Homework-1: IN210 Assembly Language and Computer Organization, Rung-Bin Lin
March 17, 2023, Computer Science and Engineering; International Bachelor Program in Informatics, Yuan Ze University

Bubble Sorting in RISC-V ISA

Due: March 31, 2023

Problem Description

Translate the following C++ code into RISC-V assembly code which can be simulated by the RISC-V instruction set simulator **Jupiter**. The code is used to sort a given number of integers using bubble sort algorithm.

```
void bubbleSort (int v[], size_t n)
{
   size_t i, j;
   for (i = 0; i < n; i += 1) {
      for (j = i – 1;
              j >= 0 && v[j] > v[j + 1];
              j -= 1) {
         swap(v,j);
      }
   }
}
void swap(int v[],int k)
{
   int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
// For main(), you need only translate the highlighted statements.
int main()
{
int numInt; // the number of integers to be sorted
int anAry[256]; // An array for holding 256 integers
cin >> numInt; // Assume numInt is smaller than 256
for(int i=0; i<numInt; i++)
      cin >>anAry[i];
bubbleSort(anAry, numInt);
return 0;
}
```

You should use RISC-V instruction set simulator **Jupiter** (with I, F, M extensions) to develop and execute the assembly code. The Jupiter simulator can be downloaded from GitHub

https://github.com/andrescv/Jupiter. To learn how to use RISC-V ISA to write a program that can be executed on Jupiter, you can study the code **Fibonacci.s** which can be downloaded from https://github.com/andrescv/Jupiter/blob/main/examples/fibonacci.s . You need to learn how the **assembler**, **directives**, **Ecalls**, etc. are used to develop the assembly code. In particular, **Ecalls** implements a set of ABI's (Application Binary Interface) that can be used by an assembly program to interact with an operating system to read data from a keyboard and write data onto a monitor. Refer to https://jupitersim.gitbook.io/jupiter/assembler/ecalls for Ecalls and https://jupitersim.gitbook.io/jupiter/assembler/directives for directives.
It is very easy to use Jupiter. Simply download the related files in its web site. Decompress the file and store the decompressed data in a new folder. Find a file named **Jupiter.bat** in the **bin** folder. Doubly click this file to invoke Jupiter. Then, you can write, assemble, execute, and debug the code directly on Jupiter.

Input Format

When the code is executed on Jupiter, just follows the instruction presented on the monitor to provide input data. Each input line should contain only one data item (this restriction is imposed by the simulator).

Output Format

The output format should be exactly the same as the example given below. Certainly, different inputs will result in different outputs presented on the monitor.

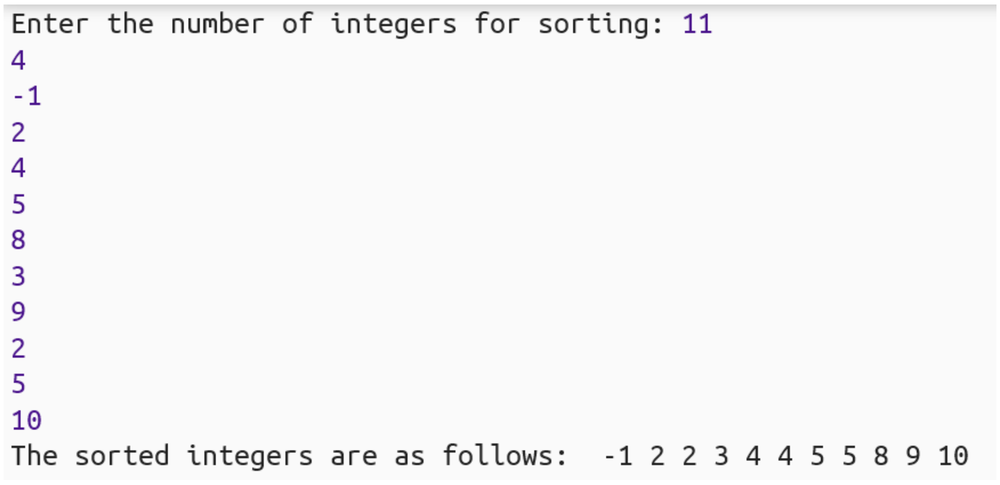
Late Assignment is allowed till April 8. Five points will be deducted for each day late. Such a grace period is used to encourage you doing your assignment as soon as possible..

What Should Be Handed In:

- Assembly code for the whole program. **The first line of assembly code should consist of your student ID number and your name**. A comment should start with # at the beginning of the comment. You should write as many comments as you can. For example, each line of code has a comment. 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 .
- Some clips like the one shown in the example of input and output below. Save the clips as a file called **sID.pdf**, where ID is your student ID number. A valid file name for an output clip will look like s1091111.pdf. You can paste the clips into a WORD file and then create a PDF file from the WORD file.
- The homework will not be graded if you do not follow the above rules.

