CA ASSIGNMENT – 2

MIPS ARCHITECTURE

- IMT2020106 Chandra Karthik D

- IMT2020080 Vamshi Krishna G

The sorting of the N inputs is done by Bubble sort algorithm.

Input was given in the following format:

* The number of elements in the data to be sorted (or) N is given as input in the first line.
* The memory location where the input data is to be stored is given as input in the next line.
* The memory location where the output data after sorting is to be stored is given as input

in the next line.

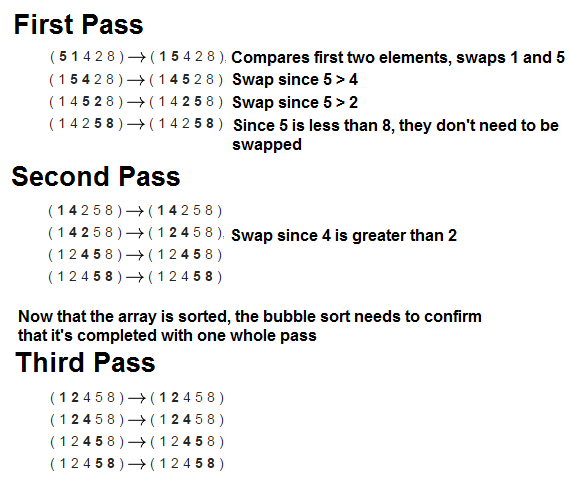
* Then the N elements that are to be sorted is given as input in the next N lines.

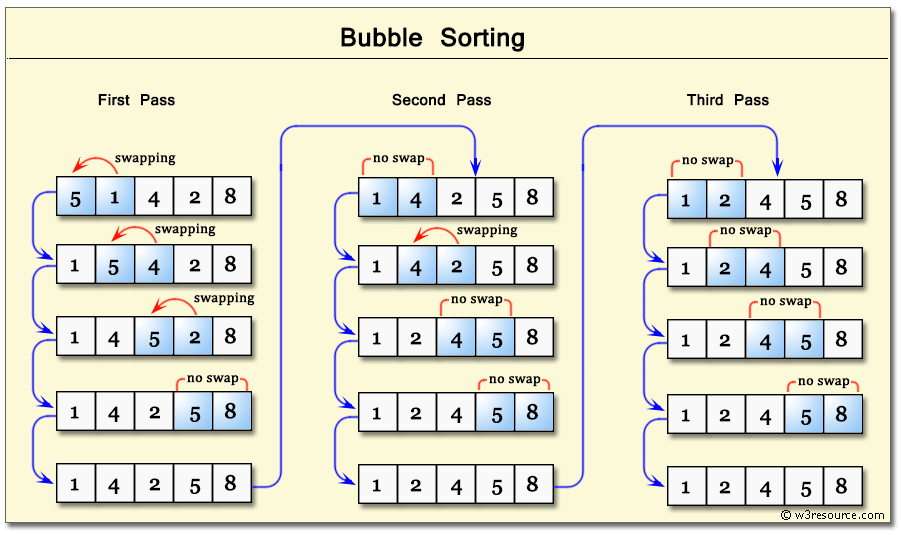
Bubble sort

Bubble sort is a basic algorithm for arranging a list or array of numbers in the correct order. The method works by examining each set of adjacent elements in the string, from left to right, switching their positions if they are out of order. The algorithm then repeats this process until it can run through the entire list or array and find no two elements that need to be swapped.

