COE 301 / ICS 233 – Computer Organization

Merge Sort Recursive Function

The function **merge_sort** sorts an array of **n** integers recursively. It calls a function **merge** that merges the upper half with **m=n/2** elements with the lower half that has the remaining **(n-m)** elements. Translate this function into MIPS code. Write a main function to allocate dynamically an array of **n** integers (**n** is a user input that must be greater than 1), read an array of **n** integers, print the array before sorting, sort the array, and then print the array after sorting. You need to write functions for reading and printing the array, and pass parameters properly according to the MIPS convention.

```
void merge_sort(int A[], int n) {
  if (n <= 1) return;                 // nothing to sort
  int m = n/2;                        // half the array length
  merge_sort(&A[0], m);               // Sort first half of A[]
  merge_sort(&A[m], n-m);             // Sort second half of A[]
  merge(A, n);                        // Merge the two sorted halves
}

void merge(int A[], int n) {
  int B[n];                           // Allocate B[] on the stack
  int m = n/2;                        // half the array length
  int i = 0;                          // i = index to 1st half of A[]
  int j = m;                          // j = index to 2nd half of A[]
  int k = 0;                          // k = index to B[]
  // Merge the two halves of array A[]
  // Put the merged values into local array B[]
  while (i<m && j<n) {
    if (A[i] <= A[j]) B[k++] = A[i++] // Copy element from 1st half
    else B[k++] = A[j++];             // Copy element from 2nd half
  }
  // Copy the remaining elements in the first half of A[]
  while (i<m) B[k++] = A[i++];
  // Copy the k merged element in B[] back into A[]
  for (i=0; i<k; i++) A[i] = B[i];
}
```

You need two additional functions to read and print an integer array **A[]** of **n** integers:

```
void read_array (int A[], int n) { . . . }
void print_array (int A[], int n) { . . . }
```

Submission Guidelines:

This assignment can be solved individually or in groups of two students only. No group should have more than two students. Both students should contribute to the solution. **At the beginning of your program, write the names of the two students who worked on the program.** If this program was solved individually then write your name only. The rest of the code should be well written and well document.

All submissions should be done through Blackboard. Submit the source code of the program. Make sure that your program is well written and documented. The program will be graded according to its correctness and documentation. It is your responsibility to make sure that the program works. A program that does not assemble or run will receive zero on correctness.

Grading Scheme:

Dividing the program into procedures and passing parameters properly	**[3 points]**
Allocating an array of *n* integers dynamically	**[1 point]**
Reading the array	**[2 points]**
Sorting the array properly using recursive **`merge_sort`**	**[10 points]**
Printing the array before and after sorting	**[3 points]**
Program readability and comments	**[1 point]**