# Introduction to Computer Architecture Project 4

#### **Cache Simulator**

**Hyungmin Cho**Department of Software
Sungkyunkwan University

## **Project 4 Overview**

- In Project 4, you'll implement a cache simulator
  - This cache simulator takes a list of data memory access (trace) from a file
  - The simulator's role is to report the number of total cache misses.
  - This simulator should support two types of cache configurations
  - We do not consider instruction cache. Only data cache is simulated
- This project is not directly related to Project 1 3.

- Please note that there are several changes from the previous projects
  - Submission rule, due date, etc...

# Cache Type 0

■ 16KB (16,384 bytes) of total data capacity

■ 16 bytes (4 words) per block

Direct-mapped

Write-through, no write allocate policy

32-bit memory address

# **Cache Type 1**

■ 64KB (65,536 bytes) of total data capacity

64 bytes (16 words) per block

2-way set associative, LRU replacement policy

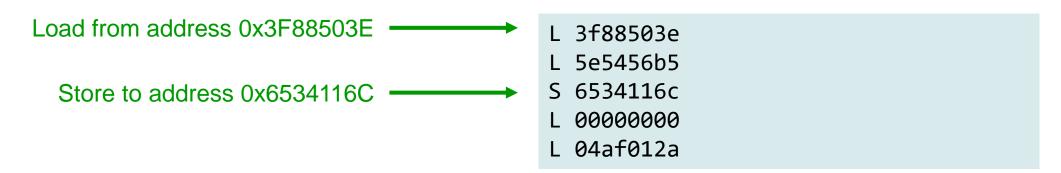
Write-back, write allocate policy

32-bit memory address

#### **Trace File Format**

#### Text format

- ❖ Please note that this is different from the binary files used in project 1 3
- Each line indicates a load or store access
  - A line is either starting with 'L' for a load access or 'S' for a store access.
  - After a single space character, there is an 8-digit hexadecimal number for the address
    - > The address may not be aligned in 4-byte (e.g., L 10000001), but it does not affect the hit / miss count.
  - Example



# **In/Output Format**

- Your program get 2 integers from command-line arguments
  - Simulator type and test file number.

```
Program example

Executable file example

./cache-sim 0 1 ← means run type0 simulator with file trace1.txt

Python example

python3 cache-sim.py 0 1 ← means run type0 simulator with file trace1.txt
```

- Your program must return 2 integer values
  - Miss count and memory write count.
  - Put those two values in a single line (separated by a space)

```
Output example

./cache-sim 1 100

← means run type1 simulator with file trace100.txt

← means miss count is 2231, and the number of memory writes is 233
```

## **Directory Structure**

- Test trace file will locate in the same directory that stores source file.
  - It means your program read ./trace1.txt if second argument is 1.

Dir example

Working\_dir

- \*\*\*\*.cpp
- Makefile
- trace1.txt
- trace2.txt

#### **Build / Run Command Rule**

- You have to create Makefile like in previous projects.
  - \* 'make run' must be implemented.
  - Your program needs 2 arguments, so in your Makefile you should write \$(ARGS) after run command to pass arguments to your program.
  - Notice that you must use the correct spelling of \$(ARGS)
  - To run your program with ARGS variable, don't forget the quotation marks.

```
Makefile example for python

all:
.PHONY=clean
.PHONY=run
run:
   python3 cache-sim.py $(ARGS)
clean:
```

To run your program (example)

\$ make run ARGS="0 1"

## **Evaluating your program**

- For this project, you can get your score immediately at submit.
  - Like previous project, you should put your source codes and Makefile into a submission directory (ex, src/).
  - Run ~swe3005/bin/autoscore proj4 src/
  - Note that only your last execution score would be saved.
  - The autostore script will be fully activated from June 8<sup>th</sup>
- You must submit your final code using submit command as well.
  - Run ~swe3005/bin/submit proj4 src/

#### **Announcement**

 Question is acceptable until June 24<sup>th</sup> (after two days of the assignment due date)

■ TA's email address is <u>siisee111@gmail.com</u>

■ The subject should start with "[CA-project4] ..."

#### **Example Inputs**

- Total 7 test cases are provided.
  - \* ~swe3005/2020s/proj4/

	Total accesses	Cache Type 0		Cache Type 1	
		Miss #	Memory write #	Miss #	Memory write #
trace1.txt	40	5	20	2	0
trace2.txt	8192	2048	0	512	0
trace3.txt	15	13	0	7	0
trace4.txt	11	8	3	4	0
trace5.txt	100	64	52	44	0
trace6.txt	1000	683	518	503	50
trace7.txt	311494	41491	91969	6900	1708

- There are additional files (typeX\_traceN.txt) files that indicates hit/miss for each access.
  - These files are provided to help understanding of the operation.

## **Project 4 Due Date**

- 2020 June 22<sup>nd</sup> (Monday), 23:59:59
  - Please note that the due date is until Monday, not Wednesday.
- Late penalty (max 2 days): 20% per day
  - Only 2 days of late submission is allowed for the last project.