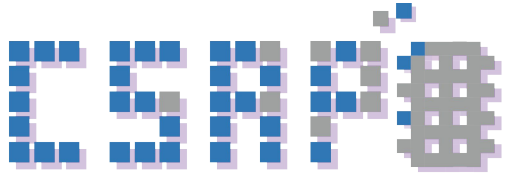
Computer Architecture Lab Session

4. Cache Lab Hints

2019 / 05 / 22

comparch@csap.snu.ac.kr



Computer Systems and Platforms Laboratory
School of Computer Science and Engineering
Seoul National University

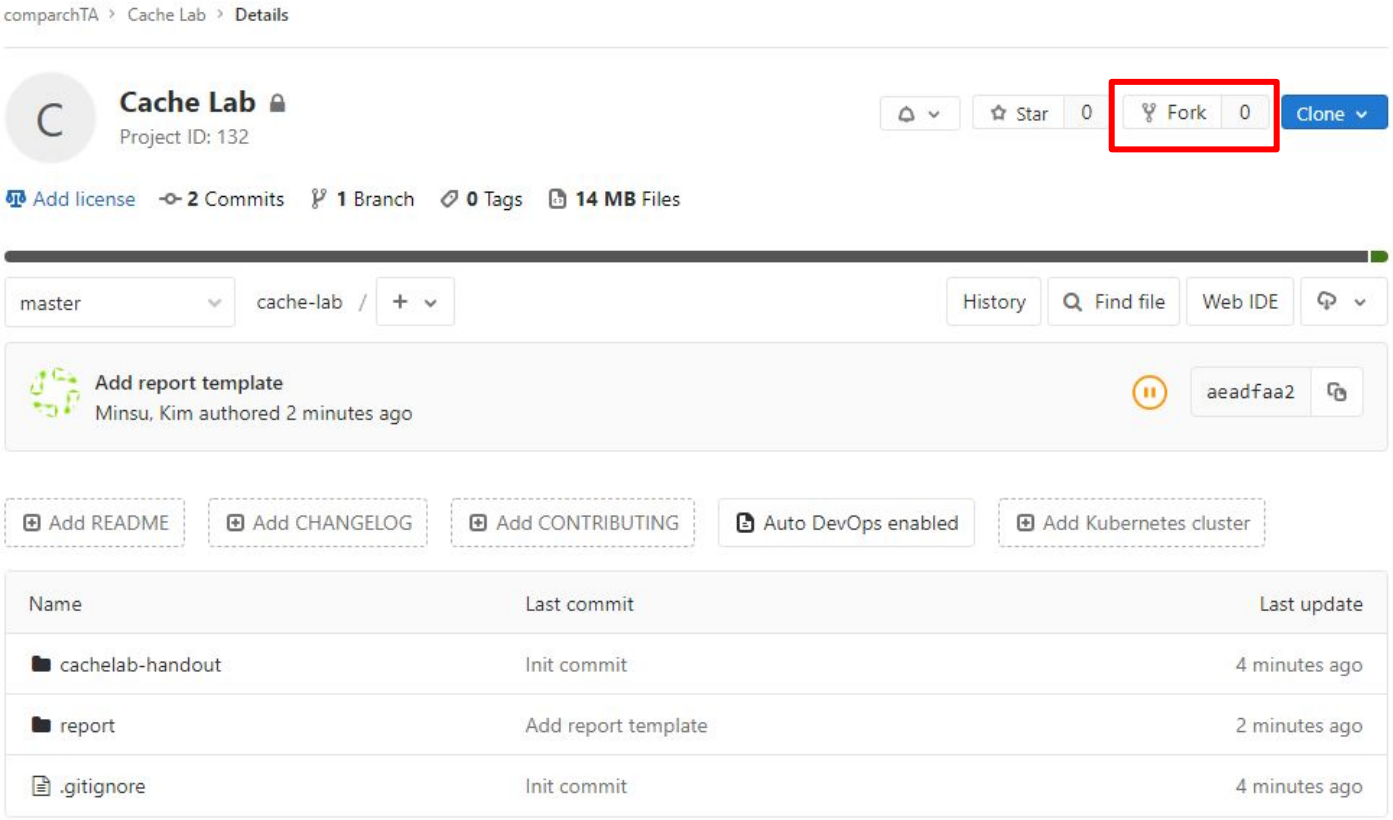
Overview

- Goal
  - Understand the organization and operation of caches by implementing your own cache simulator.
  - Understand the replacement and write policies when cache miss occurs.
- Project Environment
  - OS : Gentoo Linux VM with kernel version 4.9.6
  - Programming Language : C
  - Build Tool : make
  - Memory Tracing Tool : Valgrind

