In this paper, we present an algorithm for sorting n numbers in time O(log n) that requires asymptotically fewer processors. We first present parallel bucket-sorting algorithms in which at the expense of greater space requirements, the number of processors used are reduced. An algorithm for sorting n numbers using n (parallelic) processors in time O(log n) has been proposed. It is based on the model of a computer that follows the instructions of a unique instruction stream. In this case, the instructions are linear function of the bits in the binary representation of the processor number.

1)The algorithm is expressed formally below. Each area will be of size n, the number of input numbers to be sorted. Within each area, j, the processors (pi) having ci = j will leave marks indicatingtheir presence. Then, in a binary-tree fashion, they will search for the presence of (the marks of) other active processors. If two processors discover each other's presence, the lower ranking one (i.e. the one with smaller index) will continue while the higher ranking one will deactivate.//add working

A[j, 0] will be the number of different fs such that ci = j. An inactive processor, p~, will keep its count at the head of the largest block which had no other processors. Each of the duplicates has a count of the total number. of c/s that are greater than it plus the numbers that are equal to it. The algorithms given above assume that an area (,4) of memory has been initialized to zero.

Many instances of this algorithm can be executed one after another. For serial programs, one can include at each location a pointer to a backpointer on a stack. Each time an entry is accessed, verification can be made that the contents are not random.

The methods we are about to offer sort n arbitrary numbers in time O. (log n). They are based on an improvement of a Gavril [8] technique that combines two linearly-ordered sets. Each active processor can reinitialize the contents of the location it is now working on after initialising the location its friend is working on (to zero) at each stage of the process. Algorithm 3 will be the first algorithm we utilise, and it will take n 3/2 processors. Do a bucket sort of the items in each sector using the key count[j] as the element.

The items inside each sector will then be rearranged in terms of rank order. The same resources as Batcher's algorithms are used to create an algorithm to sort n numbers in O(log 2 n) time using O(n) processors. These algorithms do have memory-fetch conflicts even though they avoid memory-store conflicts.