Example of Bubble Sort Implementation

Data to be sorted: 5, 1, 4, 2, 8





C Program for Bubble Sort Algorithm

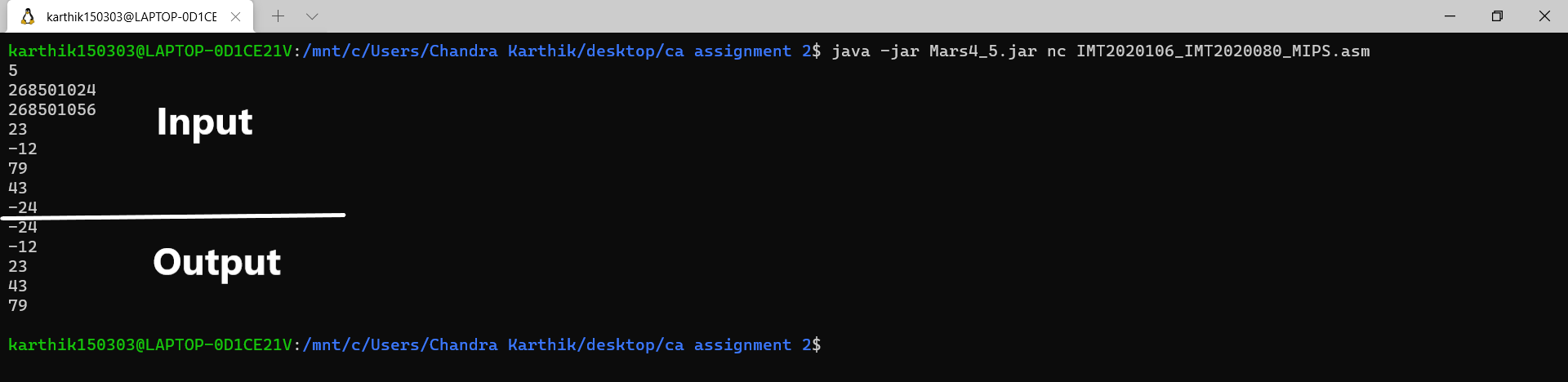
|  |
| --- |
| void swap (int \*i, int \*j)  {  int t = \*i;  \*i = \*j;  \*j = t;  }  void bubbleSort (int array[], int n)  {  int a, b;  for (a = 0; a < n-1; a++) {  for (b = 0; b < n-i-1; b++) {  if (array[b] > array[b+1]) swap(&array[b], &array[b+1]);  }  }  } |

