AJOU UNIV.
1973

Computer Architecture

Fall 2025

What I offer

- **Sample input/output**
- **Skeleton code**
- **Makefile**
- **and this slide**

(renaming is not mandatory)

What you have to submit!

~~1. rename! (main.c → 202512345.c)~~

2. submit!

Assignment 2: Cache Simulator

- **Objective**
  - The objective of this assignment is to implement a cache simulator that models the behavior of a single-level cache and to design and evaluate a new replacement policy

- **Purpose of the assignment**
  - To help you gain a deeper understanding of cache organization
  - To improve your C programming skills
  - To give you hands-on experience designing a new replacement policy and observing how design choices affect cache performance

- **Note**
  - You can modify any part of “main.c”

Assignment 2: Cache Simulator

- **Implementation instructions**
  - The cache simulator will simulate an L1 split cache with the following characteristics:
    - Cache organization: Split L1 cache with separate I-cache and D-cache
    - Cache sizes: 1024, 2048, 4096, 8192, or 16384 bytes
    - Block sizes: 8, 16, 32, 64, or 128 bytes
    - Associativity: Direct-mapped, 2-way, 4-way, or 8-way set-associative
    - Replacement policies: LRU (Least Recently Used), FIFO (First-In, First-Out), or NEW (a custom policy that you will design)
    - Write/allocate policy (D-cache): Write-back and write-allocate
    - Cache level: L1 (separate I-cache and D-cache)

    - You don’t have to consider the write buffer

Assignment 2: Cache Simulator

- **Execution command**

```
Usage: ./cacheSim.out <policy> <trace_file> [cycle_params]
  <policy>         FIFO, LRU, NEW or BEST (case-insensitive)
  <trace_file>     input trace in .txt format
  [cycle_params]   Required only for BEST policy:
                     <i_hit> <i_miss> <d_hit> <d_miss>
  Example (FIFO):   ./cacheSim.out FIFO trace1.txt
  Example (LRU):    ./cacheSim.out LRU trace1.txt
  Example (NEW):    ./cacheSim.out NEW trace1.txt
  Example (BEST):   ./cacheSim.out BEST trace1.txt 1 100 1 50
```

Assignment 2: Cache Simulator

- **Execution command parameters**
  - LRU
    - Run the simulator using the LRU replacement policy
  - FIFO
    - Run the simulator using the FIFO replacement policy
  - BEST
    - Run both LRU and FIFO for all cache configurations, then print the best policy, block size, associativity, miss rate, total cycles, and writes for each configuration (for the given cache size)
  - [Optional] NEW
    - Implement your own replacement policy that performs better than LRU and FIFO for specific configurations (to be described later)

Assignment 2: Cache Simulator

- **[Optional] NEW replacement policy**
  - Before attempting this part, make sure you have fully completed FIFO, LRU, and BEST
  - Your task is to design a new replacement policy that performs better than both FIFO and LRU when:
    - cache size = 1024 bytes
    - block size = 8 bytes
    - associativity = 8 ways
    - input file = trace1.txt (open case)
  - A “better” policy means that both the I-cache and D-cache miss rates are lower than those of FIFO and LRU

Assignment 2: Cache Simulator

- **[Optional] NEW replacement policy**
  - You must implement `print_new_cache_state`
    - Refer to `print_two_cache_state` as an example
      (You can use this function for debugging even if you do not attempt the NEW replacement policy)
    - I will use this function to verify whether your NEW policy is truly better than LRU and FIFO
    - You must also use `print_new_cache_state` to show the intermediate results and prove you did not cheat in your report
    - Any attempt to fake or manipulate miss rates will result in minus 20 points in the final exam
      (e.g., always recording hits)

Assignment 2: Cache Simulator

- **[Optional] NEW replacement policy**
  - This policy does not affect this Assignment 2 score
  - However, there will be a question about your NEW replacement policy and how you implemented it on the final exam
    - This question will be worth 20 points (20/100)
    - To receive full credit, you must

      Step 1. implement the policy,

      Step 2. implement the print_new_cache_state,

      Step 3. show that it performs better (pdf file), and

      Step 4. explain your design clearly (in final exam)

