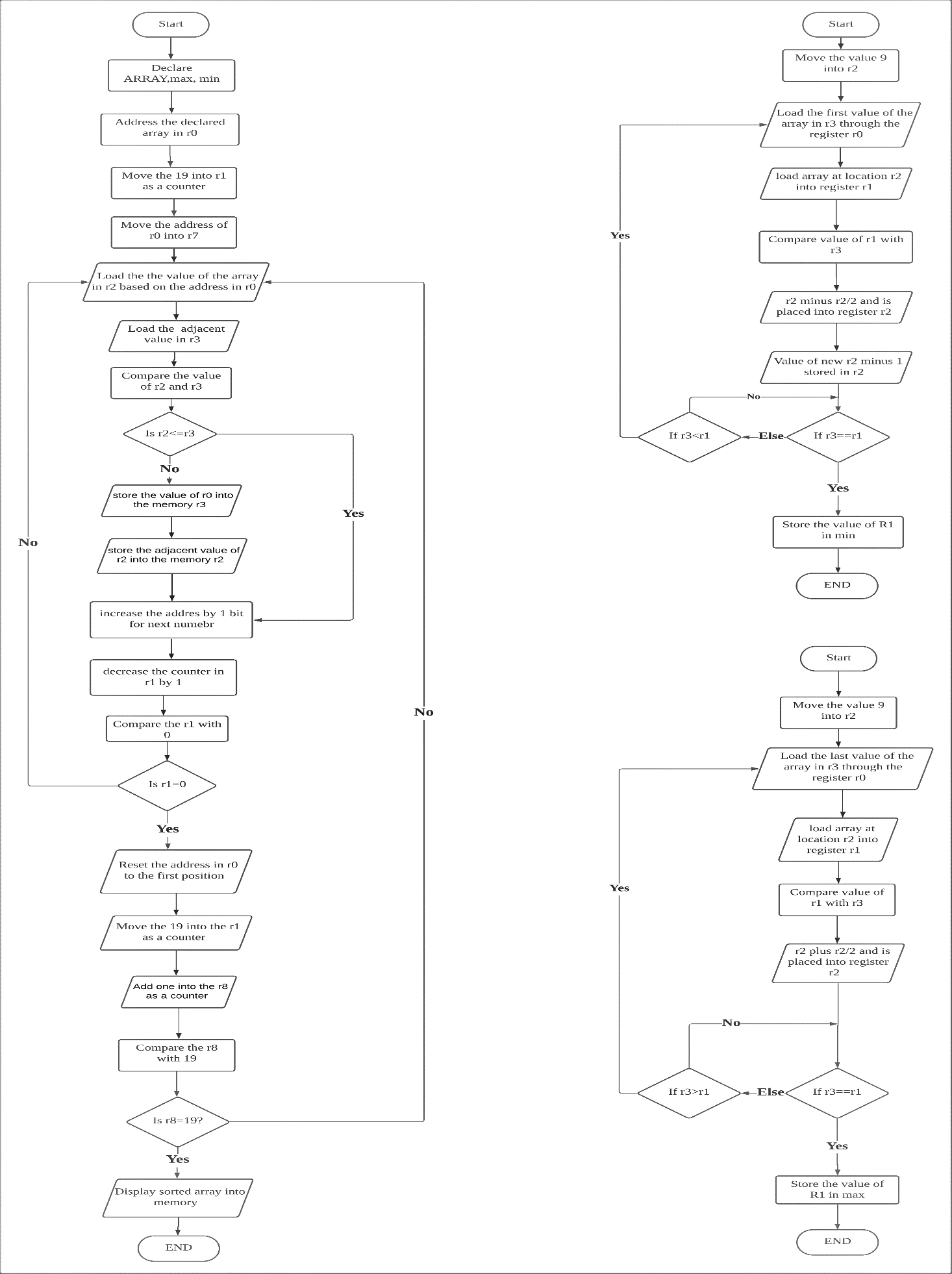
**Q1:Binary Search Tree Algorithm**

**Flowchart**



Binary Search largest value

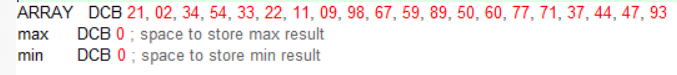
Binary Search smallest value

Sorting

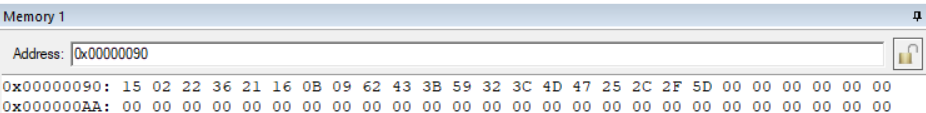
**Explanation**

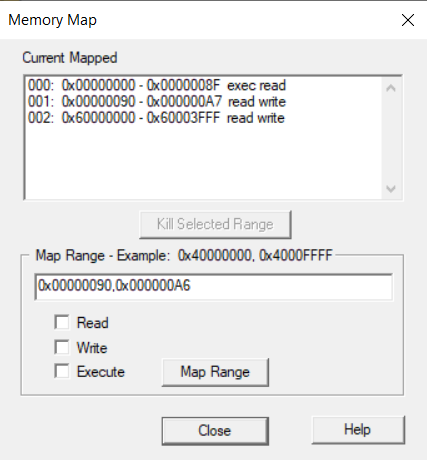
For our ARM assembly language code of the Binary Search Tree Algorithm, there are two main functions in our program which are the sorting the given array of 20 elements and searching the smallest valueand largest valuefrom the array.

We had declared “ARRAY”, “max” and “min” which are the memory space to store the given 20 elements of array, largest value, and smallest value respectively.



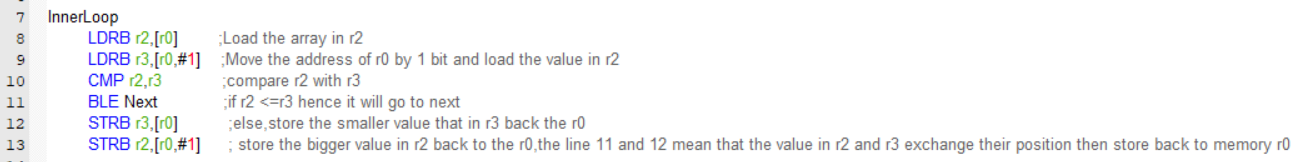
The memory is in the address of 0x00000090.In the debugging mode, we had mapped the memory range from the address 0x00000090 to 0x000000A6 in order to read and write since the DCB directive allocates one byte of memory for each element. There is 23 bytes from the 0x00000090 to 0x000000A6.

In the debugging mode of the memory map:



The sorting array function in our program is based on the concept of bubble sort. Bubble sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are not in the ascending order. The sorting of array will perform when we launch the ARM assembly language code from the line 7 to line 24 in our program. There are two loops in the sorting function which are the “InnerLoop” and “Next”. Firstly, we address the given 20 