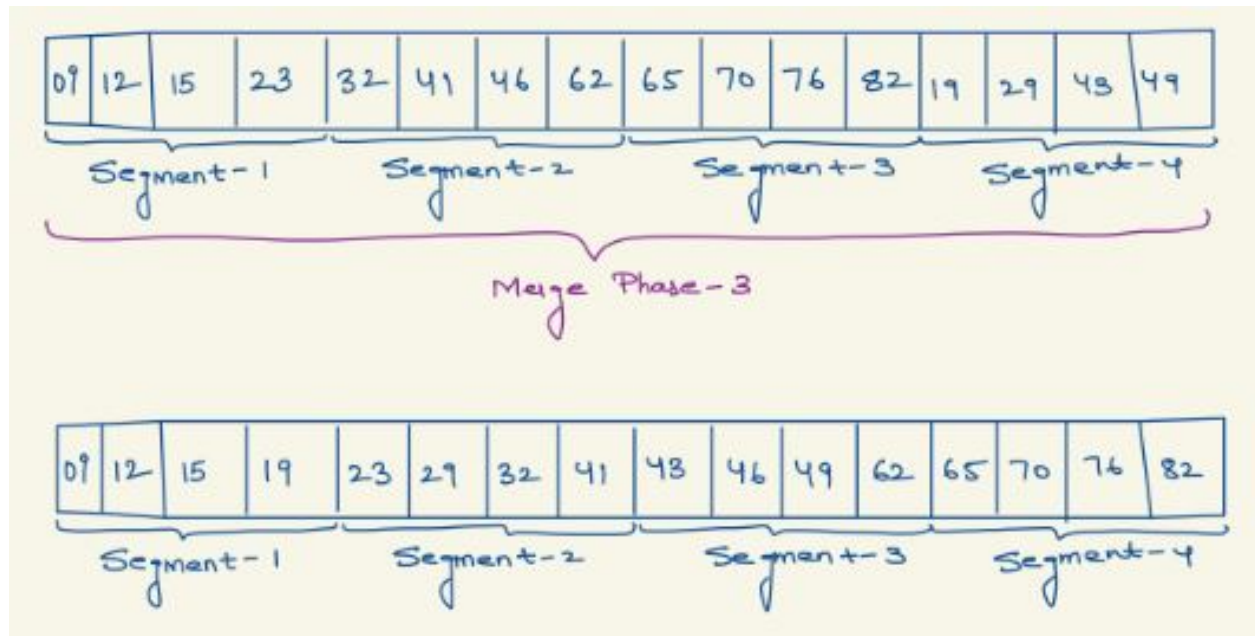
Parallel Sorting using Multi Threading

Submission Date: 27th January 2021, 9:00 PM

Objective: The objective of this assignment is to perform sorting of numbers efficiently using multiple threads.