    - For the Step 3, submit a short report (< 1 page) in PDF format and email it to me (jhkwak@ajou.ac.kr)

Assignment 2: Cache Simulator

- **Input format**

  `<time_stamp> <label> <address>`

  - <time_stamp>
    - Indicates the index (order) of the cache access
  - <label>
    - 0: read data, 1: write data, 2: instruction fetch
  - <address>
    - The address is in hexadecimal and uses byte addressing

- **Sample input**

```
...
12 0 100122f8
13 2 40bc98
14 1 7ffebac8
```

Assignment 2: Cache Simulator

- **Output format**
  - The skeleton code provides the output functions
  - Please refer to the sample output and the skeleton code
  - Remove all debugging print statements before submitting your code

```
                                $ ./lfu.out FIFO  long_input/trace1.din
Reading trace file: long_input/trace1.din
Trace contains 1000002 memory accesses.
Simulating FIFO policy...

MissRate
          FIFO/8                                    FIFO/16                                   FIFO/32                                   FIFO/64                                   FIFO/128
I cache    1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384
Direct | 0.2952 0.2438 0.1879 0.1268 0.0741 0.1757 0.1450 0.1125 0.0769 0.0452 0.1101 0.0908 0.0716 0.0506 0.0294 0.0756 0.0624 0.0498 0.0359 0.0208 0.0539 0.0445 0.0354 0.0263 0.0158
2  Way | 0.2871 0.2324 0.1674 0.1046 0.0566 0.1702 0.1384 0.1011 0.0650 0.0348 0.1059 0.0868 0.0646 0.0435 0.0233 0.0722 0.0601 0.0452 0.0314 0.0172 0.0514 0.0428 0.0329 0.0235 0.0138
4  Way | 0.2894 0.2304 0.1634 0.0959 0.0475 0.1705 0.1377 0.0994 0.0605 0.0297 0.1056 0.0872 0.0643 0.0406 0.0199 0.0719 0.0601 0.0448 0.0292 0.0149 0.0511 0.0420 0.0326 0.0219 0.0118
8  Way | 0.2903 0.2323 0.1604 0.0928 0.0453 0.1711 0.1393 0.0975 0.0590 0.0284 0.1052 0.0881 0.0636 0.0398 0.0193 0.0716 0.0609 0.0446 0.0291 0.0140 0.0504 0.0421 0.0331 0.0220 0.0113

          FIFO/8                                    FIFO/16                                   FIFO/32                                   FIFO/64                                   FIFO/128
D cache    1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384
Direct | 0.2103 0.1546 0.0920 0.0617 0.0409 0.1991 0.1409 0.0785 0.0501 0.0303 0.2040 0.1435 0.0740 0.0454 0.0252 0.2225 0.1554 0.0816 0.0496 0.0249 0.2631 0.1881 0.1078 0.0615 0.0276
2  Way | 0.1758 0.1186 0.0796 0.0495 0.0348 0.1601 0.1021 0.0653 0.0386 0.0248 0.1632 0.1035 0.0611 0.0335 0.0191 0.1769 0.1116 0.0654 0.0341 0.0172 0.2151 0.1355 0.0807 0.0424 0.0188
4  Way | 0.1622 0.1109 0.0749 0.0472 0.0322 0.1464 0.0947 0.0605 0.0356 0.0222 0.1475 0.0934 0.0542 0.0303 0.0163 0.1594 0.1022 0.0563 0.0297 0.0140 0.1904 0.1227 0.0711 0.0356 0.0154
8  Way | 0.1600 0.1075 0.0738 0.0458 0.0320 0.1443 0.0909 0.0597 0.0339 0.0217 0.1436 0.0890 0.0523 0.0293 0.0158 0.1581 0.0972 0.0543 0.0283 0.0139 0.1887 0.1176 0.0665 0.0328 0.0148

Write Count
          FIFO/8                                    FIFO/16                                   FIFO/32                                   FIFO/64                                   FIFO/128
I cache    1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384
Direct |      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0
2  Way |      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0
4  Way |      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0
8  Way |      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0

          FIFO/8                                    FIFO/16                                   FIFO/32                                   FIFO/64                                   FIFO/128
D cache    1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384   1024   2048   4096   8192   16384
Direct | 23873 17783 10752  7304  4489 20661 14587  8174  5325  3149 20191 14370  7086  4312  2347 20885 14866  7209  4260  2016 23002 16643  8725  5037  2070
2  Way | 20247 13243  9555  6395  4102 17209 10326  6974  4458  2848 16588  9916  6048  3440  2039 17198 10134  6026  3105  1655 20081 12165  7049  3643  1704
4  Way | 18829 12624  9032  6283  4029 15753  9779  6422  4244  2705 14818  9082  5287  3146  1835 15230  9383  5089  2757  1380 17293 11028  6278  3033  1377
8  Way | 18883 12362  8991  6145  4043 15806  9513  6375  4141  2716 14758  8785  5085  3081  1820 15564  9194  4921  2590  1400 17926 10761  5847  2797  1305
```

Assignment 2: Cache Simulator

- **Program language**
  - Use C language

- **Grading policy**
  - 2 open cases (trace1.txt, trace2-short.txt)
    - FIFO (8 points each)
    - LRU (12 points each)
    - BEST (5 points each)
  - 2 hidden cases (under 2,000,000 lines)
    - FIFO (8 points each)
    - LRU (12 points each)
    - BEST (5 points each)
  - Total: 100 points

Assignment 2: Cache Simulator

- **Deadline & late submission policy**
  - Submission deadline: December 17, 23:59:59
  - Even 1 second late counts as late
  - A 20% deduction will be applied for each day the submission is late

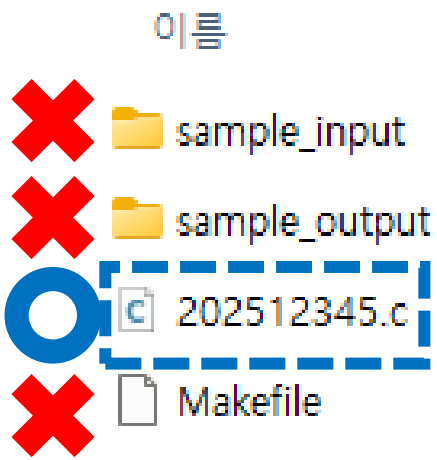
- **AI, LLM**
  - Cursor, chatGPT, Gemini, …
  - Feel free to use!
  - But…

Assignment 2: Cache Simulator

- **The assignment material will also be included on the exam**
  - Simply clicking (딸깍) will not earn you a good score on the final
  - You can use AI, but make effort to solve the problems on your own first
  - I believe that using AI effectively is also an important skill in this era
  - However, do not copy directly— it is my duty to give you an F

How to submit?

- ~~**Rename your file "main.c" to "[YourStudentID].c"**~~
  - E.g., "main.c" → "202512345.c"
  - Ajou Bb ignores such file name renaming…
- **Submit "[YourStudentID].c"**
- **If the format is incorrect, 20% of the score will be deducted**
  - **No excuse!**



Small tips

- **Your solution will be 700+ lines of code**
- **Decide the structure of i-cache and d-cache**
  - Refer to `print_two_cache_state`
- **Use the "diff" command and I/O redirection**
  - How to use? → $ man diff
  - ">" & "<" → refer to the provided Makefile & study them yourself
- **As for the NEW policy, remember that it only needs to outperform FIFO and LRU under the given conditions**

Recommended environment

- **You will use “gcc”, so you need Linux**
  - Recommended: Windows WSL
  - WSL’s basic mirror server would be “archive.ubuntu.com” or something (very slow), you have to change it to Korean mirror server (recommended: KAIST server)
  - Do it your self
    - Hint

    1. /etc/apt/sources.list.d/ubuntu.sources or /etc/apt/sources.list
    2. http://ftp.kaist.ac.kr/ubuntu/

  - Other options: Direct installation, dual booting, virtual machine, …