Finding Median

Due: May 15, 2022

Problem Description

Write an assembly program using RISC-V instruction set architecture(ISA) to find the median for a list of integers. If there are an even number of integers, print the two medians. For example, given a list of integers 5, 5, 6, 9, 1, -1, 3, 8, 17, -5, your program should print 5 and 5. You should use RISC-V instruction set simulator **Jupiter** to develop and execute the assembly code. The Jupiter simulator can be downloaded from GitHub https://github.com/andrescv/Jupiter. You can find the instructions for assembler, directives, Ecalls, etc. In particular, **Ecalls** implements a set of ABI's (Application Binary Interface) that can be used by the assembly code to interact with an operating system.

Input Format

The input should be given by following the instruction prompted on the monitor. The first line gives the number of test cases. It is then followed by the input data for each test case read from keyboard. The input of each test case consists of n+1 integers. **Each integer takes a line**. The first integer gives the number of integers and then the *n* integers for a test case. Assume the number of integers for a test case is not greater than 1000.

Output Format

The output for a test case takes one line where the median(s) of the list of integers is (are) printed. If there are two medians, they are separated by a whitespace, the smaller first and then the larger. The output format should be exactly the same as the example given below.

What Should Be Handed In:

- Assembly code for finding medians. **The first line of assembly code should consist of your student ID number and your name**. Write as many comments for the code as possible. It is suggested that every instruction should have a comment to explain what the instruction does. The file name of the assembly code should be **sID.s** where ID is your student ID number. A valid file name will look like **s1091111.s**. The file extension should be **.s**.
- A clip like the one shown in the example of input and output below. Save the clip as a file called **sID.png**, where ID is your student ID number. A valid file name for an output clip will look like s1091111.png.
- The homework will not be graded if you do not follow the above rules.

Other information:
https://github.com/riscv-non-isa/riscv-asm-manual/blob/master/riscv-asm.md
RISC-V ASSEMBLY LANGUAGE, Programmer Manual, Part I
https://shakti.org.in/docs/risc-v-asm-manual.pdf
https://github.com/andrescv/Jupiter/blob/main/examples/fibonacci.s
https://github.com/andrescv/Jupiter

Supplements

To help you to work out this problem, the code and information below is provided to help you make a start.

```
#Student ID: 1098776
#Name: 林榮彬
.globl __start
.rodata
    msg1: .string "Input the number of test cases: "
    msg2: .string "Input the number of integers in a test case: "
    msg3: .string "Start to read data from the keyboard: \n"
    msg4: .string "Start sorting: \n"
    msg5: .string "The sorted list is: \n"
    msg6: .string "The median(s) is(are): "
# The above messages are printed out to direct the input process or provide information about output.
    .data
        ary: .zero 4000    # set up a buffer of 4000 bytes to hold the data read from the keyboard.

    .text
# The remaining code should be written to read data from the keyboard and store the data
# in the buffer (array) ary specified in the data section. Then, sort the data stored in the
# buffer in ascending order for finding out the medians of the list of numbers data in the
# buffer ary. You can use any sorting algorithm. The simpler is better. The code for bubble
sort in page 148 of the textbook can be used for sorting the data. However, the code is not
working correctly. A few lines need to be added to make the code work correctly. You are
encouraged to make this happen. Note that you may not need to make the code as a
function that needs to save some registers into the call stack when called and restore these
registers from the call stack when returning.
```

Example of Input and Output

```
Input the number of test cases: 3
Input the number of integers in a test case: 4
Start to read data from the keyboard:
3
2
1
6
The median(s) is(are): 2 3
Input the number of integers in a test case: 5
Start to read data from the keyboard:
6
28
9
23
44
The median(s) is(are): 23
Input the number of integers in a test case: 6
Start to read data from the keyboard:
-1
-6
23
10
0
18
The median(s) is(are): 0 10
```