Assignment 5
Recursion

Deadline: 25 March 2022 at 23:59 on Submitty

Working individually, complete the assignment below. Submit your solution to Submitty (`https://submit.scss.tcd.ie`). By submitting your solution, you are confirming that you have familiarised yourself with College's policy on plagiarism (`https://libguides.tcd.ie/plagiarism`).

Your mark will be the auto-graded mark assigned by Submitty (10 marks).

You are allowed to submit five attempts for the assignment without penalty. Subsequent attempts will attract a 1 mark penalty each, up to a maximum penalty of 5 marks.

Submitty will allow you eight "late days" over the full semester. This means, for example, you can submit one assignment late by eight days or eight assignments late by one day (or part thereof) each, without penalty. Once your "late days" are used up, you will receive zero marks for any late submissions.

Instructions

Quicksort is a well known sorting algorithm developed by Tony Hoare [1]. In this assignment, you will implement the quicksort algorithm in ARM Assembly Language using *recursion*.

To do this, you will implement three ARM Assembly Language subroutines that implement the interfaces described below. The pseudo-code for two of the subroutines is also provided below.

Develop your solution using the `asmt-5-recursion` template program in the CSU1102x repository. Submit your solution to Submitty (`https://submit.scss.tcd.ie`).

swap subroutine

Interface:

```
@ swap subroutine
@ Swap the elements at two specified indices in an array of words.
@
@ Parameters:
@   R0: array — start address of an array of words
@   R1: a — index of first element to be swapped
@   R2: b — index of second element to be swapped
@
@ Return:
@   none
```

partition subroutine

Interface:

```
@ partition subroutine
@ Partition an array of words into two parts such that all elements before some
@   element in the array that is chosen as a 'pivot' are less than the pivot
@   and all elements after the pivot are greater than the pivot.
@
@ Based on Lomuto's partition scheme (https://en.wikipedia.org/wiki/Quicksort)
@
@ Parameters:
@   R0: array start address
@   R1: lo index of partition to sort
@   R2: hi index of partition to sort
@
@ Return:
@   R0: pivot - the index of the chosen pivot value
```

Pseudo-code:

```
int partition (array, lo, hi) {
    pivot = array[hi];
    i = lo;
    for (j = lo; j <= hi; j++) {
      if (array[j] < pivot) {
        swap (array, i, j);
        i = i + 1;
      }
    }
    swap(array, i, hi);
    return i;
}
```

quicksort subroutine

Interface:

```
@ quicksort subroutine
@ Sort an array of words using Hoare's quicksort algorithm
@ https://en.wikipedia.org/wiki/Quicksort
@
@ Parameters:
@   R0: Array start address
@   R1: lo index of portion of array to sort
@   R2: hi index of portion of array to sort
@
@ Return:
@   none
```

Pseudo-code:

```
int quicksort (array, lo, hi) {
    if (lo < hi) { // !!! You must use signed comparison
                   // (e.g. BGE) here !!!
        p = partition(array, lo, hi);
        quicksort(array, lo, p - 1); // call's itself - recursion!
        quicksort(array, p + 1, hi); // call's itself - recursion!
    }
}
```

Hints

Write the subroutines one at a time and test each one before moving on to the next one. You can modify the program at the label `Main` in test.s to test each subroutine.

When implementing the `quicksort` subroutine, use a signed comparison (e.g. BGE rather than BHS) for the `if` statement. This is necessary because `hi` can be a negative value (-1).

Submitty will test each of your subroutines separately so, if you cannot get `quicksort` to work, you can still get marks for swap and partition.

References

[1] `https://en.wikipedia.org/wiki/Quicksort`