Computer Architecture
Assignment 1

Members:
Yash Sengupta (IMT2022532)
Teerth Balghat (IMT2022586)

Explanation of the Assembly Code:

The code implements bubble sort.
There are 2 registers used $s1 and $s2 both of which are initialised to 0. The sorting is done in place so first we copied all the input values from the location stored in $t2 register to the location stored in register $t3.
This is done using the move1 loop.
Next we are sorting the numbers at the location stored in $t3 while keeping the input as it is. Nested loops are used in the codes(sort1 and sort2). There are two functions calc_i and calc_j used which are used to calculate the ith and the jth address from the starting address $t3. If A[i] > A[j] then the swap procedure is called which swaps the two numbers in the locations. That is how the sorting takes place.

Explanation of the Program that converts the Assembly code to Machine Code:

The re module is used to split the instructions into tokens
The dictionary contains opcodes of various functions used in the code
the memory location of each label , the number of all the various registers used and the binary representation of some immediate fields used in the code.
The memory locations of some labels like the “swap” are 16 bits as the form part of the beq instruction while some labels like calc_i and calc_j have 26 bits memory locations as the form part of the jump instruction.The binary representation of the immediate fields are also written taking consideration of the type of instruction.
s is the assembly code.
We are assuming the fact that the compiler during compilation has already stored the location of the various labels and the immediate fields in functions like beq. These values are stored in the dictionary.
We are splitting the assembly code into instructions using the split() function.
The “machine_code” variable holds the entire binary code while the machine_temp holds the binary value of each instruction.
We specify the delimiters according to which each instruction is split into tokens.

The last two fields of the add and sub instructions include the shift_amountt and function_field which has been added according to the machine code.
The "jr $ra" function has the same machine code every time so that machine code has been hardcoded.
The entire machine_code has been displayed in 32 bits as it is the length of each instruction in MIPS.

Output: