# Computer Architecture - CS2323. Autumn 2024

## Lab-7 (Cache Simulator)

**-----------------------------------------------------------------------------------------------------------------**

### Problem Statement:

Extend your [Lab4 (RV64I simulator)](https://docs.google.com/document/d/1up4f4AjflKpoHE5VvZHARoxk5ljz6_j8jv7mH1VxX60/edit?usp=sharing) code to support cache simulation to identify which accesses would cause a cache hit and which accesses would cause a cache miss when an assembly program is executing. To support cache simulation within your simulator, the following new commands starting with cache\_sim should be added to the Lab4 submission:

1. cache\_sim enable config\_file - Enables cache simulation for D-cache within the simulator. Uses parameters from config\_file for cache settings (see format below).
2. cache\_sim disable - Disables cache simulation. This is the default state at the start of the simulator.

The above 2 commands can not be executed while a file is being executed or loaded.

1. cache\_sim status - prints the enable/disable status of the D-cache simulation. If cache simulation is enabled, it also prints the cache configuration values in the format:

Cache Size: 32168  
Block Size: 16  
Associativity: 8  
Replacement Policy: LRU  
Write Back Policy: WT

1. cache\_sim invalidate - invalidates all entries of the D-cache.
2. cache\_sim dump myFile.out - writes all the current D-cache entries to the file “myFile.out” in the following format. Notice that only the valid entries have been printed in order of their Set value.

Set: 0x00, Tag: 0x100, Clean

Set: 0x01, Tag: 0x123, Clean

Set: 0x01, Tag: 0x736, Dirty

Set:0x02, Tag: 0x145, Dirty

Set:0x10, Tag: 0x321, Clean

1. cache\_sim stats - prints the cache statistics for the executing code at the current instance in the format:  
     
   D-cache statistics: Accesses=100, Hit=93, Miss=7, Hit Rate=0.93

If cache simulation is enabled, your program should generate a file named **~~cache.output~~ filename.output** where filename is the name of the executed file without the extension**,** which should contain cache simulation data in a format as shown:

### Example output:

R: Address: 0x20202, Set: 0x02, Miss, Tag: 0x202, Clean

W: Address: 0x10306, Set: 0x06, Hit, Tag: 0x103, Dirty

R: Address: 0x20511, Set: 0x11, Miss, Tag: 0x205, Clean

Also, when invoking the simulator’s “run” command, you should print the cache statistics at the end.   
  
If you run multiple files in a single simulator instance, say file1 and file 2, the output file should be generated independently for each executed file.   
The output file should be named file1.output and file2.output.

The program should allow customization for various cache parameters as follows, which are provided as a config file in a specific format:

1. Total size of the cache (specified in Bytes)
2. Size of each cache block (specified in Bytes)
3. Associativity (1: Direct mapped, 0: Fully associative, any other number: set associative) - up to 16
4. Replacement policy (FIFO, LRU, RANDOM)
5. Write policy (WB: WriteBack, WT: WriteThrough). Assume No-Allocate for WT and Allocate for WB.

The cache configuration is provided as an input file (e.g., cache.config) in the following format:

SIZE\_OF\_CACHE (number)

BLOCK\_SIZE (number)

ASSOCIATIVITY (number)

REPLACEMENT\_POLICY (FIFO or LRU or RANDOM)

WRITEBACK\_POLICY (WB or WT)

### Example config file:

32168

16

8

LRU

WT

For simplicity (if it helps), assume the following bounds and steps:

1. Cache size: max. 1 MB, always a power of 2
2. Block size: max. 64 Bytes, always a power of 2
3. Associativity: max. 16, always a power of 2
4. Input address: always 20 bits long. Assume it to be a 32-bit integer with the first 12 bits set to 0. So, only 20-bits are valid values.

You could approach the problem in multiple sub-parts, as mentioned below, with partial weightages for each part.

1. **Part-1:** Only support read access modeling for a direct-mapped cache
2. **Part-2:** Part-1 + caches with associativity + FIFO, LRU and RANDOM replacement policies
3. **Part-3:** Part-2 + write access modeling

### Submission instructions:

The assignment is to be done in the same project group as the Lab4. Submit your code and a Makefile, which will compile your code and generate an executable named riscv\_sim. Prepare a short report on your coding approach and what all you did to test your code to be correct. You should clearly indicate which parts (part-1, 2, 3) are supported by your code.

ONLY ONE MEMBER OF A GROUP SHOULD MAKE THE SUBMISSION.

Submit a zip file (Lab7\_CSYYBTECHZZZZZ\_CSYYBTECHXXXXX.zip) containing the following:

1. Source files of your code
2. Makefile - A makefile that would compile your code and generate an executable named riscv\_sim.
3. README - Containing information about various files in the directory, usage instructions, etc.
4. Test Cases - if you tried out specific input files.
5. report.pdf - a short report on your implementation/coding approach and what all you did to test your code to be correct.

**Demo**: We may later conduct a demo where you will show the functioning of your code to the TAs and accordingly be assigned marks.

**Examples:**

1. **Input1.s:**

| **.data**  **.dword 10, 20, 30, 40, 50**  **.text**  **lui x3, 0x10**  **ld x4, 0(x3)**  **ld x4, 8(x3)**  **ld x4, 16(x3)**  **ld x4, 24(x3)**  **ld x4, 32(x3)** |
| --- |

**config.txt:**

| **256**  **16**  **1**  **LRU**  **WB** |
| --- |

**Command Line Interface:**

| cache\_sim enable config.txt  load input.s  run  Executed ... PC=...  Executed ... PC=...  Executed ... PC=...  Executed ... PC=...  Executed ... PC=...  Executed ... PC=...  D-cache statistics: Accesses=5, Hit=2, Miss=3, Hit Rate=0.40  cache\_sim dump filename.ext |
| --- |

**input1.output:**

| R: Address: 0x10000, Set: 0x0, Miss, Tag: 0x100, Clean  R: Address: 0x10008, Set: 0x0, Hit, Tag: 0x100, Clean  R: Address: 0x10010, Set: 0x1, Miss, Tag: 0x100, Clean  R: Address: 0x10018, Set: 0x1, Hit, Tag: 0x100, Clean  R: Address: 0x10020, Set: 0x2, Miss, Tag: 0x100, Clean |
| --- |

**filename.ext**:

| Set: 0x0, Tag: 0x100, Clean  Set: 0x1, Tag: 0x100, Clean  Set: 0x2, Tag: 0x100, Clean |
| --- |

1. **Input2.s**

| **.data**  **.dword 20, 30, 40, 50, 60**  **.text**  **lui x3, 0x10**  **sd x0, 0(x3)**  **ld x4, 8(x3)**  **ld x4, 0(x3)**  **ld x4, 16(x3)**  **ld x4, 24(x3)** |
| --- |

**config.txt:**

| **1024**  **16**  **1**  **FIFO**  **WT** |
| --- |

**Command Line Interface:**

| cache\_sim enable config.txt  load input.s  run  Executed ... PC=...  Executed ... PC=...  Executed ... PC=...  Executed ... PC=...  Executed ... PC=...  Executed ... PC=...  D-cache statistics: Accesses=5, Hit=2, Miss=3, Hit Rate=0.40 |
| --- |

**input2.output:**

| W: Address: 0x10000, Set: 0x0, Miss, Tag: 0x40, Clean R: Address: 0x10008, Set: 0x0, Miss, Tag: 0x40, Clean  R: Address: 0x10000, Set: 0x0, Hit, Tag: 0x40, Clean  R: Address: 0x10010, Set: 0x1, Miss, Tag: 0x40, Clean  R: Address: 0x10018, Set: 0x1, Hit, Tag: 0x40, Clean |
| --- |

**Clarifications and Questions:**

1. What should be written in the Clean/Dirty column if the policy is WT?  
   If the policy is WT, all cache entries can be assigned Clean.
2. You can assume that no read/write will span more than 1 cache block. You can emit an error message for an erroneous input.
3. ~~The entries of the cache have to be invalidated along with the memory at the end of execution of a file.~~ The cache is cleared along with the registers whenever a new .s file is loaded in the simulator. (<https://moodle.cse.iith.ac.in/moodle/mod/forum/discuss.php?d=294>)
4. You also have to simulate the memory in accordance with the cache. For example, if we have a write-back policy, any dirty memory block would have an older value in memory.
5. In the case of write through no allocate policy, any write to a memory block will not go through or be stored in the cache.

**Note**: Submissions are subject to similarity check and hence submit only your own work. Any similarity will be strictly dealt with.

Your submissions will be evaluated on Ubuntu 20.04 machines and hence you must check them to be working on such a setup. If you do not have those machines, please visit the B-524 lab.

**ONLY FOR THOSE WHO DID ONLY R-FORMAT IN LAB3 (OTHERS CAN IGNORE THIS).**

1. You should implement a cache simulator which takes a config file as input and generates a cache.output as in the above defined statement.
2. Since load/store instructions are not implemented in your previous labs, the access sequence is provided in another file (cache.access) in the following format:

Mode: Address

Mode: Address

…

Mode refers to access mode: R: Read or W: Write

The address refers to the memory address that is accessed

Example input file:

R: 0x00203302

R: 0x00202011

W: 0x00203302

1. Please name the executable ‘riscv\_r\_sim’ to simplify the checking process.