Assignment 2 - Programming Part (50%)

In this assignment, you will implement the **QuickSort** algorithm in RISC-V assembly (RV64I) using the **RARS** RISC-V simulator.

By completing this assignment, you will be familiar with RISC-V calling conventions and be able to work with arrays and pointers in assembly language.

Setting Up RARS (RISC-V Simulator)

We recommend using this fork of RARS as it is actively maintained. Alternatively, you can download RARS 1.7 from this link.

To run RARS, Java 11 or later is required. Once Java is installed, launch RARS with the following command:

```
java -jar <PATH to RARS>
```

Implementing the QuickSort Algorithm

Download the template files (`main.s` and `sort.s`) from NTU COOL. The `main.s` file includes necessary I/O operations, while your task is to complete the `sort.s` file by implementing the QuickSort algorithm in RISC-V assembly.

Run your completed program using this command:

```
java -jar <PATH to RARS> rv64 sm main.s sort.s
```

You must implement the QuickSort algorithm using the **Hoare partition scheme**. Follow the pseudocode provided in the Wikipedia QuickSort entry.

Input and Output Specifications

Input

- The first line specifies the number of elements to sort.
- The subsequent lines contain the individual elements.

Output

- The sorted elements should be printed in ascending order, one per line.

For example, if the input is:

```

```

```
5
2
1
4
5
3
```

The expected output will be:

```
1
2
3
4
5
```

Constraints

- The maximum number of elements will be **1000**.
- Each element will be a **64-bit signed integer**.

Debugging in RARS

To debug your program in RARS, launch RARS in GUI mode using the following command:

```
java -jar <PATH to RARS>
```

Ensure the following options are enabled in the Settings menu:

- 64 bit
- Initialize Program Counter to global 'main' if defined
- Assemble all files in the directory

You can then open `sort.s`, assemble it, run it, and debug as needed.

Report Guidelines

You are required to submit a report written in **Markdown** format, viewable in **Visual Studio Code**. The report can be written in either Chinese or English.

The report should include:

- A description of how you implemented the QuickSort algorithm, including any challenges you encountered and how you addressed them.
- A clear explanation of your code, including how you use the registers and the logic behind each part of your assembly code.

Submission Instructions

Submit a **zip** file via NTU COOL. The zip file must contains the directory structure shown below:

```
<Student_ID>_HW2
├── main.s
├── sort.s
├── report.md
└── <any other files referenced in report.md>
```

Ensure the zip file includes the top-level folder `<Student_ID>_HW2`, so that when extracted, your files are contained within this folder.

Name the zip file as `<Student_ID>_HW2.zip`, ensuring that letters in your Student ID are in uppercase.

Policies

- **Plagiarism is strictly prohibited.** Copying others' work or sharing your own work will result in zero points for this assignment.
- **AI-generated content is not allowed.** Using AI-generated code will result in zero points.
- **You must write the assembly code yourself.** While you may reference compiler-generated code for learning purposes, you are expected to write your own code and explain it in your report.