Goal

Through this lab, you would modify the RISC-V CPU simulator to support 2 cores and simulate different programs in the different cores at the same time. Then you need to modify the simulator's cache hierarchy to support basic cache coherence protocol.

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

Lab 0 will take **7 points** of overall for CS211 in Fall 2022.

The due date for Lab 3

23:59:59, 1st December 2022. Any submission past the deadline shall not receive any points for this lab.

Deliverable

1. A report in English of your design, implementation, test, etc. In a nutshell, it shall include everything that makes a self-contained practice.
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: http://pan.shanghaitech.edu.cn/cloudservice/outerLink/decode?c3Vnb24xNjY2NzA5NDM4ODMyc3Vnb24=

The task of Lab 3

Task 1

Since 2005, improvements in system performance have mainly been due to increasing cores per chip. CPUs with single core are no longer satisfying our requirements.

Therefore, in this task, you are asked to enhance the simulator to support 2 cores and concurrently run different programs on the different cores.

Requirements:

1. Each core has its registers, PC, pipelines, two-level private caches (i.e., L1 and L2 caches), and so on. But these two cores share one last-level cache (i.e., L3 cache).
2. In order to run different programs separately, you need to run your simulator in the following way.

```
# -c0: riscv-elf-file-name of the program running in core 0
# -c1: riscv-elf-file-name of the program running in core 1
./Simulator -c0 ../riscv-elf/test_share1.riscv -c1 ../riscv-elf/test_share2.riscv
```

3. Simulator does not distinguish between the virtual address and the physical address. Therefore, you need to find out how to avoid any conflict between two programs' address spaces.

Hint:
A program shall have different segments in the ELF file, such as text segment, data segment, BSS segment (block started by symbol), heap, and stack. [Memory Layout of C Programs]

Task 2

Although multicore CPUs improve the performance of computer systems, they also incur problems with data coherence, in particular, cache coherence.

In order to maintain your simulator's correctness, you are required to add cache coherence protocol to it.

Requirements:

1. You are supposed to implement MESI cache coherence protocol based on directories (Lecture 14).
2. Running the `lab3-core0.c` and `lab3-core1.c` in two different cores at the same time.
    1. Put them into your `test` directory and compile them with `riscv64-unknown-elf-gcc`.
    2. You should allocate a shared memory, whose start address is 0x100000 and length is 4MB, and initialize the contents of shared memory as zeros at the beginning in your simulator.
    3. `lab3-core0.c` and `lab3-core1.c` access the same addresses `a[0]` and `a[257]` which are true sharing for the cache. If your MESI cache coherence is correct, these programs should print desired correct values.
    4. Add your own test cases, which must have false sharing and verify the correctness of your implementations.
3. In order to simplify the implementation of MESI, you don't have to consider that one core is writing a cache line while the other is concurrently reading or writing the same cache line at the same clock cycle.

Bonus:

Bonus points may be given to students that successfully deal with the situation in which one core is writing a cache line while the other is concurrently reading or writing the same cache line at the same cycle.

Ref:

0. Use well the duration of five weeks.
1. https://www.geeksforgeeks.org/memory-layout-of-c-program
2. Should you have any further questions or problems, please contact TAs (Jiang Qisheng and Hu Yanpeng).