COE 205, Term 071

Computer Organization & Assembly Programming

Programming Assignment# 3
Due date: Monday, Dec. 10, 2007

Q.1. Quicksort is a sorting algorithm whose worst-case running time is $O(n^2)$ on an input array of n numbers. In spite of this slow worst-case running time, quicksort is often the best practical choice for sorting because it is remarkably efficient on the average: its expected running time is O(n lg n). Quicksort is based on divide-and-conquer paradigm. The three-step divide-and-conquer process for sorting a typical array A[p...r] is as follows:

1. **Divide**: The array A[p..r] is partitioned (rearranged) into two nonempty subarrays A[p..q] and A[q+1..r] such that each element of A[p..q] is less than or equal to each element of A[q+1..r]. The index q is computed as part of this partitioning procedure.
2. **Conquer**: The two subarrays A[p..q] and A[q+1..r] are sorted by recursive calls to quicksort.
3. **Combine**: Since the subarrays are sorted in place, no work is needed to combine them: the entire array A[p..r] is now sorted.

The following procedure implements quicksort:

```
QUICKSORT(A, p, r)
1. if ( p < r) Then
2.    q ← PARTITION(A, p , r)
3.   QUICKSORT(A,  p , q)
4.   QUCIKSORT(A, q+1, r)
```

To sort an entire array A, the initial call is QUICKSORT(A, 1, length[A]).
The partition procedure is defined as follows:

```
PARTITION(A, p, r)
1.  x ← A[p]
2.  i ←  p-1
3.  j ←  r+1
4.  While TRUE
5.       Repeat
6.           j ←  j-1
7.       Until A[j] ≤ x
8.        Repeat
9.              i ←  i+1
10.      Until A[i] ≥ x
11.      If (i < j) Then
12.          exchange A[i] ↔A[j]
13.      Else
14.         return j
```

15. End While

(i) Write a procedure, PARTITION, to implement the partition procedure given above. All input parameters are to be passed on the stack.

(ii) Write a procedure, QUICKSORT, to implement the quicksort procedure given above. All input parameters are to be passed on the stack.

(iii) Ask the user to enter how many numbers need to be sorted, N.

(iv) Ask the user to enter the maximum number range, M.

(v) Generate N random numbers, in the range 0 to M-1, and store them in an array, IntArray.

(vi) Use the QUICKSORT procedure you implemented to sort the array, IntArray.

(vii) Display the array, IntArray, before sorting and after sorting.

(viii) Display the execution time of sorting in seconds.

A sample execution of the program is shown below:

Enter the numbers to be sorted:5
Enter the maximum number range:10
Array before sorting is:
2 1 3 5 6
Array after sorting is:
1 2 3 5 6
Sorting time is 1 seconds.

The solution should be well organized and your program should be well documented. Submit a soft copy of your solution in a zip file. Your solution should be submitted in a word file that contains the following items:

i) Your name and ID

ii) Assignment number

iii) Problem statement

iv) Your solution along with the code

v) Discussion of what worked and what did not work in your program. Include snapshots that demonstrate the working parts of your program. If things did not work and you attempted to solve them, mention that and write about the difficulty that you have faced.

The soft copy should also contain both source code file (i.e. .asm) and the executable file (i.e. .exe).