Problem Statement:

Write a RISC V assembly program that implements Merge Sort algorithm to sort N integers.

Input to this procedure should be taken from a random function, as shown in the description below. The elements in the array should be stored from 0x10001000, sorted array is stored at address 0x10002000 and the number of elements should be stored at x3 (global pointer register). Seed value, array start address and number of elements in array are written as hard-coded constants in assembly.

Numbers are assumed to be signed integers of 4 byte

Description

You may follow the following pseudo code:

```
#define N 10
Int Array[N];
Int SortedArray[N];

Int seed 0x1000 0000

int random()
{
x = seed;
x ^= (x << 21);
x ^= (x >>> 35); //unsigned right shift
x ^= (x << 4);
seed = x;
return x;
}


/* Code taken from https://www.geeksforgeeks.org/c-program-for-merge-sort/ */

// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
void merge(int arr[], int l,
                    int m, int r)
{
        int i, j, k;
```

```
int n1 = m - l + 1;
int n2 = r - m;

// Create temp arrays
int L[n1], R[n2];

// Copy data to temp arrays
// L[] and R[]
for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

// Merge the temp arrays back
// into arr[l..r]
// Initial index of first subarray
i = 0;

// Initial index of second subarray
j = 0;

// Initial index of merged subarray
k = l;
while (i < n1 && j < n2)
{
        if (L[i] <= R[j])
        {
                arr[k] = L[i];
                i++;
        }
        else
        {
                arr[k] = R[j];
                j++;
        }
        k++;
}

// Copy the remaining elements
// of L[], if there are any
while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
}

// Copy the remaining elements of
// R[], if there are any
```

```
        while (j < n2)
        {
                arr[k] = R[j];
                j++;
                k++;
        }
}

// l is for left index and r is
// right index of the sub-array
// of arr to be sorted
void mergeSort(int arr[],
                        int l, int r)
{
        if (l < r)
        {
                // Same as (l+r)/2, but avoids
                // overflow for large l and h
                int m = l + (r - l) / 2;

                // Sort first and second halves
                mergeSort(arr, l, m);
                mergeSort(arr, m + 1, r);

                merge(arr, l, m, r);
        }
}

main()
{

   for(i=0; i < N; i++) {
      SortedArray[i] =Array[i] = random();
    }

    mergeSort(SortedArray, 0, N - 1);

}
```