Report Assignment IV

Problem 1:

- Assumptions: A certain number of entered integers is permitted in the mips program, while x86 program the array of integer is assumed to be already in the memory and its size as well.
- Design Approach:
    a. In this problem the two main functions are implemented namely the bubble-sort function and inside of it there is a swap function. The implementation is set so that the parameters and return value of the bubble-sort function is pushed on the stack only once; while, the return values of the swap are saved as many times as the function is called on the stack, then jump to the return address of the last swap after which the stack is restored and the previous swap return is loaded and jumped to, this process continue until the return value of the bubble-sort function is reached. Finally the values of the parameters stored at the beginning of the function are restored and maybe an additional function is implemented to print the values of the array to confirm it is sorted in an ascending order.

Problem 2:

- Assumptions: A certain number of entered integers is permitted in the mips program, while x86 program the array of integer is assumed to be already in the memory, its size as well as the value that the array is searched for.
- Design Approach:
    a. In this problem, binary search is implemented, recursionly, to find some value in an integer array. Two cursors are set at the beginning of the function: one to the left, first element, and one to the right, last element. By adding both cursors and dividing them by two a cursor pointing to the middle element is obtained, the value of middle cursor is checked if it is not word-aligned it is converted to be so. Then the value pointed to by the middle cursor is checked with the searched value if they are a match the function return the index, of it is not a match another call to the function binary search is made with different parameters depending on whether the searched value is bigger or smaller than the value pointed to by the middle cursor. After every new function call the stack is restored and the previous functions return address is loaded and jumped to until we reach the return address of the original function in the main. Note all other paramaters to the function are only stored and loaded once at the beginning and end of

the function respectively, only the return addresses of the function are the ones variably decrementing the stack.