Goal

Through this lab, you would compile a simple RISC-V CPU simulator with 5 stage pipelines, branch prediction, and cache simulation.

Then you need to modify the source code of the simulator and rewrite part of a C program (`./RISCV-Simulator/test/lab0.c`) to achieve CAS (Compare and swap) using `LR/SC (Load Reserved/Store Conditional)` instructions.

You also need to verify the correctness of your CAS and upload a comprehensive and readable summary report in PDF format to ShanghaiTech Pan.

Lab 0 will take **4 points** of overall for CS211 in Fall 2022.

The due date for Lab 0

23:59:59, 23rd September, 2022. Any submission past the deadline shall not receive any point for this lab.

Deliverable

1. A report in English of how you modify the lab0's C program and what you do with the source code of the simulator, including the file(s) and function(s).
2. Source code with git commit history: the compressed file which contains source code with comments, related test programs, etc.

Both of them shall be submitted on ShanghaiTech Cloud Driver (Campus Network Only or via VPN). The submission link is as follows.
Submission Link

Instruction on how to use RISC-V Simulator

Compile simulator

https://github.com/hehao98/RISCV-Simulator

```
cd RISCV-Simulator
mkdir build
cd build
cmake ..
make
```

Compile `riscv-gnu-toolchain`

https://github.com/riscv-collab/riscv-gnu-toolchain

```
# download
git clone https://github.com/riscv-collab/riscv-gnu-toolchain.git

# compile
cd riscv-gnu-toolchain/
mkdir build; cd build
../configure --with-arch=rv64ia --prefix=/path/to/riscv64ia
make -j$(nproc)
```

Compile example program

```
cd RISCV-Simulator

# compile example code
/path/to/riscv64ia/riscv64-unknown-elf-gcc -march=rv64ia test/lib.c
test/example.c -o riscv-elf/example.riscv

# dump riscv file to human readable
/path/to/riscv64ia/riscv64-unknown-elf-objdump -D riscv-elf/example.riscv >
riscv-elf/example.s
```

Run simulator

```
# help
./Simulator riscv-elf-file-name [-v] [-s] [-d] [-b param]
Parameters:
        [-v] verbose output
        [-s] single step
        [-d] dump memory and register trace to dump.txt
        [-b param] branch prediction strategy, accepted param AT, NT,
BTFNT, BPB

# run example
cd build
./Simulator ../riscv-elf/example.riscv
```

Note:

1. You must use `--with-arch=rv64ia` and `-march=rv64ia` when compiling `riscv64-unknown-elf-gcc` and `example`, respectively, to make sure that the example program uses `RV64IA` instruction set.

   The reason why we use `RV64IA` rather than `RV64I` is that `LR` and `SD` are not supported by `RV64I`.

2. The only library you can use is `test/lib.c`. Using other libraries may cause compile errors or simulator crashes.

The task of Lab 0

CAS (Compare and swap) is a commonly used lock-free operation to modify data atomically.

You can use `Load Reserved (LR)` and `Store Conditional (SC)` instructions to achieve CAS in RISC-V.

What you need to do is:

1. You need to add the `Load Reserved (LR)` and `Store Conditional (SC)` instructions in the simulator according to what we have learned in the course of CS110/CS110P (or equivalent courses in other universities) and RISC-V Instruction Set Manual 2.2.
2. You need to achieve the CAS using `LR/SC` in the lab0 program (`./RISCV-Simulator/test/lab0.c`).
3. Compile and run lab0.c. Run it with your modified simulator.
4. Verify the correctness of your implementations.
5. Prepare a comprehensive and readable report in PDF form.

Hint:

You may use inline assembly in your lab0 program.

Note:

1. In order to simplify the implementation of `LR/SC`, you don't have to consider the following constraints of `LR/SC`.
    1. The address has to be aligned to 4 or 8 bytes.
    2. The number of instructions between `LR` and `SC` is limited.
2. You also do not have to consider the `FENCE` in your CAS code.

How to use git

```
# see the current status (modified files, changes not staged)
git status

# add modified files into the staging area
git add file1.c file2.c file3.c

# commit files with commit message
git commit -m "commit message"

```

Ref:

1. https://hehao98.github.io/posts/2019/03/riscv-simulator/
2. https://riscv.org/wp-content/uploads/2019/12/riscv-spec-20191213.pdf
3. https://www.runoob.com/git/git-tutorial.html
4. Should you have any further question or problem, please contact TAs (Jiang Qisheng and Hu Yanpeng).