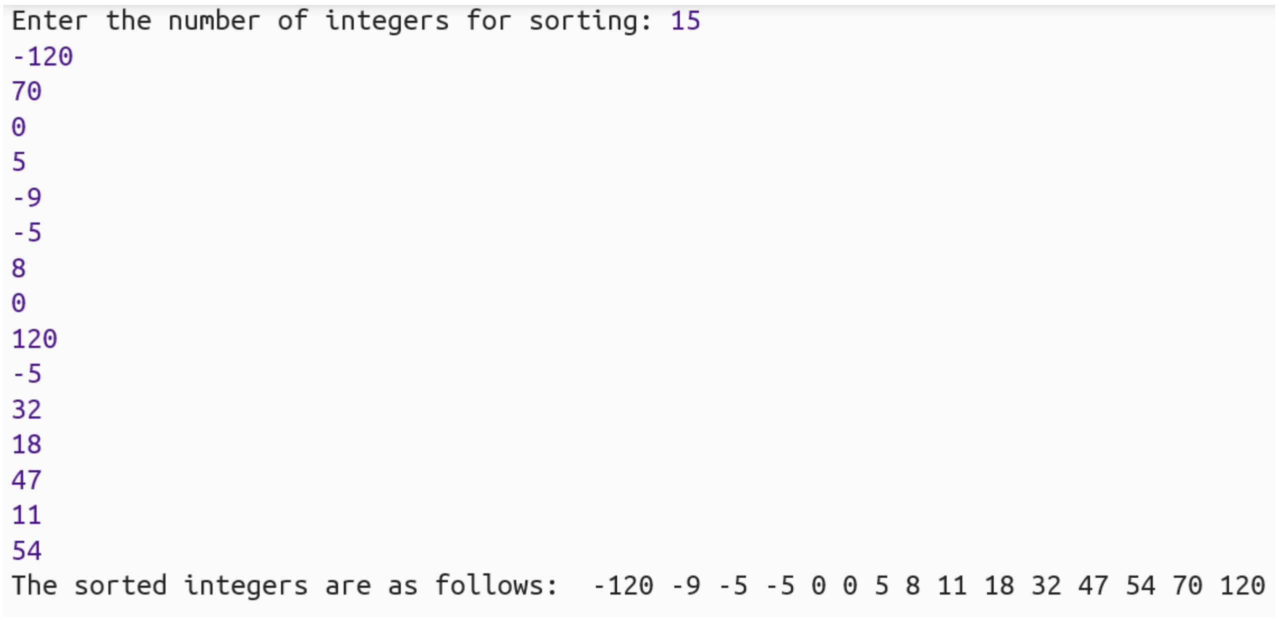
Other information:
https://github.com/riscv-non-isa/riscv-asm-manual/blob/master/riscv-asm.md
https://shakti.org.in/docs/risc-v-asm-manual.pdf
https://github.com/andrescv/Jupiter
https://jupitersim.gitbook.io/jupiter/assembler/directives

Example 1 of Input and Output



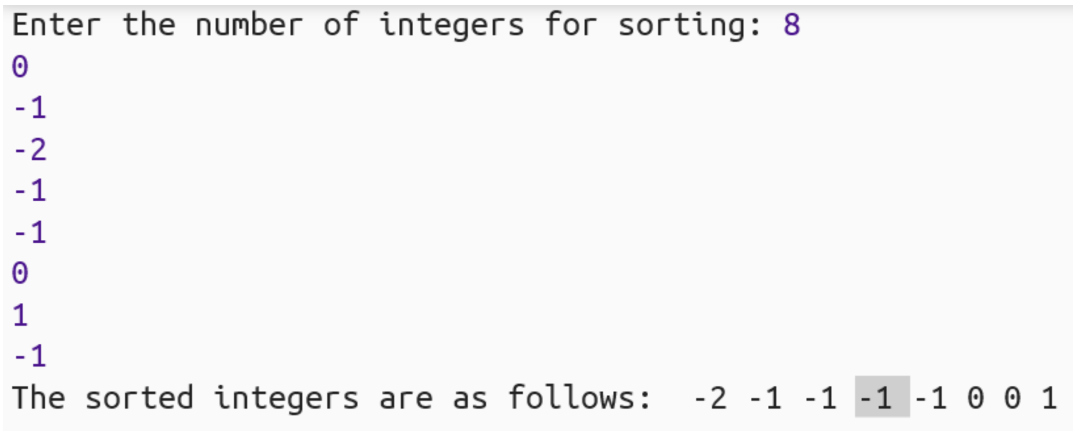
```
Enter the number of integers for sorting: 11
4
-1
2
4
5
8
3
9
2
5
10
The sorted integers are as follows:  -1 2 2 3 4 4 5 5 8 9 10
```

Example 2 of Input and Output



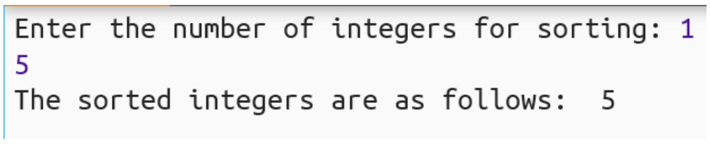
```
Enter the number of integers for sorting: 15
-120
70
0
5
-9
-5
8
0
120
-5
32
18
47
11
54
The sorted integers are as follows:  -120 -9 -5 -5 0 0 5 8 11 18 32 47 54 70 120
```

Example 3 of Input and Output



```
Enter the number of integers for sorting: 8
0
-1
-2
-1
-1
0
1
-1
The sorted integers are as follows:  -2 -1 -1 -1 -1 0 0 1
```

Example 4 of Input and Output



```
Enter the number of integers for sorting: 1
5
The sorted integers are as follows:  5
```