elements into the register 0 and we move 19 into register 1. The register 1 is now act as a counter. At the line 5, we move the first address of register 0 into the register 7. In fact, the register 7 is used to reset the register 0 into the first address when it exits the InnerLoop.

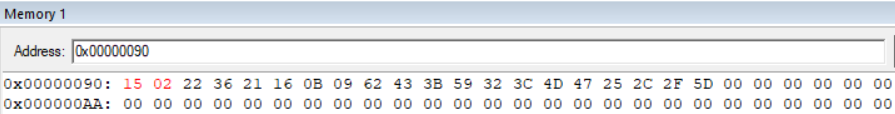
InnerLoop:



In the InnerLoop that started at line 8, we load the value of in the array into the register 2 base on the address of register 0 and load the adjacent value in the same array into the register 3. We use the CMP instruction to compare the value of register 2 and register 3.If the register 2 is less than or equal to register 3 then the program will jump to the Next loop. There is other action if the value of register 2 is bigger than register 3. The value in register 2 will exchange the position with the value in register 3 and store back to the register 0.The Innerloop will stop until the register 1 reach 19.

Example if register 2 bigger than register 3:

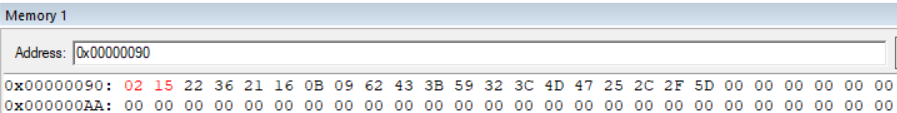
Before:



R2=15, R3=02

The value of 15 is store in register 2 and the value of 02 is store in register 3. Since 15 > 02,then they will exchange their position.

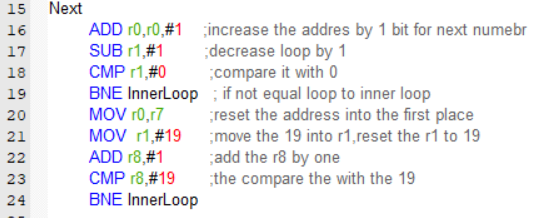
After:



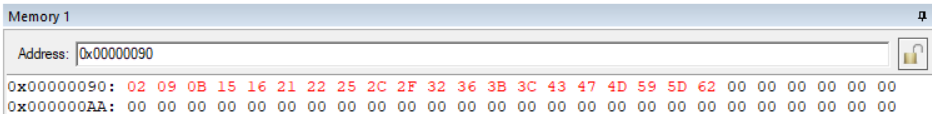
R2=02, R3=15

The position of bigger value, which is the 15 will move backward, and the smaller value which is the 02 will replace the bigger value’s position and store back to the memory.