Sorting Technique: In this assignment, you will experiment with multi-threading the performance impact using real-time measurements. Given an array of size 2^n of double values. You have to divide the array into 2^p segments of equal size, 2^{p-n}. Sort each segment using a sorting algorithm S of your choice using 2^p threads. You must then merge each of the 2^p segments. You can perform the merge using one of the following methods:
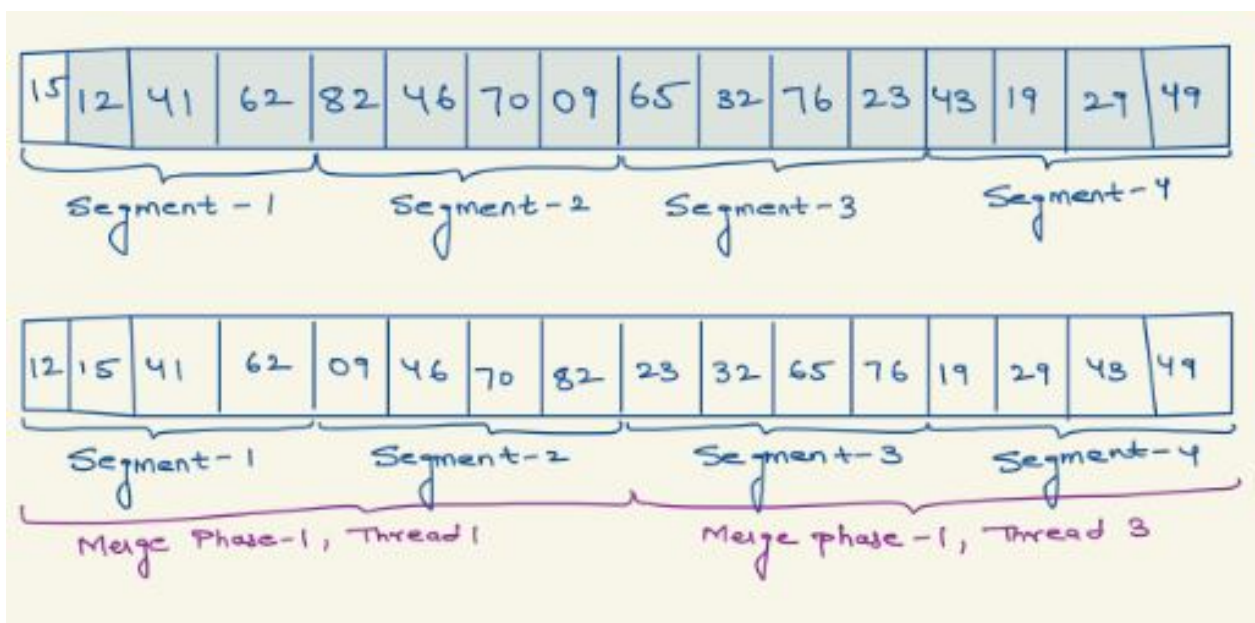
Method 1: In this method once all the threads have performed the sorting and created the sorted segment, the main thread will perform the merge operation among the different segments one by one which results in the final sorted array. The following figure illustrate it.
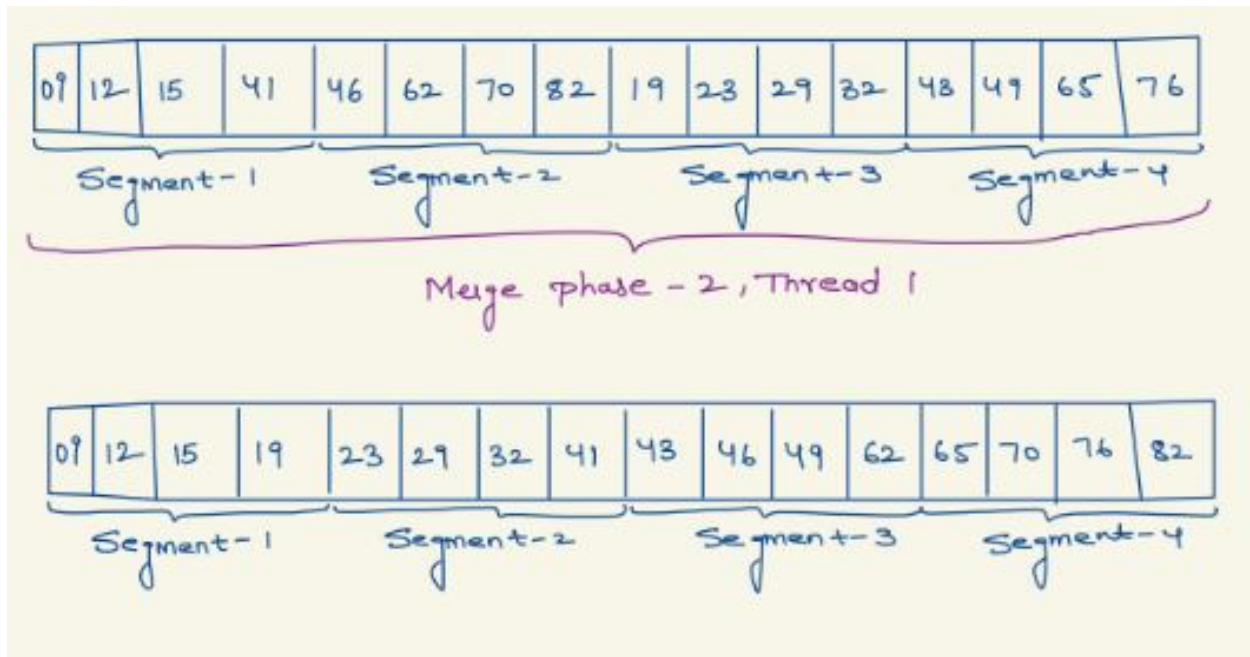
| 09 | 12 | 15 | 23 | 32 | 41 | 46 | 62 | 65 | 70 | 76 | 82 | 19 | 29 | 43 | 49 |

Segment-1 Segment-2 Segment-3 Segment-4

Merge Phase-3

| 09 | 12 | 15 | 19 | 23 | 29 | 32 | 41 | 43 | 46 | 49 | 62 | 65 | 70 | 76 | 82 |

Segment-1 Segment-2 Segment-3 Segment-4

Method 2: In this method once all the threads have performed the sorting then half of the total number of threads will perform the merging of the segments while the remaining half exit. This merging occurs in phases while half total number of threads exit at the end of each phase. These steps will be repeated till the array is completely sorted.

To precisely identify the merging thread among two threads, we choose the thread with smaller id to perform the merge step while the other thread will terminate. Consider the following example: In first phase for merging of segment 1 and 2 thread 1 will perform the merging operation while thread 3 will perform the merging of segments 3 and 4. Again thread 1 will perform merging of segment 1,2 and 3,4. This is illustrated using the figures below.

| 15 | 12 | 41 | 62 | 82 | 46 | 70 | 09 | 65 | 32 | 76 | 23 | 43 | 19 | 27 | 49 |

Segment – 1 Segment-2 Segment-3 Segment-4

| 12 | 15 | 41 | 62 | 09 | 46 | 70 | 82 | 23 | 32 | 65 | 76 | 19 | 29 | 43 | 49 |

Segment-1 Segment-2 Segment-3 Segment-4

Merge Phase-1, Thread 1   Merge phase-1, Thread 3

2

Input: The input to the program will be a file (named inp.txt) where the first digit will be n where size of the array is 2^n. Second digit in the file will be parameter p where the no. of segments are 2^p which is also the number of threads.

Example: inp.txt

5 2   // Size of Array 2^5, No.of threads 2^2

Output: Output.txt should contain the initial array before sorting and Array after sorting in the next line. Time taken in microseconds for sorting in third line.

Example: output.txt

8 15 22 13 16 27   // Array Before sorting
8 13 15 16 22 27   // Array After sorting
Time taken: 563 Microseconds.

Report: As a part of this assignment, you have to prepare a report which will describe the low-level design of your program and give an analysis of its output. As a part of the report, you have to measure the performance of the algorithms by following the mentioned steps. Calculate the time taken by method 1 and 2. Don't include the timing of reading and writing to file in this. Create a graph showing the time taken by these solutions for different values of the number of threads and fixing the size of the array to 2^8. The x-axis should be the parameter p 2, 3, 4, 5, 6 where the number of segments and threads are equal to 2^p. The y-axis will include time taken by the program to terminate (in micro/milliseconds).

Next, create another graph showing the times using both the methods while fixing the p to 4, i.e., the number of segments and threads to be 16. The x-axis should be the parameter n, the array size, i.e. 5, 6, 7, 8, 9. Report should include the detailed analysis of the graph.

Deliverables:

1. Report describing the low-level design of your program and analysis of output and graph. This file should be named as Asgn1_ <Roll No.> Report.pdf

2. Prepare a README file that contains the instructions on how to execute your submitted file. The file should be named as README.txt

3. Name the source code file in the following format: Asgn1_sol1_ <Roll No.>.c and Asgn1_sol2_ <Roll No.>.c

4. Upload the source code, report and README as a zip archive file and name it as Asgn1_ <Roll No.>.zip

Please see the instructions given above before uploading your file. Your assignment will NOT be evaluated if there is any deviation from the instructions posted there.

The policy for grading this assignment will be

- Design as described in report and analysis of the obtained results: 50%

- Execution: 40%

- Code documentation and indentation: 10%.

As mentioned before, all assignments for this course has the late submission policy of a penalty of 10% each day after the deadline.

Kindly remember that all submissions are subjected to plagiarism checks.