0. Getting Started

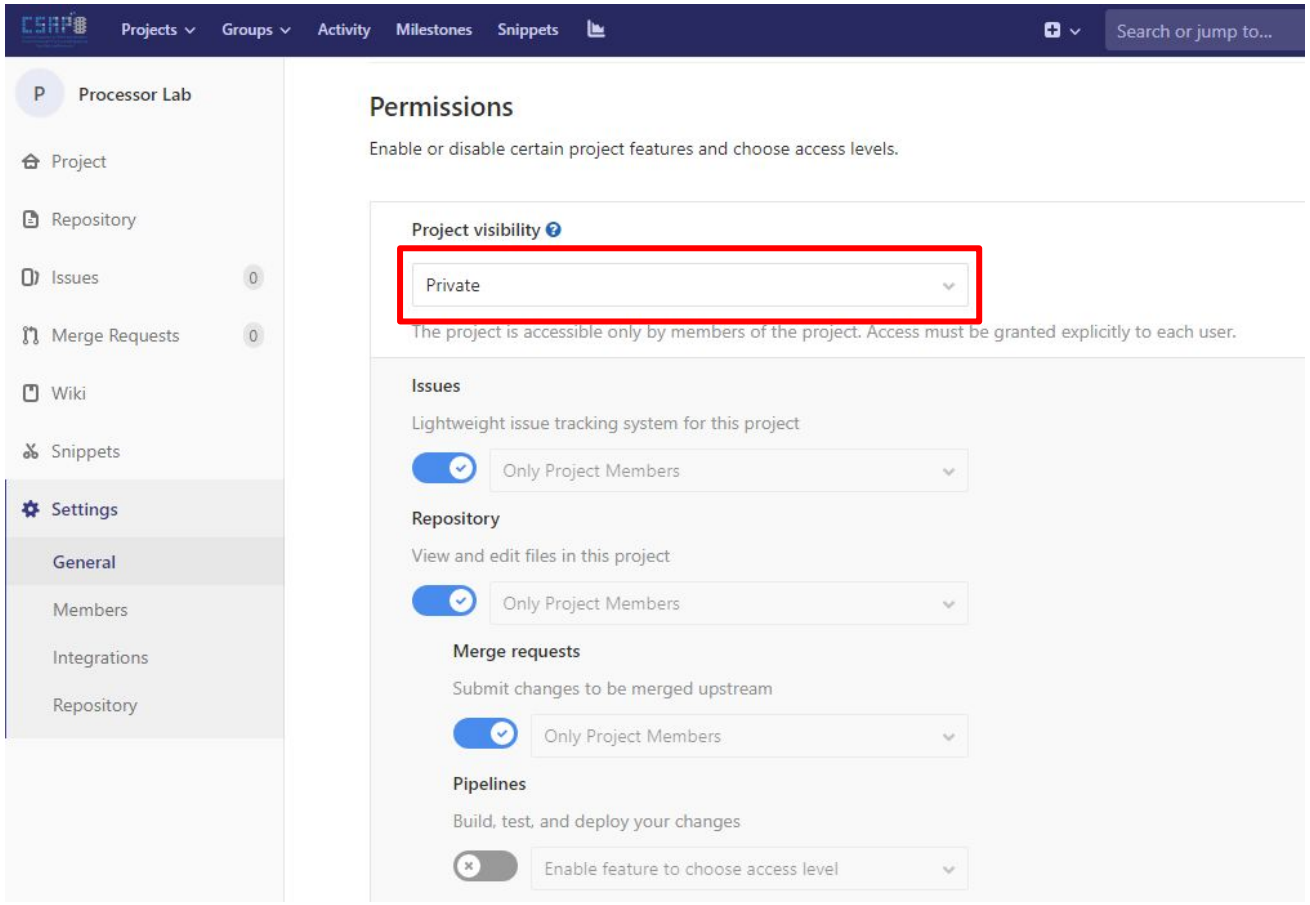
Fork Repository

- Fork from the repository `comparchTA/Cache Lab`
- You will have your project on `https://git.csap.snu.ac.kr/<your_id>/cache-lab`



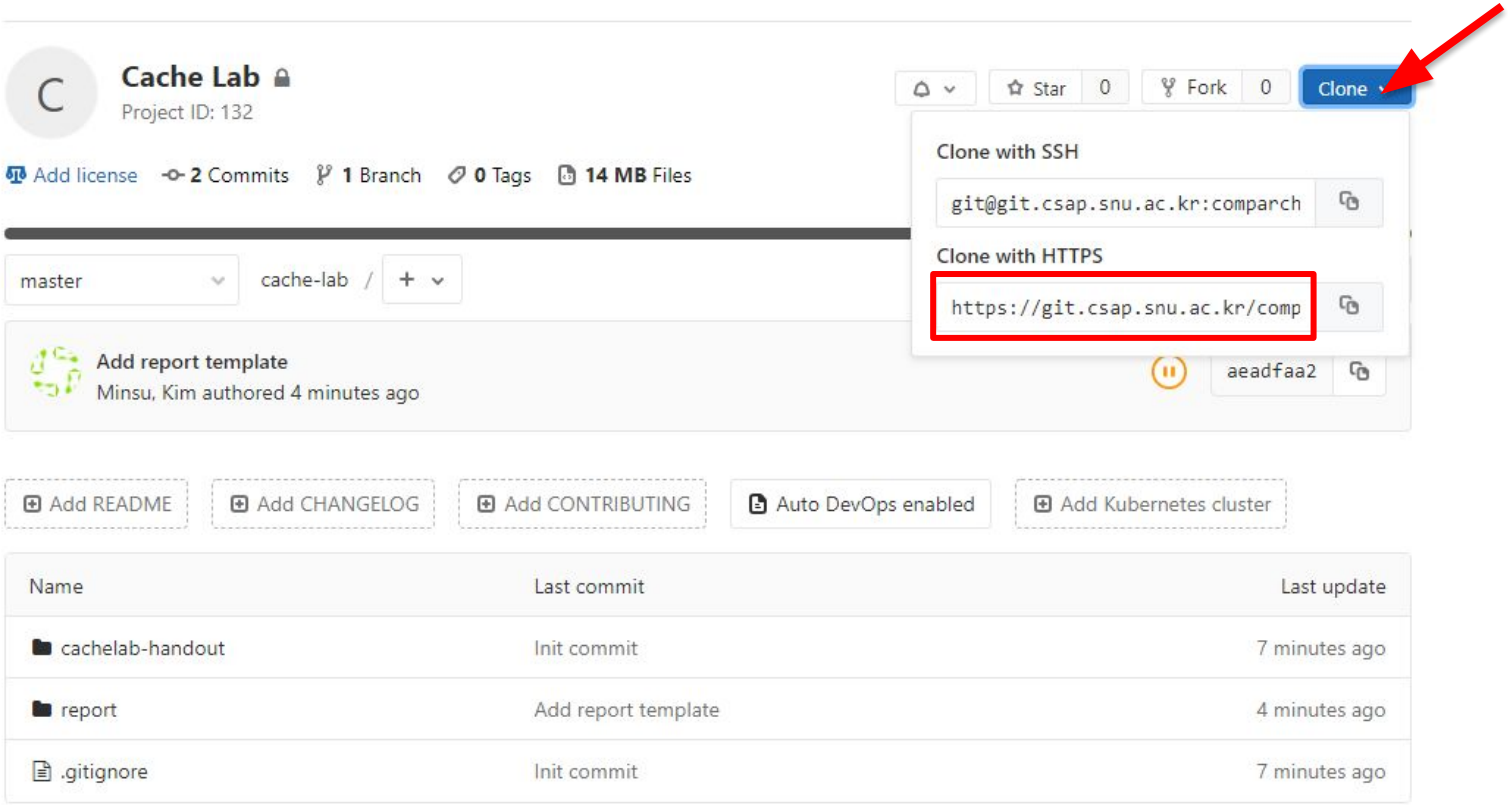
Fork Repository

- Make sure that your repository is **PRIVATE**
  `(Settings > General > Permissions > Project visibility)`



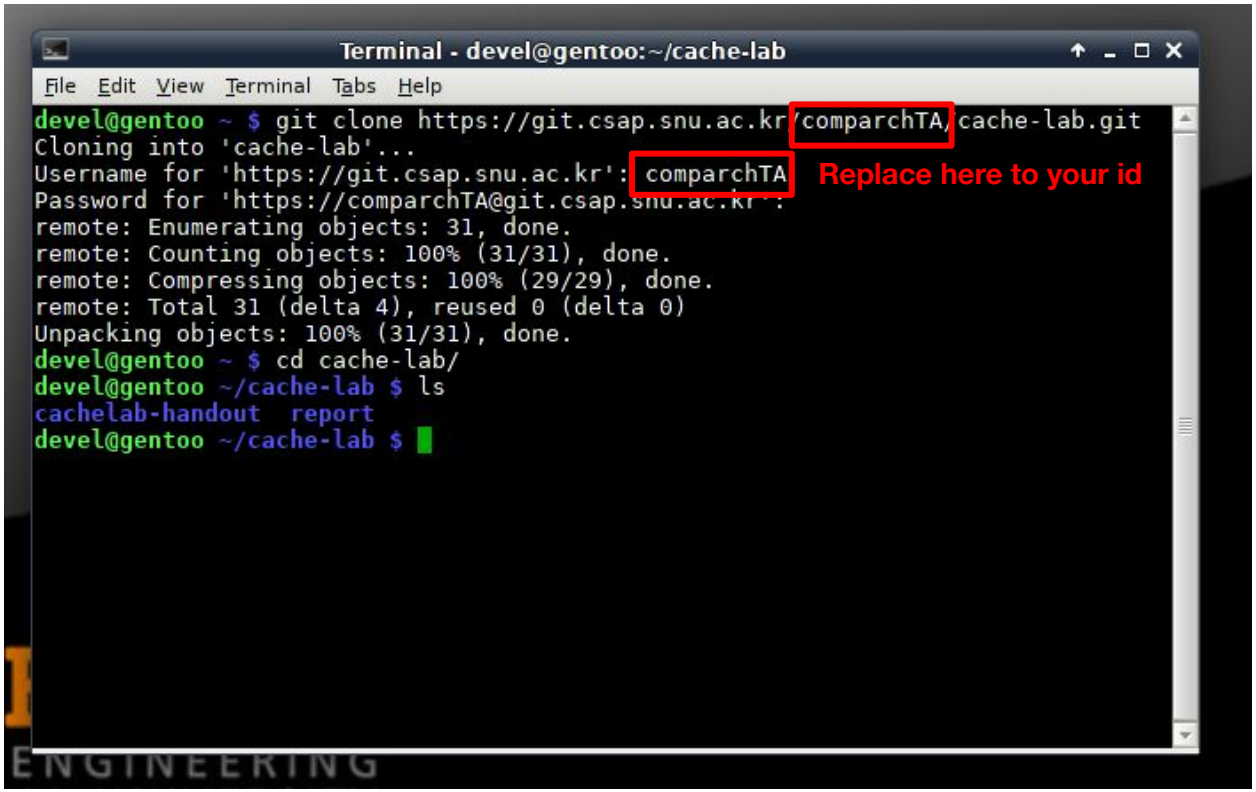
Clone Repository

- Go to your project webpage:
  `https://git.csap.snu.ac.kr/<your_id>/cache-lab`
- Press `Clone`
- Copy the HTTPS URL to clipboard



Clone Repository

- On your terminal in VM, clone repository as follows:
  - git clone `https://git.csap.snu.ac.kr/<your_id>/cache-lab.git`



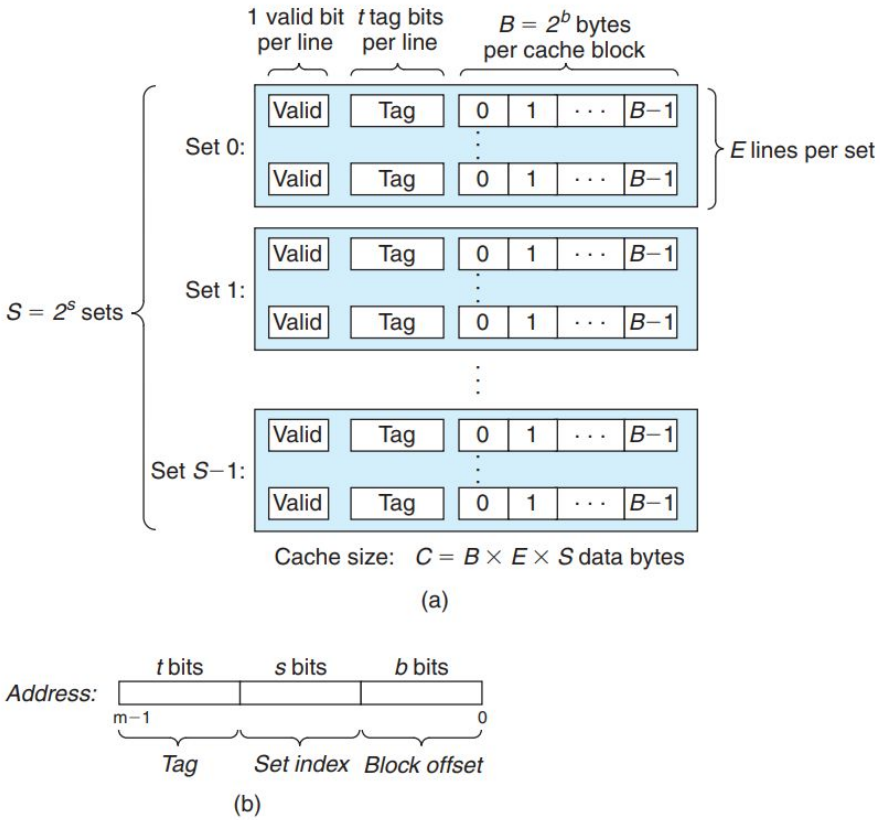
1. Cache Simulator

Project Goal

- Implement cache memory simulator.
- Your simulator should support various replacement policies.
  - Round-Robin
  - Random
  - Least-Recently Used
- Your simulator should support various write allocate policies.
  - Write-allocate
  - No write-allocate
- Describe your implementation in your report.
  - How to implement the simulator.

Cache Memory Overview

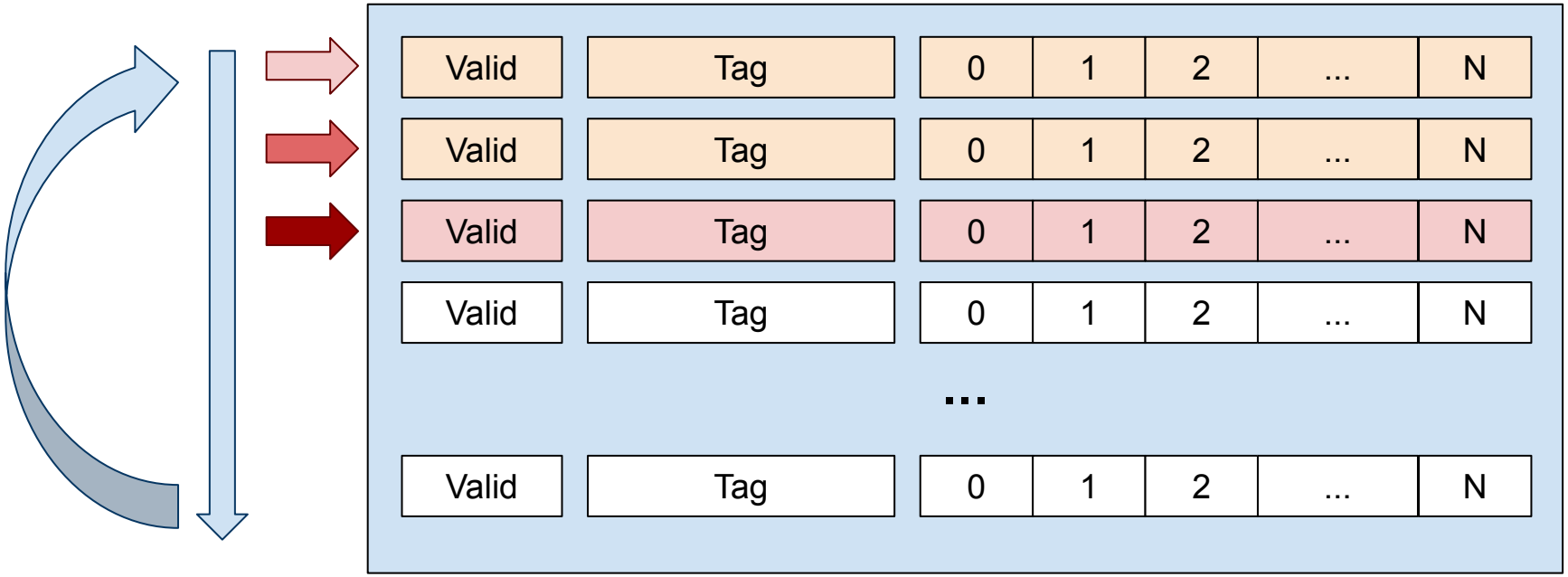
- Cache memory is small SRAM between CPU register file and main memory.
- Cache memory reduces the performance gap between the CPU and main memory.



- Cache memory has a number of sets which includes cache lines.
- Cache line has valid bit, tag bits, and data blocks.
- The number of cache lines in one set depends on associativity.
- You can access data block by splitting data address.

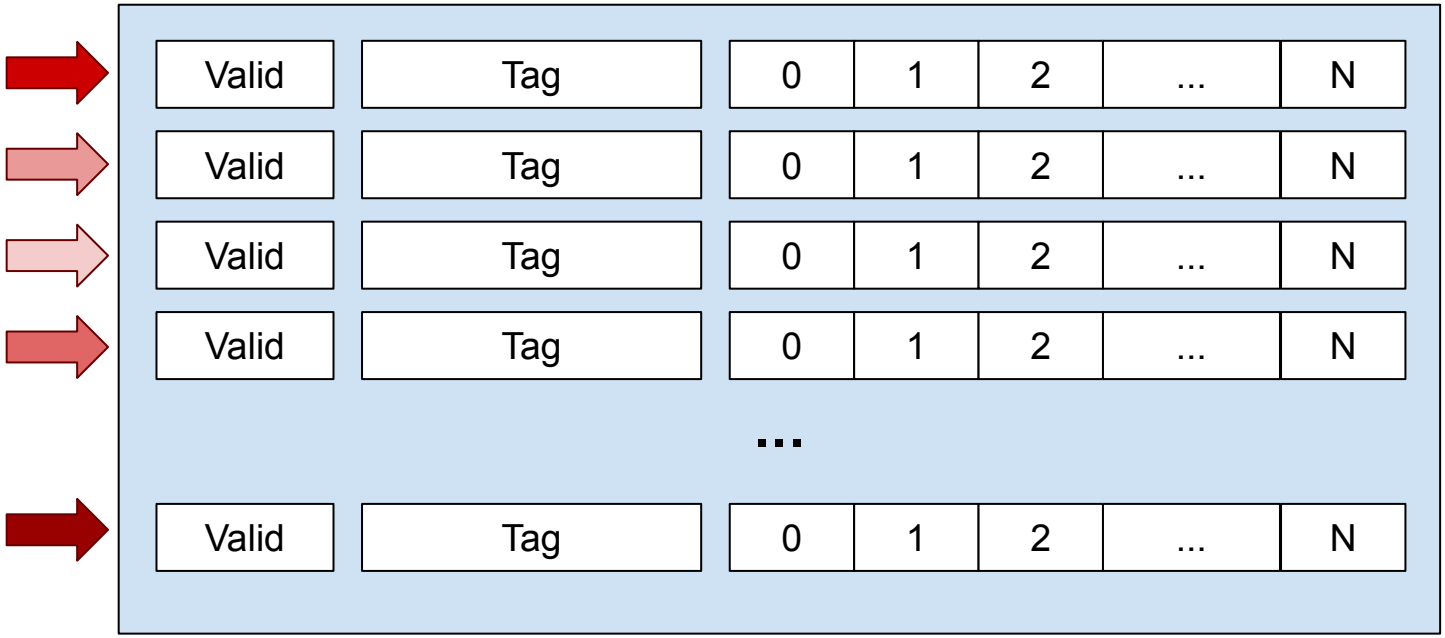
Replacement Policy

- Round-Robin
  - Choose the replacement target sequentially.
  - If the target pointer arrives at the final cache line, it goes to the first one.
  - You can use modulo operation to get the target pointer location.



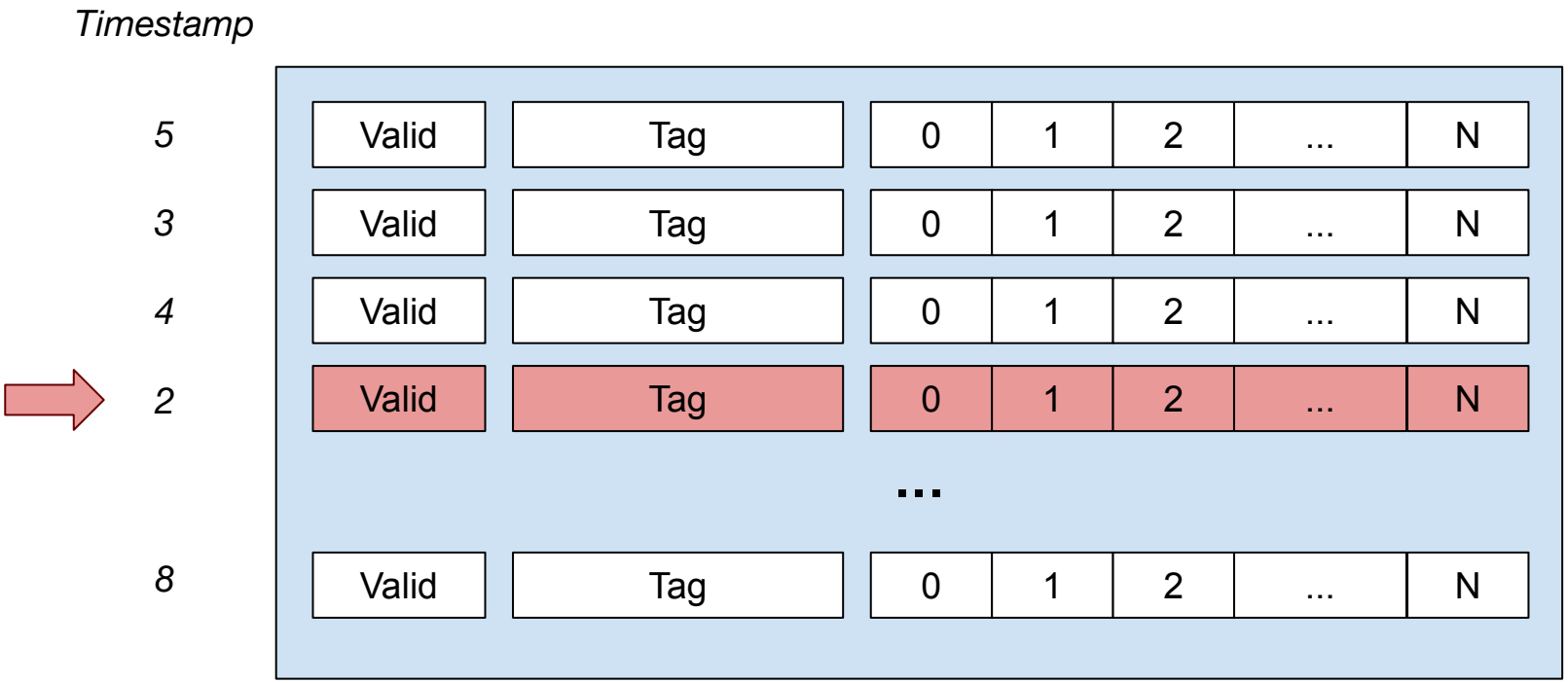
Replacement Policy

- Random
  - Choose the replacement target randomly.
  - You can use `rand()` function in `stdlib.h`.



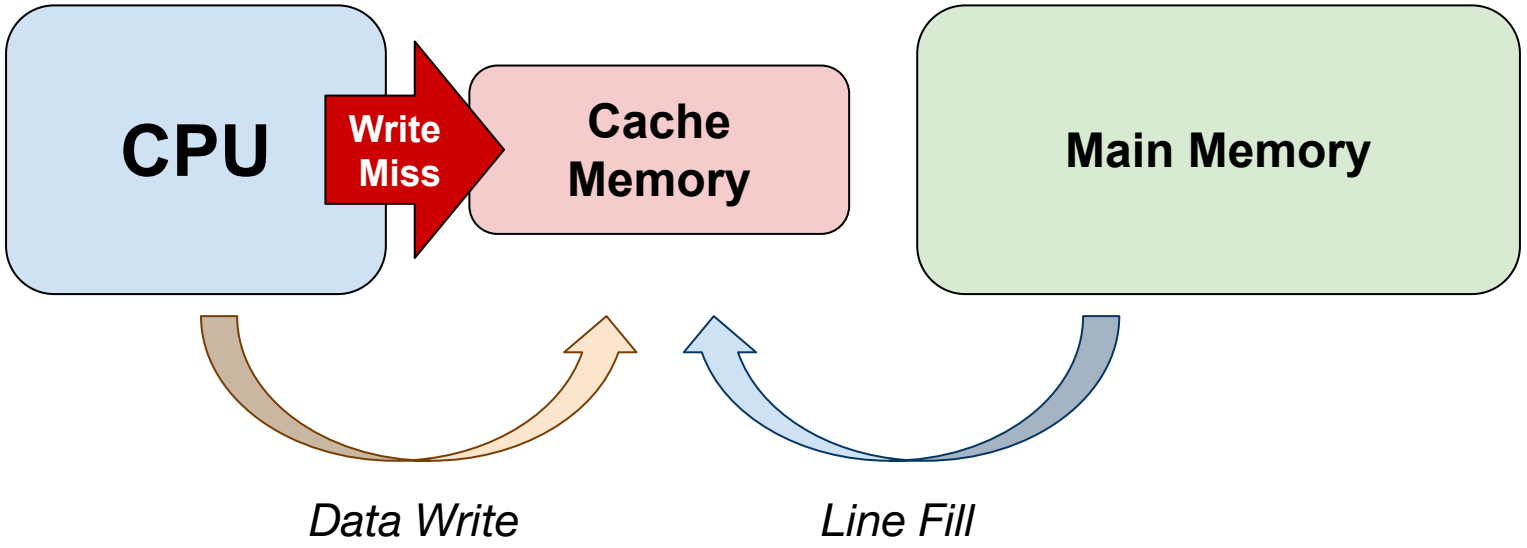
Replacement Policy

- Least-Recently Used
  - Replace the least recently used cache line.
  - Each cache line should have timestamp when it was used recently.
  - Compare the timestamps when choosing the replace target.



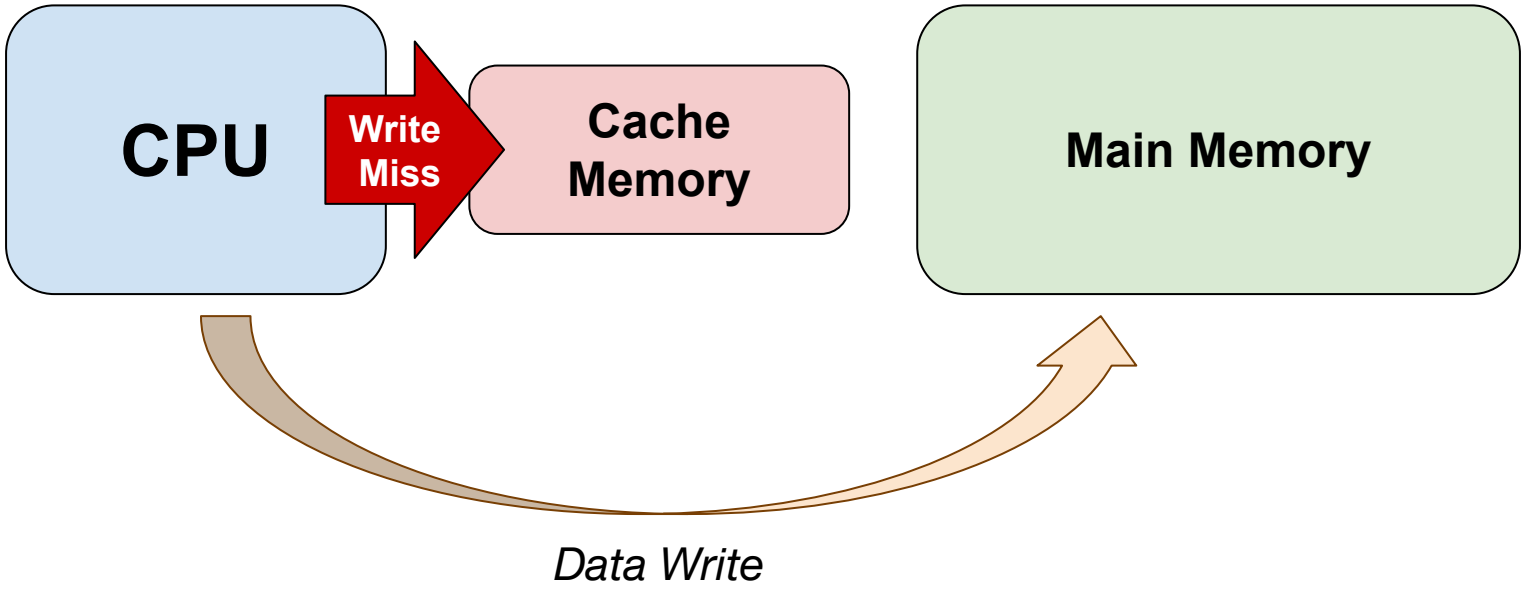
Write Allocate Policy

- Write-allocate
  - When the write cache miss occurs, allocate the cache line to the cache memory and write the updated data to cache memory.
  - Cache memory needs cache line fill before writing the data to cache.



Write Allocate Policy

- No write-allocate
  - When the write cache miss occurs, write the data to main memory directly without cache line allocation.
  - You should count up the miss counts, but there is no cache line fill.



Cache Simulator Details

- Usage
  - You get some command line parameters to set the cache configuration

```
$> ./cachesim [-hv] -c <capacity> -b <block size> -w <ways> -r <replacement policy> -W <write policy>
```

Parameter	Description	Required / Optional
-h / --help	Show help screen	Optional
-v / --verbose	Be verbose while running	Optional
-c / --capacity <*number*>	Set the cache capacity	Required
-b / --blocksize <*number*>	Set block size	Required
-w / --ways <*number*>	Set the number of ways	Required
-r / --replacement <*number*>	Set the replacement policy	Required with Default
-W / --write <*number*>	Set the write policy	Required with Default

Cache Simulator Details

- Replacement policy number
  - 0 - Round Robin < *default* >
  - 1 - Random
  - 2 - LRU (Least-Recently Used)
- Write allocate policy number
  - 0 - Write-allocate < *default* >
  - 1 - No write-allocate
- You can use other replacement policy in reference binary `cachesim-ref`. But you can implement only three of them (Round Robin, Random, LRU).
- You get the policy parameters by integer number in command line.

Cache Simulator Details

- Cache simulator reads its input from `stdin`. Use the pipe operator to cat a trace into the simulator as follows.

```
$> cat traces/test.2.trace | ./cachesim -c 256 -b 16 -w 1 -r 2
      blocksize:      16
      ways:            1
      sets:           16
      tag shift:       8
      replacement:    LRU (least-recently used)
      on write miss:  write-allocate
   Processing input...

   Cache simulation statistics:
      accesses:        9
      hit:             4
      miss:            5
      evictions:       3
      miss ratio:     55.56%
```

Cache Simulator Details

- You can use `bunzip2` command to read compressed trace files.

```
$> bunzip2 -c traces/cat.1.trace.bz2 | ./cachesim [cache options]
```

- If you use a verbose mode, cache simulator shows the information for each data access.

```
Processing input...
R   10 [ t:   0, s:   1 ]: miss alloc (free)
R   20 [ t:   0, s:   2 ]: miss alloc (free)
W   20 [ t:   0, s:   2 ]: hit
R  110 [ t:   1, s:   1 ]: miss evict alloc (0)
R  210 [ t:   2, s:   1 ]: miss evict alloc (0)
```

Read / Write

Address
Tag
Set index

miss / hit
evict
line allocate
free line / evicted line index

Implementation

- Implement a cache simulator.
  - You can implement following `TODO` sections in skeleton codes `cache.c/h`
- Implement three of replacement policies and two of write policies.
  - Round Robin
  - Random
  - LRU (Least-Recently Used)
  - Write-allocate
  - No write-allocate
- You don’t need to implement verbose mode but we strongly encourage you to implement some form of feedback to debug your simulator.
- You can check the correctness of your simulator using reference binary `cachesim-ref`.

Experiment

- Organize and do some experiments for interesting topics in cache memory system.
- Experiment topics:
  1. When the cache size and associativity is fixed, find the relationship between miss rate and block size.
  2. Find the relationship between miss rate and associativity.
- Write the explanation about this experiments and results in your report.
- You should attach some diagrams or graphs to explain your results.

Build Project

- We will build your project using `Makefile`.
- You don't need to write the `Makefile` in this project, just use predefined `Makefile`.
- How to use `Makefile`?
  - Build the project

    ```
    $> make
    ```

  - Clean the project (i.e. remove all output files)

    ```
    $> make clean
    ```

2. Submission and Grading

Submission

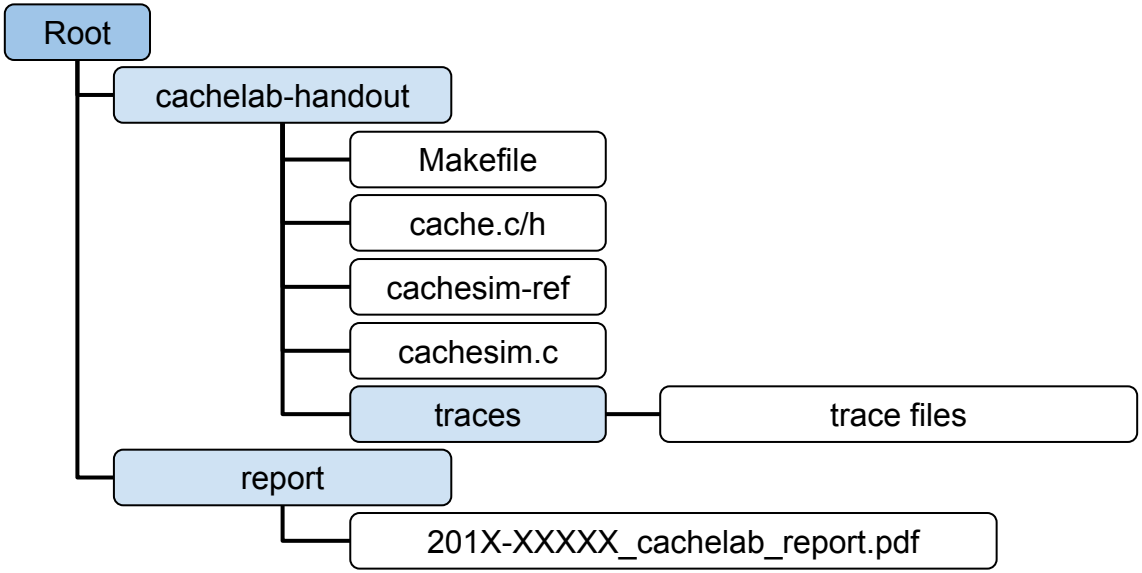
- Submit your source codes and report to master branch of remote git repository.

```
$> git add <path/file_name>

$> git commit -m “<commit message>”

$> git push origin master
```

- Project structure and File naming
  - Follow the project structure and file naming rules.



Report

- Use the report template under `report` directory.
- Before start to write a report, erase all italic font descriptions in the report template.
- Your report should include the brief statements for the project and the explanation about the program you implemented.
- Your report should not be longer than 8 pages (excluding the cover page).
- Avoid copy-pasting screenshot of your code. We already have your code.
- You should attach some diagrams or graphs to depict your experiment results.
- The file name format should match `201X-XXXXX_cachelab_report.pdf`. Replace the number at front to your student id and place your `report` under report directory.

Grading

Round-robin replacement policy implementation and correctness	**20 points**
Random replacement policy implementation and correctness	**20 points**
LRU replacement policy implementation and correctness	**20 points**
write-allocate / no write-allocate policy implementation and correctness	**10 points**
Report	**30 points**

If you violate the submission formats, for example remote repository url format, file naming, and project directory structure, there will be some deductions on your points.

For questions contact

comparch@csap.snu.ac.kr