Next:

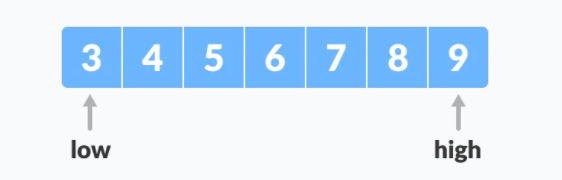


In the Next loop, we increase the address of register 0 by one bit to get the adjacent value. After that, we subtract the counter which is the register 1 by one and compare with the value 0.If not equal, it will jump back to InnerLoop and run until the register 1 is equal to 0.Once the register 1 is equal to 0,we reset the address of r0 into the initial address by using the register 7 and reset the register 1 that acts as a counter to 19.Then,we add the register 8 that initially is 0 by one and compare with 19. If register 8 is not equal to 19, then jump back to the InnerLoop. The register 1 acts as the counter for InnerLoop whereas the register 8 acts as a counter for the Next loop. The Next loop will execute when the register 8 reach 19 and display the sorting array at the memory. The sorting function and the 20 elements were sorted in the ascending order.

The result after the sorting function is done: 

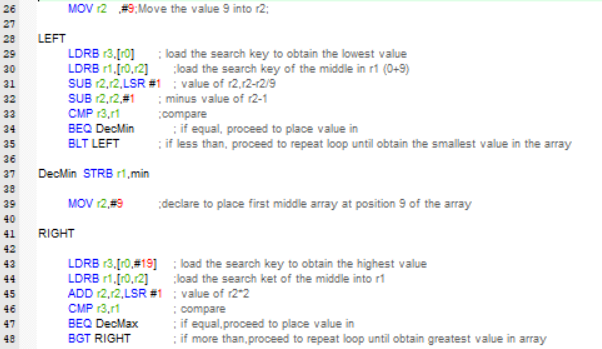
Binary search is a type of search that works well with sorted lists. The concept of the binary search is the divide and conquer. The sorted list is divided into halves and the desired item is compare with the middle element. Then, the position of the middle element is returned if a match is discovered. Otherwise, we look in either of the halves, depending on the match's outcome.

Example of binary search:



The formula of the binary search is (high+low/2).Let the 4 be the search value and we get the middle value from the (9+3)/2=6.The search value is compared  with the middle element. If the search value is greater than the middle value, then the low = middle + 1 and the array move to the left-hand side. If the search value is less than the middle value, then high=middle-1 and the array move to the right-hand side. Since the 4 is less than 6 then it will move to the left-hand side. So, the array is now left with the array 3,4 and 5.We continue with the pervious process that (3+5)/2=4.Then,the search value 4 is found and the program will end.

The binary search function perform form the line 26 to the line 48 in ARM assembly language code. There is a LEFT loop to find the smallest value while the RIGHT loop is to find the largest value.



Before we enter the LEFT loop, we move the value of 9 into the register 2 which use as the middle element of the binary search .Since we are finding the smallest element and we focus on the left-hand of array. Then, we load the register 3 with the value of the position one in the register 0’s memory as the search key. In the line 31 to 32,we perform the formula(low+high)/2 by using the instruction LSR and SUB. We subtract the value of register 2 with the value in register 2 than divided 2 (9-9/2)=5(rounded up).Then, we subtract the value in r2 by 1(5-1=4). Remember than the value of 4 is actually point to the position of 5th position in a memory. We compare the value of r3 with the value of r1.If the value of register 3 and register 1 is equal then store the smallest value in the space of min which is the memory space. If the value of register 3 is less than the value in register 1,then repeat the action in LEFT loop again.

The same concept applies to the RIGHT loop that use to search the largest number in the sorted array. We move the value of 9 into register 2 which use as the middle element of the binary search the largest number before entering the RIGHT loop. Then, we load the value which has highest position of 20 in the memory of register 0 into the register 3 as the search key.We load the middle value of the register 0 into the register 1.In the line 45, we perform the formula(low+high)/2 by using the instruction ADD and LSR. So, we will get the value of 14 and write the value to the r2.Remember that 14 is actually point to the 15th position in the array.Then,compare the value in the register 3 with the register 1.If both of them are equal to each other, then store the value int the space of max which is the memory space. If the value in register 3 is greater than the register 1,continue the RIGHT loop again.

The result in the memory max and min:

