Assignment #12 - Recursive Functions **CS 218 Fall 2021**

Purpose: Write a MIPS program that will implement a mergeSort.

A template file has been included to base your program on.

Instructions

Using the random number function from Assignment #11, fill the array with random values from 1 to 100.

Print out the unsorted array. Print out a new line after every 10 elements.

Use the merge sort function (described below) to sort the array.

Print out the sorted array.

Print out the number of merges required.

Merge Sort

The merge sort is a recursive algorithm that divides an array into two sub-arrays that are half the size of the original. Each sub-array is merge sorted recursively and then the two sorted sub-arrays will need to be merged together into a single sorted array.

The base (stopping) case for the merge sort is when the array being sorted has only a single value, in this case nothing need be done and the function can just return. The base case does **not** count as a merge.

Otherwise, the function will need to as evenly as possible split the array into two halves, calling merge sort on each half.

```
    SubArray1 Size = Array Size / 2
    SubArray2 Size = Array Size - SubArray1 Size
    SubArray1 Address = Array Address
    SubArray2 Address = Array Address + (SubArray1 Size*4)
```

To merge the two sub-arrays, we will need to compare the values at the start of each array. Whichever sub-array has the lower value will be pushed to the stack. Then we will compare the next value from that sub-array with the other. This process should repeat until one of the sub-arrays runs out of elements. The remaining sub-array should then be pushed.

Since the values on the stack are in reverse order, they will need to be popped back into the array starting from the end and working towards the front.

The function should return the number of merges which will be 0 for the base case or 1 + merges from the lower half + merges from the upper half.

Submission

Once completed, upload the MIPS assembly source code file (.asm) to the class website.

Example Execution

```
Unsorted:
50 50 6 45 84 20 64 90 73 2
43 37 33 57 79 45 1 83 35 26
39 34 36 76 75 93 85 5 40 94
43 8 1 42 19 33 39 87 74 100
83 41 38 41 72 2 4 43 5 42
52 99 78 17 80 31 81 40 1 69
50 51 53 76 19 1 33 48 41 81
3 61 25 28 2 48 42 19 37 11
64 75 97 43 23 48 82 93 24 84
71 93 18 58 10 62 43 57 76 86

Sorted:
1 1 1 1 2 2 2 3 4 5
5 6 8 10 11 17 18 19 19 19
20 23 24 25 26 28 31 33 33 33
34 35 36 37 37 38 39 39 40 40
41 41 41 42 42 42 43 43 43 43
43 45 45 48 48 48 50 50 50 51
52 53 57 57 58 61 62 64 64 69
71 72 73 74 75 75 76 76 76 78
79 80 81 81 82 83 83 84 84 85
86 87 90 93 93 93 94 97 99 100

Merges: 99
```