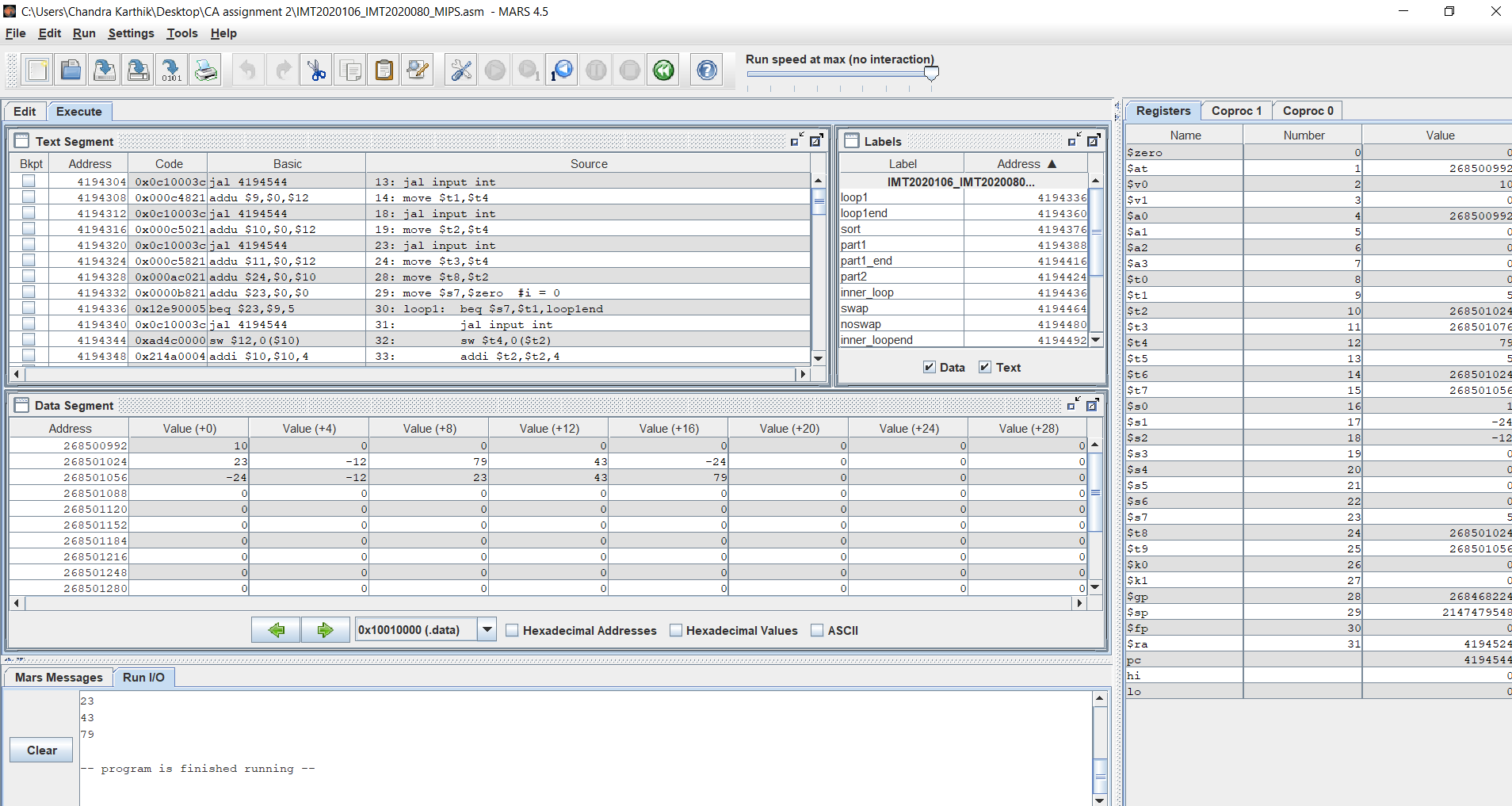
EXPLANATION

* The data stored in $t1, $t2, $t3 are stored in $t5, $t6, $t7 respectively so that the sort function could be implemented without the direct usage of registers $t1, $t2 and $t3.
* Now sorting is done using a function of the name “sort”.
* All the numbers starting from the address given in $t6 (i.e., Address in $t2) are copied into consecutive memory locations starting from address in $t7 (i.e., Address in $t3). This is done using a loop named “part1”.
* The value in $s0 acts as a loop counter for the “part1” loop which increments by 1 for every loop iteration.
* After the loop “part1” is executed, the copied data stored at the address in $t7 (i.e., Address in $t3) is to be sorted. So, to sort this another loop “part2” is used.
* Initialize the value in $s4 to N (i.e., the number of elements in the data) which is used as loop counter for the loop “part2” which decrements by 1 for every loop iteration.
* In the loop “part2” the loop executes until value in $s4 becomes equal to 0 after decrementing by 1 in each iteration. Since the initial value in $s4 is N. there are N iterations of loop “part2”.
* Now inside the “part2” loop another loop is used known as the “inner\_loop”. Copy the address in $t7 (i.e., Address in $t3) into $t9 before “inner\_loop” is executed, so that the value in $t9 could be altered in every iteration of the “inner\_loop” and could be reinitialized to its initial value after each iteration of the outer loop “part2”.
* Also initialize another value in $s0 as 1 so that it could be used as a loop counter for “inner\_loop” which increments by 1 for every iteration of “inner\_loop”.
* In the inner loop 2 values (values in 0($t9) and 4($t9)) are compared in every iteration, in which if the value stored in 4($t9) is lower than value stored in 0($t9), the values in both the addresses are swapped. After one complete execution of an “inner\_loop” in one iteration of outer loop “part2”, the greatest element of the all the pairs of elements that are compared in the “inner\_loop” is stored in the greatest(rightmost) address. Now the ignoring that element the other elements are compared and greatest of them is sorted just to the left of the element which is ignored in previous iteration of outer loop. This is continued till all elements are sorted (i.e., till the outer loop “part2” is completely executed).
* A more detailed explanation is given in the form of comments in the .asm file.

Test results

Test result1:





Test result2:

