

Introduction to Computer Architecture

Project 3 – 10,000-Line Project

Spike RISC-V Simulator

Hyungmin Cho

Department of Software
Sungkyunkwan University

Project 3 Overview

■ Goal 1:

- ❖ Use the RISC-V instruction simulator's cache simulation functionality to find the best cache configuration for the given applications.
- ❖ You can do this without understanding the C++ code

■ Goal 2:

- ❖ Study the cache simulation in the Spike RISC-V instruction simulator
- ❖ Modify the simulator to implement the LRU replacement algorithm
- ❖ Need to modify C++ code

Spike RISC-V simulator

- Spike is a RISC-V instruction simulator, similar to what you have created for proj2.
- Spike supports full RISC-V ISA, including various extensions as well.
- Spike also simulate some simple cache model to give you the resulting cache statistics (e.g., # of accesses, # of hits, etc.).
 - ❖ **In proj3 we will focus on the level 1 data cache only!**
- With more than 40,000 lines of code, it surpasses the requirement for the 10,000-line project.
- **You need to build (compile) Spike on the department server.**

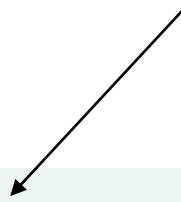
Building (Compiling) Spike RISC-V simulator

Originally, this repository came from <https://github.com/riscv-software-src/riscv-isa-sim>

■ Build steps

- ❖ Clone the Spike git repository to your local directory

```
git clone ~swe3005/2023f/proj3/riscv-isa-sim
```



- ❖ Checkout the v1.1.0 version

```
cd riscv-isa-sim
git checkout v1.1.0
cd ..
```

- ❖ Create a “build” directory

```
mkdir build
```

- ❖ Run configuration in the build directory

```
cd build
../riscv-isa-sim/configure
```

- ❖ Run compilation in the build directory

```
make
```

← After you make changes in the source, code run “make” again in the build directory.

Directory Structure

- Let's assume “proj3” is your working directory.


```
proj3
|- riscv-isa-sim
|   |- arch_test_target
|   |- ci-tests
|   |- ...
|   |- goal1.txt
|   |- goal2.txt
|   |- ...
|- build
|   |- spike
|   |- ...
```

- This is the spike project directory
- Source files exist here.
- Modify source codes here.
- **Submit this directory only**
- You need to add these two files in your submission
- This is where the object files & output files exist.
- Run “make” command here
- Execute “spike” program here
- **Do not submit this directory**

Executing the Spike Simulator

- In the build directory....

```
./spike ~swe3005/2023f/proj3/pk ~swe3005/2023f/proj3/bench/CCa.elf
```



This is the “proxy kernel” binary that helps the simulator to run the RISC-V binary program without modeling a full-scale operating system.



Name of the RISC-V program binary

It's okay if you don't understand what this means.

Just use this argument here.

Executing the Spike Simulator With Cache simulation

- In the build directory....

```
./spike --dc=32:8:64 ~swe3005/2023f/proj3/pk ~swe3005/2023f/proj3/bench/CCa.elf
```

This argument sets the data cache configuration.
The meaning of the values are Sets:Ways:BlockSize.

--dc=32:8:64 means that an 8-way data cache with 32 sets, where each cache block is 64 bytes.
That is, the total cache capacity is 16 KB.

If you simulate with this parameter, the simulator prints the cache statistics after the execution.

```
D$ Bytes Read:      413541
D$ Bytes Written:   138694
D$ Read Accesses:   54814
D$ Write Accesses:  21250
D$ Read Misses:     130
D$ Write Misses:    1042
D$ Writebacks:      875
D$ Miss Rate:       1.541%
```

We use this data cache miss rate only

Goal 1: Finding the Best Cache Configuration

- Let's suppose the data cache capacity is fixed to 16KB. Also, the size of the block is fixed to 64 bytes.

- ❖ The possible cache configuration is as follows

```
--dc=1:256:64  
--dc=2:128:64  
--dc=4:64:64  
--dc=8:32:64  
--dc=16:16:64  
--dc=32:8:64  
--dc=64:4:64  
--dc=128:2:64  
--dc=256:1:64
```

- Your goal is to find the best cache configuration for EACH of the application in the benchmark suite.
 - ❖ There are total 40 benchmark applications in `~swe3005/2023f/proj3/bench/`
 - `CCa.elf`, `CCe.elf`, ... `STL2b.elf`

Goal 1: Finding the Best Cache Configuration

■ What to submit:

- ❖ For each of the 40 applications, find the best cache configuration and the resulting cache miss rate.
- ❖ Write the result in the text file named **goal1.txt** and include it in your submission
- ❖ The contents of the text file should look like the following

```
CCa --dc=64:4:64 1.504%
CCe --dc=128:264 1.402%
...
...
...
STL2b -dc=1:256:64 9.095%
```

- ❖ The 40 applications should appear in the alphabetical order by their name.
 - CCa, CCe, CCh, CCh_st, CC1, CCm, CF1, CRd, CRf, CRm, CS1, CS3, DP1d, DP1f, DPcvt, DPT, DPTd, ED1, EF, EI, EM1, EM5, M_Dyn, MC, MCS, MD, MI, MIM, MIM2, MIP, ML2, ML2_BW_ld, ML2_BW_ldst, ML2_BW_st, ML2_st, MM, MM_st, STc, STL2, STL2b

Goal 2: Modify the Cache Replacement Algorithm

- The current Spike simulator models a random replacement algorithm.
- Your goal is to modify the simulator to implement the **true LRU** cache replacement algorithm, and submit the modified Spike source code
- Studying the current spike simulator and finding out where to modify is the requirement of the 10,000-line project.
- Therefore, the following questions will not be answered. Discussing the related topic between the students are also prohibited.
 - ❖ The locations (file names, locations in the file) to modify.
 - ❖ The structure of the Spike source code.
 - ❖ The exact value of the cache statistics (miss rate, the best cache configurations, etc..)

Goal 2: Best Cache Configuration Under the LRU

- In addition to the modified source code, you need to measure how the cache behavior changed with the LRU replacement algorithm.
- Again, for each of the 40 applications, find the best cache configuration and the resulting cache miss rate when the LRU replacement algorithm is used.
 - ❖ Write the result in the text file named **goal2.txt** and include it in your submission
 - ❖ The contents of the text file should look like the following

```
CCa --dc=16:16:64 1.425%  
CCe --dc=16:16:64 1.332%  
...  
...  
...  
STL2b -dc=1:256:64 9.095%
```

You can see that the true LRU algorithm improved the cache miss rate

Randomness In the Cache Miss Result

- The proxy kernel simulation structure in Spike has some indeterministic behavior.
- This may result in non-identical miss rate depending on your environment.
- Also, the best cache configuration may not be the same.
- When we score your submission, we will allow the small differences caused by such randomness.

Reference Implementation?

- No reference implementation or output files will be given for proj3.

Project Environment

- You **must** use the department's In-Ui-Ye-Ji cluster
 - ❖ `swui.skku.edu`
 - ❖ `swye.skku.edu`
 - ❖ `swji.skku.edu`
 - ❖ ssh port: 1398

Submission

- Submit the source code directory only. Do not submit the build directory
- Don't forget to include "goal1.txt" and "goal2.txt" in the submitted directory.

```
proj3
|- riscv-isa-sim
|   |- arch_test_target
|   |- ci-tests
|   |- ...
|   |- goal1.txt
|   |- goal2.txt
|   |- ...
|- build
|   |- spike
|   |- ...
```

- If your directory structure looks like the left, go to proj3 directory, and run the following

```
~swe3005/bin/submit proj3 riscv-isa-sim
```

Submitted Files for proj3:

File Name	File Size	Time
proj3-2020123456-Sep.05.17.22.388048074	6908805	Thu Sep 5 17:22:49 2020

Make sure the file size is not too different from this

Submission

- Verify the submission
 - ❖ `~swe3005/bin/check-submission proj3`
- Make sure you have submitted “goal1.txt” and “goal2.txt”

```
~swe3005/bin/check-submission proj3 | grep goal
[243] riscv-isa-sim/goal2.txt                2068 bytes
[1546] riscv-isa-sim/goal1.txt                2068 bytes
```


Project 3 Due Date

- 2023, Dec. 15th, Friday, 23:59:59
- No late submission
- No extensions
 - ❖ Beware of the server down issue.
 - ❖ It should be better to submit before the deadline.